Evolutionary Intelligence (2022) 15:1571–1585
https://doi.org/10.1007/s12065-021-00568-z

RESEARCH PAPER

An adaptive mutation strategy for differential evolution algorithm based on particle swarm optimization

Abhishek Dixit[1] · **Ashish Mani**[2] · **Rohit Bansal**[3]

Received: 11 April 2020 / Revised: 22 December 2020 / Accepted: 6 January 2021 / Published online: 16 February 2021

Abstract

Differential evolution (DE) algorithm is a very effective algorithm used for solving wide range of optimization problems. However, the performance of DE is dependent on the control parameters and to choose the right parameter value and tuning of these parameters is a challenging task. Therefore, a novel variant of differential evolution algorithm based on particle swarm optimization (DEPSO) is proposed to improve the overall performance of Differential evolution algorithm. In our proposed approach, we are using DE mutation strategy during the initial phase of evolution and therefore enlarge its search space possibly to the extent that helps in finding more encouraging results and thus avoid premature convergence. During the subsequent phase of evolution process, this value of sigmoid function reduces with the increase of number of iterations. In this scenario, there is a greater probability of operating PSO mutation strategy and thus this sigmoid function helps in improving the precision and convergence speed. The Performance of our proposed algorithm is tested with 10 benchmark test functions on 50 and 25 dimensions set, also tested with 11 test functions on 30- and 100-dimension test functions. We have also tested our proposed algorithm with 8 test functions on high dimension set as 500- and 1000-dimensions. The performance comparison shows that our proposed variant is giving significant improvement in convergence speed and thus avoiding premature convergence. Average performance of DEPSO is better than classical DE, PSO and other algorithms in comparison.

Keywords Differential evolution (DE) · Particle swarm optimization (PSO) · Global optimization · Evolutionary algorithm (EA)

1 Introduction

Differential Evolution algorithm, proposed by Storn and Price [1, 2] is well known for its exploration capability using three control parameters as mutation, crossover and population size (NP). In the recent years, there are various research work done that showcase the capability of DE in solving diverse optimization problems such as image segmentation [3, 4], pattern recognition [5, 6], and functions optimization [7, 8]. In all these papers, the range of recommended values of control parameters are given. The values of control parameters determine the efficiency of Differential evolution algorithm to find the optimal solution for a given problem. But to choose the right parameter values by trial and error is often time-consuming approach. The tuning of these control parameters is simple but specific to the problems and these tuned parameters varies during the evolution process [9].

Several variants of Differential evolution algorithm are proposed in literature suggesting the influence of these

✉ Ashish Mani
amani@gmail.com

✉ Rohit Bansal
rohitbansaliitr@gmail.com

Abhishek Dixit
abhishekdixitg@gmail.com

[1] Department of Computer Science, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India

[2] Department of EEE, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India

[3] Department of Management Studies, Rajiv Gandhi Institute of Petroleum Technology, Jais, Amethi, Uttar Pradesh, India

parameters. Liu et al. [10] proposed Fuzzy adaptive differential evolution algorithm (fADE). In this algorithm, new adaptive mutation and crossover parameters based on Fuzzy logic have been introduced. Another novel version of Differential evolution algorithm JADE was proposed by Zhang et al. [11]. This algorithm proposes a new mutation strategy by adaptive control parameter with optional external archive. Wang et al. [12] roposed another variant of Differential evolution algorithm by designing a new adaptive mutation strategy and named it as IