Chapter 4
Advanced Selection Operation for Differential Evolution Algorithm

Pravesh Kumar and Vanita Garg

1 Introduction

The term optimization refers to the process of identifying the most viable solution and thereby reaching the extreme point of the objective functions. The goal of identifying these optimal solutions is typically to design a problem to minimize total cost or to maximize probable reliability, among other things. Because of the high quality of optimal solutions, we place a high value on optimization approaches in scientific, engineering, and business decision-making situations. We can divide optimization approaches into two categories: classic and nontraditional methods.

Traditional methods may not be able to solve such issues due to the presence of nonlinearity, non-continuity, non-differentiability, and many local/global optimums.

Recently, many nature-inspired and evolutionary algorithms have been created to handle optimization challenges.

Genetic algorithms, ant colonies, particle swarm optimization, differential evolution algorithm, artificial bee colony, teaching learning-based algorithm, Jaya algorithms, and firefly algorithms are some of the most common algorithms in use today.

Biogeography-based optimization algorithm also comes in the category of nature-inspired algorithms. Garg and Deep have proposed LX-BBO in [47]. LX-BBO is extended for solving constrained optimization problems in [49]. The same algorithm is proposed after applying mutation strategies in [48].

Differential evolution (DE) algorithm was introduced by Storn and Price in 1997 [1]. It is a prominent, stochastic, and population-based optimization algorithm, where the population consists of many individuals, each of which represents a potential solution to the optimization problem. DE produces offspring solution by

P. Kumar
Rajkiya Engineering College Bijnor (AKTU Lucknow), Lucknow, Uttar Pradesh, India

V. Garg (✉)
Galgotias University, Greater Noida, Uttar Pradesh, India

D. Singh et al. (eds.), *Design and Applications of Nature Inspired Optimization*, Women in Engineering and Science, https://doi.org/10.1007/978-3-031-17929-7_4

mutation, crossover, and selection operation, which are likely to be nearer to the optimal result.

A few of the advantages that DE has over other nature-inspired algorithms are that it is compact, has a small number of control parameters, and is easy to implement without requiring any special knowledge. Some of its more advanced capabilities include the capacity to handle nonlinear, discontinuous, non-differentiable, and multi-objective functions, among other aspects. Engineers and scientists have successfully used DE to solve a wide range of real-world problems in the engineering and science fields. Examples include the following: engineering design difficulties, pattern identification, power engineering, image processing, and noise detection.

Premature convergence or evolution stagnation, which is fatal to an algorithm that relies on population difference, is inevitable as the number of generations increases in a population. Control settings affect DE's performance as well [2]. In order to find the optimal value for these control parameters for various optimization problems, several trials must be performed.

A few of the modified variants of DE during recent years are as follows: trigonometric mutation-based DE (TDE) [3], fuzzy adaptive DE (FADE) [4], modified differential evolution (MDE) [5], DE with random localization (DERL) [6], self-adapting control parameter-based DE (jDE) [7], opposition-based DE (ODE) [8], accelerating differential evolution [9], mixed mutation strategy embedded DE [10], self-adaptive DE (SADE) [11], adaptive DE with optional external archive (JADE) [12], DE with neighborhood mutation [13], DE with Cauchy mutation (CDE) [14], clustering-based DE (CDE-Cai) [15], learning enhanced DE (LeDE) [16], DE with proximity-based mutation [17], enhanced mutation strategy (MRLDE) [18], DE with adaptive population tuning scheme [19], DE with dynamic parameters selection [20], control parameter and mutation-based DE (CDE) [21], multiple mutation strategies-based DE [22], multi-population-based DE [23], collective information-based DE [24], adaptive learning mechanism-based DE [25], novel DE for constrained [26], parameter adaptation schemes for DE (PaDE) [27], random perturbation modified DE [28], DE with dual preferred learning mutation [29], DE with neighborhood-based adaptive evolution mechanism [30], self-adaptive mutation DE with PSO [31], parameter adaptive-based DE [32], DE with adaptive multi-population inflationary [33], and dual-strategy-based DE (IDE) [34].

A well-prepared literature review of enhancement and applications of differential evolution algorithm can also be found in [35–38, 46].

In this chapter, a novel modification in selection operation for DE named "DE with advanced selection operator (DEaS)" is proposed. DEaS works in two ways: first, it reuses the rejected trial vectors by their superiority, and second, it operates selection operation in a single array strategy proposed by Babu and Angira in MDE [5].

Furthermore, this newly proposed selection operation is integrated with two other DE-enhanced variants such as DERL [6] and MRLDE [18] and named it DERLaS and MRLDEaS, respectively. The evaluation of proposed modifications has executed on benchmark problems as well as real-life applications. The numerical and

statistical significance of proposed variants is discussed later in the chapter. Here, we would also like to mention that this work is an extended version of our previous studies carried out in [39, 40].

Organization of the chapter is as follows: The introduction of basic DE algorithm is given in Sect. 2. The proposed advanced selection operation and functioning of DERLaS and MRLDEaS are explained in Sect. 3. In Sect. 4, benchmark functions, real-life applications, and experimental settings are given.

The results and comparison of the algorithms is given in detail in Sect. 5, and the chapter is finally concluded in Sect. 6 with its future scope.

2 Basic Differential Evolution (DE)

In this section, the basic concept and working of differential evolution algorithm (DE/rand/1/bin) is illustrated. DE works in four steps, such as initialization, mutation, crossover, and selection operation, for which the details are presented as below:

(i) *Initialization Phase*: The first phase of DE algorithm is to initialize a uniform random set of solutions called population. Here, each solution is a *d*-dimensional vector also called an individual. Equation 4.1 generate the initial population $Pop = \{ P_i^{(gen)}, i=1,2,...N\}$ *of d*-dimensional *N* vectors.

$$P_i^{(0)} = P_{LB} + rand\,(0,\,1) \times [P_{UB} - P_{LB}] \tag{4.1}$$

Here:

- *rand (0, 1)* is the uniform random number between 0 and 1
- P_{UB} and P_{LB} are the upper and lower bound, respectively, of search space.

(ii) *Mutation Phase*: To perform the mutation operation, three mutually separate vectors, say $P_a^{(gen)}$, $P_b^{(gen)}$, and $P_c^{(gen,}$ are selected at random from $Pop = \{P_i^{(gen)}, i=1,2,...N\}$ corresponding to a target vector P_i^{gen}, such that $a \neq b \neq c \neq i$, and then a new vector $M_i^{gen} = (m_{1,i}, m_{2,i}. . ., m_{d,i})$, also called mutant or perturbed vector, is generated by Eq. 4.2:

$$M_i^{(gen)} = P_a^{(gen)} + SF \times \left[P_b^{(gen)} - P_c^{(gen)}\right] \tag{4.2}$$

Here:

- *SF* is the scaling factor and use to controls the amplification of the difference $[P_b^{(gen)} - P_a^{(gen)}]$.
- It may have a value between [0 and 2] to as per suggested by Storn and Price.

(iii) *Crossover Phase*: In crossover operation, a trial vector $T_i^{(gen)} = (t_{1,I}, t_{2,i}. . ., t_{d,i})$ is generated corresponding to the target and mutant vector. It is defined in Eq. 4.2:

$$t_{j,i}^{(gen)} = \begin{cases} m_{j,i}^{(gen)}, \; if \; CR < rand_j \forall j = = I_j \\ p_{j,i}^{(gen)} \quad \text{otherwise} \end{cases} \tag{4.3}$$

Here,

- CR is the crossover constant having value between 0 and 1.
- $rand_j$ [0, 1] is the uniform random number between 0 and 1.
- I_j : randomly chosen index from 1, 2, ...d. to make sure that at least one component of trial vector will pick from mutant vector.

(iv) *Selection Phase*: Selection operation is performed at the end of any generation of *DE* and ensures that fitter vector has chosen for next generation between trial vector and target vector. Equation 4.4 describes the selection operation between trail and target vector.

$$P_i^{(gen+1)} = \begin{cases} T_i^{(gen)}, \quad if \, fun\left(T_i^{(gen)}\right) < fun\left(P_i^{(gen)}\right) \\ P_i^{(gen)} \quad \text{otherwise} \end{cases} \tag{4.4}$$

3 Proposed Modification

Advance Selection Strategy

The basic selection technique of DE is based on a tournament selection between trail and target vector. The vector with the lowest fitness value is considered as a winner and goes to next-generation population. Here, it can be noticed that during this type of one-on-one competition, a rejected trial vector may have better fitness value than some other target vectors in the population, but there is no additional feature for such rejected trail vectors to prove their efficiency in the space. Also, every time-rejected trial vector takes up extra space in computer memory and may lead to low computer processor speed. Therefore, some additional inspection measures should be done so that more of these fitted trial vectors can be selected and also reduce extra space in memory. Our proposed advance selection approach offers such additional characteristic to the old selection operation of DE.

In advance selection operation, first we perform the old selection operation by comparing trail and target vector and chose the fittest vector for the next generation.

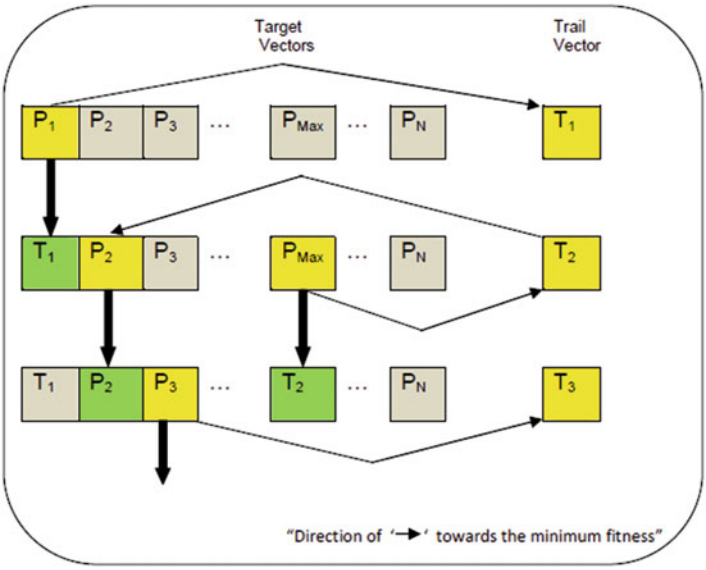


Fig. 4.1 Working design of advance selection operation

In case of rejection of trail vector, it further compares with the worst vector (having highest fitness value) of the population and swap on that place, if it has a lower fitness value than the worst vector. After updating the worst vector, this process will continue for the next trail vector.

Furthermore, a single array strategy proposed by Babu and Angira [5] also has employed with our modified selection technique. The single array strategy helps to reduce the memory space. Consequently, the proposed selection operation compresses the searching region in every generation, and hence it boosts up in the convergence speed to achieve the desire result.

The advance selection operation is demonstrated graphically in Fig. 4.1.

Pseudo Code of DEaS Algorithm

BEGIN

Generate uniformly distributed random population $Pop = \{P_i^{(gen)}, i=1,2,...N\}$.

FOR $i=1:N$

{

$P_i^{(0)} = P_{LB} + rand(0,1)*(P_{UB} - P_{LB})$

}/*END FOR */

Evaluate $fun\{P_i^{(gen)}\}$

WHILE (Termination criteria is met)

(continued)

```
{
FOR i=1:N
   {
      Execute mutation operation by Eq-2
      Execute crossover operation and generate trial vector T_i^(gen)
      Evaluate fun(T_i^(gen))
//** Advance Selection Operation with Single Array strategy***////
      IF (fun(T_i^(gen))< fun(P_i^(gen)))
            {
      P_i^(gen+1)= T_i^(gen)
      }
   ELSE
{
      IF (fun(T_i^(gen))< fun(P_max^(gen)))
      {
            P_max^(gen)= T_i^(g)
      }
      Update P_max^(gen)
}
      }/* END FOR loop*/
} /* END WHILE loop*/
END
```

Proposed DERLaS and MRLDEaS

In order to verify the effect of proposed advance selection operation on other variants, it has embedded with DERL and MRLDE. A short description of DERL and MRLDE is given as below:

DERL [6]: It is a mutation-based enhanced variants of DE. Here, first three mutually separate vectors are select randomly from the population and then chose the best fitted vector among these three has used as a base vector in the mutation operation. A detail description and its effectiveness of can be read in its original paper [6].

MRLDE [18]: It is our previously proposed DE variant, which is also a mutation-based enhanced variant of DE. In MRLDE, the whole population is divided into three regions, and then the base vector is selected randomly from the region of the best individuals. A details explanation and effectiveness of MRLDE in solving various real-life optimization problems can be study in the literature [41–45].

The pseudo code of proposed DERLas and MRLDEaS is given below:

Pseudo Code of DERLaS and MRLDEaS Algorithm

BEGIN

Generate uniformly distributed random population $Pop = \{P_i^{(g)}, i=1,2,...N\}$.

WHILE (Termination criteria is met)

{

FOR $i=1:N$

{

*/*Working of DERLaS*/*

Select three random vectors $P_a^{(gen)}$, $P_b^{(gen)}$ and $P_c^{(gen)}$

Select best of $P_a^{(gen)}$, $P_b^{(gen)}$ and $P_c^{(gen)}$ and use as base vector in mutation

operation.

*/*Working of MRLDEaS*/*

Divide Population in three sub-regions of N_1,N_2 and N_3 size of best, medium

and worst sub-region respectively according fitness.

Select $P_a^{(gen)}$, $P_b^{(gen)}$ and $P_c^{(gen)}$ from best, medium and worst sub-region

respectively.

Execute Mutation Operation

Execute crossover operation

Execute Advance selection operation

}/* END FOR loop*/

} /* END WHILE loop*/

END

4 Experimental Settings

In this section, selected benchmark problems, real-life applications, performance criteria, and parameter settings for the evaluation of proposed variants are given.

Test Functions

Fifteen traditional benchmark problems and three real-life applications are selected from the literature to test the effect of the proposed advance selection on DE, DERL, and MRLDE and also the comparison with the other enhanced DE variants. The mathematical models and properties of these are given in Appendix.

Performance Criteria

The following evaluation criterias are taken from various literatures [12, 16, 18], and to evaluate the performance and comparisons of our proposed algorithms

Number of function evaluation (NFE): The NFEs are obtained when a fixed accuracy (VTR) is attained before reaching the maximum NFE. That is, we set the termination criteria as $|F_{opt} - F_{globqal}| \leq VTR$ and record the average NFE of successful run over 50 runs.

Error: The average and standard deviation of the minimum error $f(P)$-$f(P^*)$ is observed after fixed maximum NFEs are attained of 50 runs.

Acceleration rate (AR): It is used to compare the convergence speeds of two algorithms. For two algorithms A and B, AR is defined as follows:

$$AR = \left(1 - \frac{NFE_{\mathrm{B}}}{NFE_A}\right)\%$$

Convergence graphs: The convergence graphs demonstrate the performance graphically in terms of fitness value with respect to iteration in any run.

Parameter Setting

In the study, similar parameter settings as per Table 4.1 have been taken for each algorithm for a fair evaluation and comparison.

All experiments are carried out on a computer with 2.66 GHz 10th Gen Intel® Core™ i3, 4GB of RAM, and software Dev-C++ was used to implement the programming.

Table 4.1 Parameter setting [12, 16, 18]

Size of population (N)	100
Dimension (d) for benchmark functions	30
Scale factor (SF) and	0.5
Crossover rate (CR)	0.9
Max NFE	500,000
Size of N_1, N_2, and N_3 for MRLDE and MRLDEaS	20%, 40%, and 40%, respectively
Total run	50

5 Result and Discussion

Result on Benchmark Problems

(a) *Effect of Advance Selection on DE, DERL, and MRLDE*

In this section, an analysis of the effect of proposed selection operation on DE, DERL, and MRLDE algorithm has been carried out. The results are given in numerical and statistical significance terms. The numerical results are given in terms of the average NFE and average error with a standard deviation of 50 runs in Tables 4.2 and 4.3, respectively, while the results for acceleration rate (AR) and statistical significance are given in Table 4.4.

From Table 4.2, it is clear that the algorithm with advance selection, i.e., DEaS, DERLaS, and MRLDEas, takes less NFEs compared to their original variants, respectively, for all function, except function F_8 and F_9. None of the algorithm has obtained the desired accuracy for function F_8 and F_9. The total NFEs obtained by DE, DERL, and MRLDE are 2142410, 1233200, and 768000, respectively, while the total NFEs obtained by DEaS, DERLaS, and MRLDEaS are 1886900, 985900, and 670000, respectively, for all function, except function F_8 and F_9.

Now from Table 4.3, it can be easily observed that results are more accurate obtained by algorithms with advance selection operation in terms of average error also. Here, we can obtain the results for function F_8 and F_9, also for which Deas and DERLaS give better results; however, MRLDE gives a minimum error than MRLDEas for function F_9. For function F_6, all algorithms perform the same.

In Table 4.4, results are given in terms of AR for NFEs given in Table 4.2 and statistical significance on average error and standard deviation obtained in Table 4.3.

Table 4.2 Numerical results in terms of the average NFEs of 50 runs

Fun	VTR	DE	DEaS	DERL	DERLaS	MRLDE	MRLDEaS
F_1	10^{-09}	105600	**94100**	54800	**46600**	40400	***33400***
F_2	10^{-09}	175400	**158800**	91900	**77600**	66900	***57800***
F_3	10^{-09}	404200	**389500**	215900	**188900**	156800	***136600***
F_4	10^{-03}	137900	**116600**	147200	**110500**	65200	***62500***
F_5	10^{-09}	440400	**376800**	271900	**184400**	138500	***122900***
F_6	10^{-09}	32680	**29500**	19000	**14500**	12900	***10400***
F_7	10^{-02}	200390	**146600**	92500	**77900**	38600	***33600***
F_8	10^{-03}	NA	NA	NA	NA	NA	NA
F_9	10^{-03}	NA	NA	NA	NA	NA	NA
F_{10}	10^{-09}	164600	**146400**	86200	**72400**	62600	***53900***
F_{11}	10^{-09}	108500	**95600**	58100	**48300**	40400	***35900***
F_{12}	10^{-09}	95600	**83400**	49400	**41600**	37100	***31900***
F_{13}	10^{-09}	102200	**93400**	55600	**47400**	39900	***33500***
F_{14}	10^{-09}	68400	**62300**	35600	**28600**	27900	***22700***
F_{15}	10^{-09}	106600	**93900**	55100	**47200**	40800	***34900***

Table 4.3 Numerical results in terms of the average error and standard deviation of 50 runs

Fun	Max NFE	DE	DEaS	DERL	DERLaS	MRLDE	MRLDEaS
F_1	150K	5.80E-13 (2.62E-14)	**5.80E-16 (2.62E-16)**	2.43E-29 (2.82E-29)	**5.80E-36 (2.62E-36)**	8.35E-43 (8.61E-43)	**1.67E-50 (2.31E-50)**
F_2	200K	3.06E-10 (6.70E-10)	**1.12E-11 (5.34E-12)**	1.15E-20 (4.08E-20)	**9.11E-25 (7.44E-25)**	5.15E-29 (6.61E-29)	**3.37E-34 (4.19E-34)**
F_3	500K	2.79E-11 (4.21E-11)	**3.56E-12 (3.12E-13)**	5.22E-25 (3.32E-25)	**2.88E-28 (5.43E-28)**	1.12E-37 (5.04E-38)	**4.25E-40 (3.13E-40)**
F_4	500K	2.41E-01 (9.45E-02)	**3.50E-03 (1.11E-03)**	4.83E-07 (2.64E-07)	**2.88E-10 (5.43E-10)**	2.56E-28 (2.23E-28)	**3.76E-31 (4.33E-31)**
F_5	200K	1.17E+01 (7.41E-01)	**5.75E+00 (2.13E-01)**	1.01E-03 (2.75E-03)	**6.22E-11 (4.45E-11)**	2.16E-19 (2.43E-19)	**3.12E-24 (1.43E-24)**
F_6	50K	**(0.0E+00) (0.0E+00)**	**(0.0E+00) (0.0E+00)**	**(0.0E+00) (0.0E+00)**	**(0.0E+00) (0.0E+00)**	**(0.0E+00) (0.0E+00))**	**(0.0E+00) (0.0E+00)**
F_7	300K	2.81E-02 (5.61E-03)	**4.91E-03 (4.88E-04)**	**2.41E-03 (4.81E-04)**	2.64E-03 (6.33E-04)	2.09E-03 (9.65E-04)	**2.01E-03 (5.47E-04)**
F_8	500K	6.96E+03 (1.18E +03)	**4.73E+03 (8.51E +02)**	1.11E+03 (6.20E +02)	**3.55E+02 (3.25E +02)**	6.95E+02 (8.21E +02)	**1.18E+02 (1.31E +02)**
F_9	500K	7.97E+01 (1.83E +01)	**6.09E+01 (2.23E +01)**	1.52E+01 (4.61E-01)	**1.06E+01 (3.66E +00)**	***9.01E+00 (4.16E +00)***	**1.62E+01 (9.38E-01**
F_{10}	50K	6.41E-02 (3.43E-03)	**2.25E-02 (2.61E-03)**	1.18E-04 (5.36E-06)	**1.23E-05 (3.36E-06)**	9.18E-06 (4.28E-07)	**9.96E-08 (2.45E-08)**
F_{11}	50K	2.11E-01 (1.40E-02)	**1.18E-02 (3.09E-03)**	2.88E-06 (3.21E-06)	**1.23E-09 (3.36E-09)**	1.60E-10 (2.08E-10)	**1.17E-14 (5.35E-14)**
F_{12}	50K	3.34E-03 (2.15E-03)	**1.12E-03 (1.05E-03)**	8.37E-08 (4.45E-08)	**3.42E-11 (4.56E-12)**	4.86E-13 (2.88E-13)	**1.59E-15 (3.46E-15)**
F_{13}	50K	2.45E-02 (1.25E-02)	**8.23E-03 (5.41E-03)**	9.19E-08 (3.53E-10)	**6.91E-10 (4.33E-10)**	1.11E-11 (2.05E-11)	**3.21E-14 (1.79E-14)**
F_{14}	50K	1.56E-06 (6.73E-06)	**3.81E-07 (4.53E-07)**	1.01E-11 (8.31E-12)	**8.83E-14 (1.25E-14)**	4.79E-16 (3.94E-16)	**2.20E-18 (1.52E-18)**
F_{15}	50K	3.18E-02 (7.91E-03)	**6.62E-03 (1.17E-03)**	2.79E-07 (1.07E-07)	**5.83E-10 (3.42E-10)**	1.07E-11 (6.5E-12)	**4.47E-14 (4.24E-14)**

From the table, we can see a fast convergent speed in terms of AR for each function by each algorithm with advance selection operation. The average AR of DEaS with respect to DE is 11.93%, AR of DERL with respect to DERLaS is 20.05%, and AR of MRLDEaS with respect to MRLDE is 12.76%. Here, AR of MRLDEas is also obtained with respect to DERLaS and DEaS, which are 64.49% and 32.04%, respectively.

The statistical significance of results is also presented in Table 4.4. DEaS performs statistical better than DE for all function, except F_6 and F_{14} for which there is no significance difference between the performances.

Similarly, DERLaS gives an equal performance for function F_6 and is significantly better for other functions compared with DERL.

Table 4.4 Numerical results in terms of acceleration rate (AR) and statistically significance

	DEaS/DE		DERLaS/ DERL		MRLDEaS/ MRLDE		MRLDEaS/ DEaS		MRLDEaS/ DERLaS	
Fun	AR	Sig.	AR	Sig	AR	Sig	AR	Sig.	AR	Sig
F_1	10.89	+	14.96	+	17.33	+	64.51	+	28.33	+
F_2	9.46	+	15.56	+	13.60	+	63.60	+	25.52	+
F_3	3.64	+	12.51	+	12.88	+	64.93	+	27.69	+
F_4	15.45	+	24.93	+	4.14	+	46.40	+	43.44	+
F_5	14.44	+	32.18	+	11.26	+	67.38	+	33.35	+
F_6	9.73	=	23.68	=	19.38	=	64.75	=	28.28	=
F_7	26.84	+	15.78	+	12.95	=	77.08	+	56.87	=
F_8	NA	+	NA	+	NA	+	NA	+	NA	+
F_9	NA	+	NA	+	NA	−	NA	+	NA	−
F_{10}	11.06	+	16.01	+	13.90	+	63.18	+	25.55	+
F_{11}	11.89	+	16.87	+	11.14	+	62.45	+	25.67	+
F_{12}	12.76	+	15.79	+	14.02	+	61.75	+	23.32	+
F_{13}	8.61	+	14.75	+	16.04	+	64.13	+	29.32	+
F_{14}	8.92	=	19.66	+	18.64	+	63.56	+	20.63	+
F_{15}	11.91	+	14.34	+	14.46	+	62.83	+	26.06	+
Avg *w/l/t*	11.93	**13/0/2**	20.05	**14/0/1**	12.76	**12/1/2**	64.49	**14/0/1**	32.04	**12/1/2**

"+", "−", and "=" mean significantly better, lower, and equal, respectively

MRLDEaS performs significantly better than MRLDE for all functions, except F_6, F_7, and F_9. In the case of F_6 and F_7, there is no significant difference between the performances of both, while MRLDE is significantly better than MRLDEaS in the case of function F9.

The last row of Table 4.4 shows the total number of *win/loss/tie* performance of algorithms on all functions. The *w/l/t* performance of DEaS *vs* DE is 13/0/2, DERLaS *vs* DERL is 14/0/1, MRLDEaS *vs* MRLDE is 12/1/2, MRLDEaS *vs* DEaS is 14/0/1, and MRLDEaS *vs* DERLas is 12/1/2.

(b) *Comparison of MRLDEaS with Other Enhanced DE Variants*

In this section, comparison of MRLDEas is discussed with some other well-known enhanced DE variants, such as jDE [7], ODE [8], CDE-Cai [15], and LEDE [16]. The comparison is given in Table 4.5 in terms of NFE. The results for ODE, jDE, CDE-Cai, and LeDE are taken from [16]. All parameter settings have also taken similar from [16] for fair comparison.

From the table, we can see that our proposed MRLDEaS takes less NFE for all benchmark function, except F_7, F_8, and F_9 for which CDE-Cai and JDE perform best from others. The corresponding rank is also given for each function in the table. The average rank of ODE, jDE, CDE-Cai, and LEDE is 4.54, 3.62, 2.92, and 2.15, respectively, while the rank of MRLDEaS is 1.77, which proved the effectiveness of it compared to all others.

Table 4.5 Comparison of MRLDEaS with ODE, jDE, CDE, and LEDE

	NFE					Function-wise ranks				
Fun	ODE	jDE	CDE-Cai	LeDE	MRLDEaS	ODE	jDE	CDE-Cai	LeDE	MRLDEaS
F_1	67524	60000	54121	49494	**33400**	5	4	3	2	**1**
F_2	140170	83000	84295	77464	**57800**	5	3	4	2	**1**
F_3	489210	340000	166545	140176	**136600**	5	4	3	2	**1**
F_4	145880	300000	177268	157499	**141400**	2	5	4	3	**1**
F_5	NA	NA	315282	282972	**122900**	4.5	4.5	3	2	**1**
F_6	25008	23000	17869	17123	**10400**	5	4	3	2	**1**
F_7	60230	100000	**33275**	33302	33600	4	5	**1**	2	3
F_8	147472	**89000**	115163	111013	NA	4	**1**	3	2	5
F_9	190604	**120000**	184371	187813	NA	4	**1**	2	3	5
F_{10}	106694	91000	84920	76111	**53900**	5	4	3	2	**1**
F_{11}	79888	63000	56868	50579	**35900**	5	4	3	2	**1**
F_{12}	63710	55000	45803	41384	**31900**	5	4	3	2	**1**
F_{13}	63202	60000	50720	*46529*	**33500**	5	4	3	2	**1**
Average rank						4.54	3.62	2.92	2.15	**1.77**

Table 4.6 Wilcoxon sign rank test for MRLDEaS vs ODE, jDE, CDE, and LEDE

Algorithms		$\sum R^+$	$\sum R^-$	*W value*	*Critical value at 5% level*	Significance
MRLDEaS *vs*	**ODE**	70	21	21	21	+
	jDE	66	25	25	17	+
	CDE-Cai	65	26	26	21	+
	LeDE	65	26	26	21	+

In Table 4.6, a nonparametric Wilcoxon sign rank test is also performed to check the pair-wise comparison of MRLDEas with another algorithm. From the table, we can see that our proposed MRLDEaS provides an overall significance superior performance than ODE, jDE, CDE-Cai, and LeDE.

Result on Real-Life Application

In this section, the evaluation of the proposed variants on real-life applications is discussed. In Table 4.7, the results are obtained in terms of worst, best, mean, and standard deviation of fitness value in 50 runs. The best results obtained by algorithms are given in bold cases. We can see that the proposed variants with advance selection operation perform better than their original variants in all terms. The statistical test value is also given in the table, which also proved the significance of proposed variants over the original variants, respectively.

Convergence Graphs

In this section, the convergence speed of algorithms is represented graphically by the convergence graphs in Fig. 4.2. Here, convergence graphs are given for function F_1, F_2, F_5, F_{10}, F_{11}, and F_{14}. RF_1 and RF_2. From Fig. 4.2, we can easily observe that DE, DERL, and MRLDE obtain a fast convergence speed when applying proposed advance selection operation with these algorithms. We can also see that MRLDEaS provides a faster convergence speed compared to all other variants.

6 Conclusions

In the present chapter, an advance selection strategy for DE algorithm named DEaS is proposed. This advance selection operation gives an additional opportunity to the rejected trail vectors to prove their efficiency over other target vectors. This approach

Table 4.7 Numerical and statistical results for real-life applications

Fun	Max NFE	Fitness	DE	DEaS	DERL	DERLaS	MRLDE	MRLDEaS
***RF*$_1$**	30000	Worst	1.2E+01	**1.0E+01**	8.7E-10	**6.3E-14**	3.2E-10	***3.6E-17***
		Best	8.1E+00	**5.1E-03**	3.7E-14	**9.1E-18**	7.2E-22	***0.0E+00***
		Mean	1.0E+01	**3.9E+00**	3.1E-10	**1.2E-14**	8.1E-11	***7.3E-18***
		SD	8.3E+00	**1.2E+00**	4.2E-10	**2.5E-14**	1.6E-10	***1.4E-17***
		z-value	z = ***5.14***		z = ***5.22***		z = ***3.58***	
		Significance at 5% level	+		+		+	
***RF*$_2$**	1000	Worst	4.19092	**4.20318**	4.20757	**4.20977**	4.20877	***4.20989***
		Best	4.20489	**4.21198**	4.21279	**4.21282**	4.21231	***4.21346***
		Mean	4.19942	**4.20852**	4.21139	**4.21186**	4.21076	***4.21238***
		SD	6.1E-03	2.9E-03	1.9E-03	1.2E-03	1.4E-03	*1.3E-03*
		z-value	z = **9.53**		z = ***1.48***		z = ***6.0***	
		Significance at 5% level	+		=		+	
RF3	$d = 20$ MaxNFE = *100K*	Worst	3.7523	**2.1274**	1.7603	**0.9010**	0.8551	***0.8014***
		Best	2.3012	**2.0138**	1.5816	**0.6652**	0.6625	***0.6671***
		Mean	3.1328	**2.4820**	1.6321	**0.7877**	0.7523	***0.5423***
		SD	2.0435	**1.001**	1.4099	**0.0672**	0.0466	***0.0011***
		z-value	z = ***2.02***		z = ***4.23***		z = ***3.19***	
		Significance at 5% level	+		+		+	

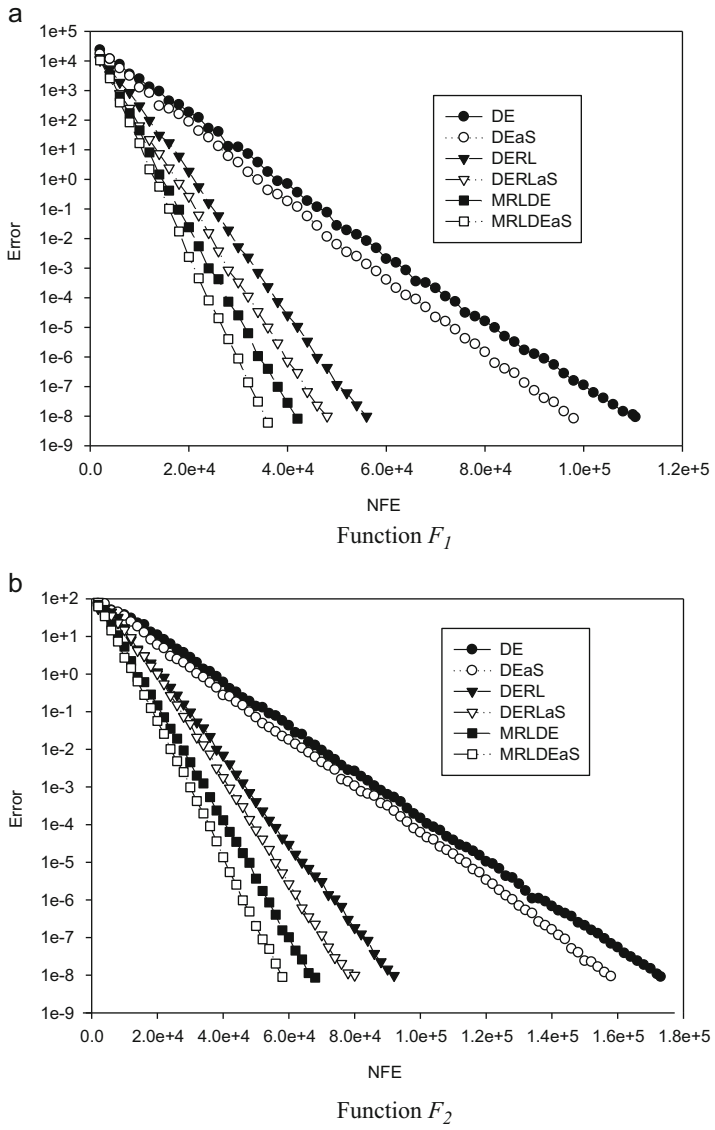
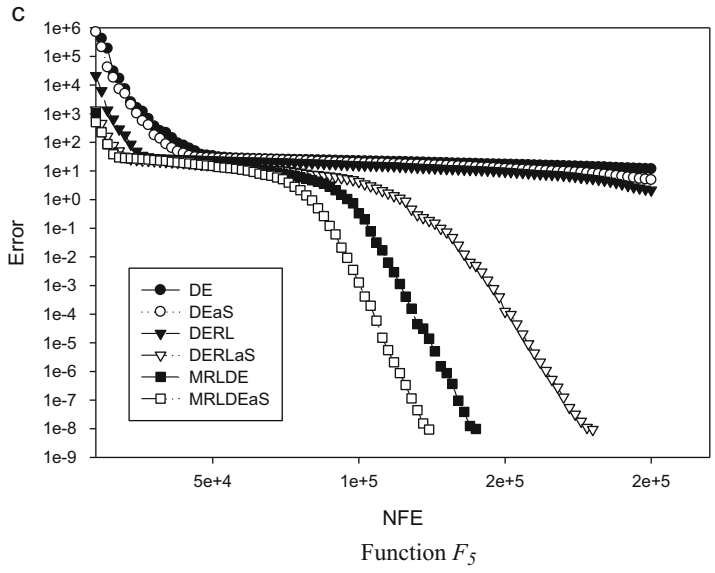


Fig. 4.2 Convergence graphs in terms of error and NFEs

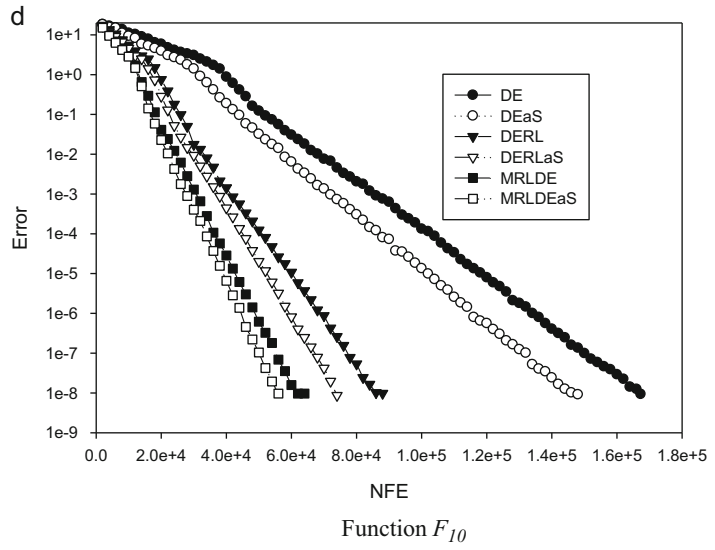
condenses the searching space in every generation and helps to obtain better convergence speed as well as diminishes the redundant memory space.

Next, the proposed selection operation is embedded with other enhanced variants, named DERLaS and MRLDEaS.

The performances of proposed variants are evaluated on 15 traditional benchmark problems and 3 real-life applications. The numerical results for DEaS, DERLaS, and



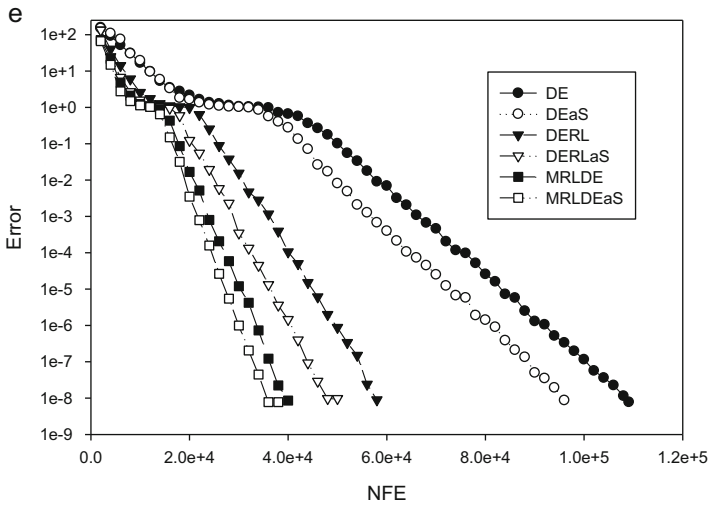
Function F_5



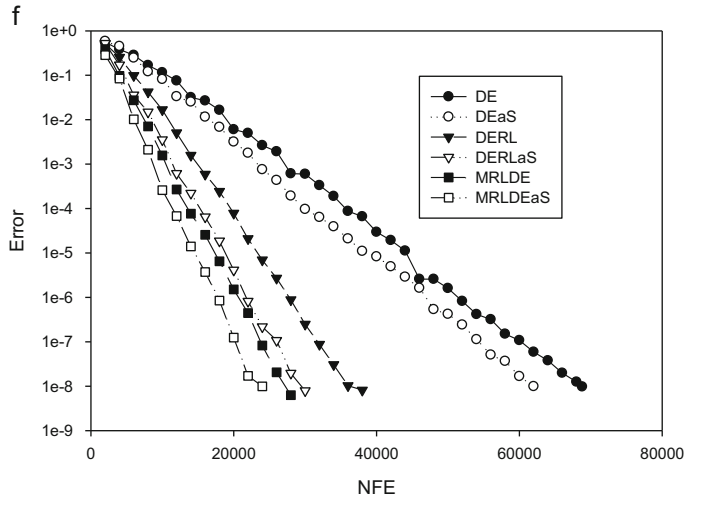
Function F_{10}

Fig. 4.2 (continued)

MRLDEaS are compared with the original variant DE, DERL, and MRLDE, respectively. Furthermore, the performance of MRLDEaS is also compared with other enhanced DE variants, such as ODE, jDE, CDE-Cai, and LeDE.



Function F_{11}

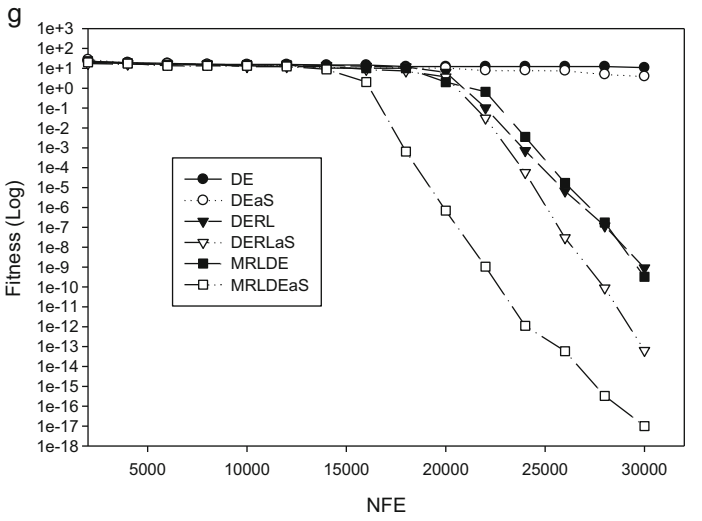


Function F_{14}

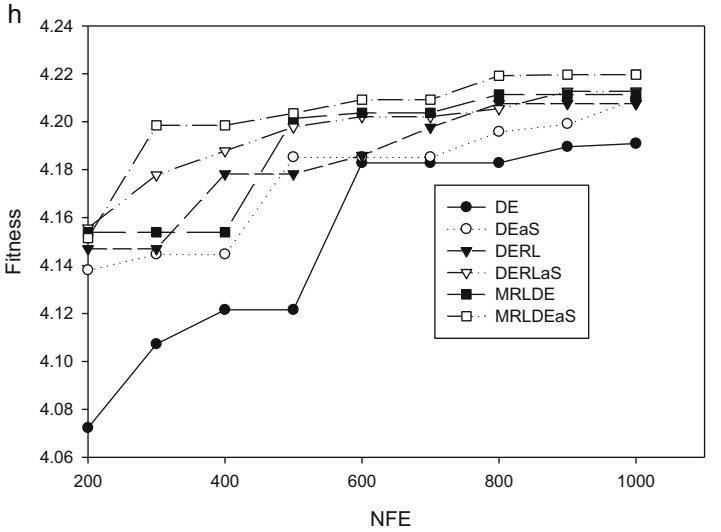
Fig. 4.2 (continued)

The numerical, statistical, and graphical results have proved the effectiveness and robustness of the proposed advance selection operation with DE and other variants DERL and MRLDE.

In future, the effect of this advance selection operation can be verified on other evolutionary algorithms for solving real-life optimization problems.



Real Life Application RF_1



Real Life Application RF_2

Fig. 4.2 (continued)

References

1. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
2. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans. Evol. Comput. **19**(2), 167–187 (2015)
3. Fan, H., Lampinen, J.: A trigonometric mutation operation to differentia evolution. J. Glob. Optim. **27**, 105–129 (2003)

4. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput. Fusion Found Meth. Appl. **9**(6), 448–462 (2005)
5. Babu, B.V., Angira, R.: Modified differential evolution (MDE) for optimization of non-linear chemical processes. Comput. Chem. Eng. **30**, 989–1002 (2006)
6. Kaelo, P., Ali, M.M.: A numerical study of some modified differential evolution algorithms. Eur. J. Oper. Res. **169**, 1176–1184 (2006)
7. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)
8. Rahnamayan, S., Tizhoosh, H., Salama, M.: Opposition based differential evolution. IEEE Trans. Evol. Comput. **12**(1), 64–79 (2008)
9. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local Search. IEEE Trans. Evol. Comput. **12**(1), 107–125 (2008)
10. Pant, M., Ali, M., Abraham, A.: Mixed mutation strategy embedded differential evolution. In: IEEE Congress on Evolutionary Computation, pp. 1240–1246 (2009)
11. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
12. Zhang, J., Sanderson, A.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
13. Das, S., Abraham, A., Chakraborty, U., Konar, A.: Differential evolution using a neighborhood based mutation operator. IEEE Trans. Evol. Comput. **13**(3), 526–553 (2009)
14. Ali, M., Pant, M.: Improving the performance of differential evolution algorithm using cauchy mutation. Soft. Comput. (2010). https://doi.org/10.1007/s00500-010-0655-2
15. Cai, Z., Gong, W., Ling, C., Zhang, H.: A clustering-based differential evolution for global optimization. Appl. Soft Comput. **11**(1), 1363–1379 (2011)
16. Cai, Y., Wang, J., Yin, J.: Learning enhanced differential evolution for numerical optimization. Soft Comput. (2011). https://doi.org/10.1007/s00500-011-0744-x
17. Epitropakis, M.G., Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Enhancing Differential Evolution Utilizing Proximity-Based Mutation Operators. IEEE Trans. Evol. Comput. **15**(1), 99–11 (2011)
18. Kumar, P., Pant, P.: Enhanced mutation strategy for differential evolution. In: Proceeding of IEEE Congress on Evolutionary Computation (CEC-12), pp. 1–6 (2012)
19. Zhu, W., Tang, Y., Fang, J.-A., Zhang, W.: Adaptive population tuning scheme for differential evolution. Inf. Sci. **223**, 164–191 (2013)
20. Sarker, R.A., Elsayed, S.M., Ray, T.: Differential evolution with dynamic parameters selection for optimization problems. IEEE Trans. Evol. Comput. **18**(5), 689–707 (2014)
21. Singh, P., Chaturvedi, P., Kumar, P.: Control parameters and mutation based variants of differential evolution algorithm. J. Comput. Method Sci. Eng. **15**(4), 783–800 (2015)
22. Xiang, W.L., Meng, X.L., An, M.Q., Li, Y.Z., Gao, M.X.: An enhanced differential evolution algorithm based on multiple mutation strategies. Comput. Intell. Neurosci. **2015**, Article ID 285730, 15 pages (2015)
23. Wu, G., Mallipeddi, R., Suganthan, P.N., Wang, R., Chen, H.: Differential evolution with multi-population based ensemble of mutation strategies. Inf. Sci. **329**, 329–345 (2016)
24. Zheng, L.M., Zhang, S.X., Tang, K.S., Zheng, S.Y.: Differential evolution powered by collective information. Inf. Sci. **399**, 13–29 (2017)
25. Meng, Z., Pan, J.-S., Kong, L.: Parameters with adaptive learning mechanism (palm) for the enhancement of differential evolution. Knowl.-Based Syst. **141**, 92–112 (2018)
26. Singh, P., Chaturvedi, P., Kumar, P.: A novel differential evolution approach for constraint optimization. Int. J. Bio-Insp. Comput. **12**(4), 254–265 (2018)
27. Meng, Z., Pan, J.-S., Tseng, K.K.: PaDE: an enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. Knowl.-Based Syst. **168**, 80–99 (2019)

28. Wei, Z., Xie, X., Bao, T., Yu, Y.: A random perturbation modified differential evolution algorithm for unconstrained optimization problems. Soft. Comput. **23**(15), 6307–6321 (2019)
29. Duan, M., Yang, H., Liu, H., Chen, J.: A differential evolution algorithm with dual preferred learning mutation. Appl. Intell. **49**(2), 605–627 (2019)
30. Tian, M., Gao, X.: Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. Inf. Sci. **478**, 422–448 (2019)
31. Wang, S.H., Li, Y.Z., Yang, H.Y.: Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. Appl. Soft Comput. **81** (2019)
32. Pan, J.S., Yang, C., Meng, F.J., Chen, Y.X., Meng, Z.Y.: A parameter adaptive DE algorithm on real-parameter optimization. J. Intell. Fuzzy Syst. **38**(1), 1–12 (2020)
33. Di Carlo, M., Vasile, M., Minisci, E.: Adaptive multipopulation inflationary differential evolution. Soft. Comput. **24**(5), 3861–3891 (2020)
34. Zhong, X., Cheng, P.: An improved differential evolution algorithm based on dual-strategy. Hindawi Math. Prob. Eng. (2020). https://doi.org/10.1155/2020/9767282
35. Plagianakos, V., Tasoulis, D., Vrahatis, M.: A review of major application areas of differential evolution. In: Advances in Differential Evolution, vol. 143, pp. 197–238. Springer, Berlin (2008)
36. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. **33**(1–2), 61–106 (2010)
37. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–13 (2011)
38. Bilal, P.M., Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A.: Differential evolution: a review of more than two decades of research. Eng. Appl. Artif. Intell. **90**, Article 103479 (2020)
39. Kumar, P., Pant, M.: Modified single array selection operation for DE algorithm. In: Proceedings of Fifth International Conference on Soft Computing for Problem Solving, AISC, vol. 437, pp. 795–803 (2016)
40. Kumar, P., Pant, M., Astya, R., Ali, M.: Real life optimization problems solving by IUDE. In: International Conference on Computing, Communication and Automation (ICCCA), pp. 368–372 (2016)
41. Kumar, S., Kumar, P., Sharma, T.K., Pant, M.: Bi-level thresholding using PSO, Artificial Bee Colony and MRLDE embedded with Otsu method. Memetic Comput. **5**(4), 323–334 (2013)
42. Kumar, P., Pant, M., Singh, V.P.: Modified random localization based de for static economic power dispatch with generator constraints. Int. J. Bio-Insp. Comput. **6**(4), 250–261 (2014)
43. Kumar, P., Singh, D., Kumar, S.: MRLDE for solving engineering optimization problems. In: International Conference on Computing, Communication & Automation, pp. 760–764. https://doi.org/10.1109/CCAA.2015.7148512 (2015)
44. Kumar, P., Pant, M.: Recognition of noise source in multi sounds field by modified random localized based DE algorithm. Int. J. Syst. Assur. Eng. Manag. **9**(1), 245–261 (2016). https://doi.org/10.1007/s13198-016-0544-x
45. Kumar, P., Sharma, A.: MRL-Jaya: a fusion of MRLDE and Jaya Algorithm. Palestine J. Math. **11**, 65–74 (2022)
46. Dor, A.E., Clerc, M., Siarry, P.: Hybridization of differential evolution and particle swarm optimization in a new algorithm: DEPSO-2S. In: Proceeding of SIDE 2012 and EC 2012, LNCS 7269, , pp. 57–65. Springer, Berlin/Heidelberg (2012)
47. Garg, V., Deep, K.: Performance of Laplacian Biogeography-Based Optimization Algorithm on CEC 2014 continuous optimization benchmarks and camera calibration problem. Swarm Evol. Comput. **27**, 132–144 (2016)
48. Garg, V., Deep, K.: Constrained Laplacian biogeography-based optimization algorithm. Int. J. Syst. Assur. Eng. Manag. **8**(2), 867–885 (2017)
49. Garg, V., Deep, K.: Efficient mutation strategies embedded in Laplacian-biogeography-based optimization algorithm for unconstrained function minimization. Int. J. Appl. Evol. Comput. (IJAEC). **7**(2), 12–44 (2016)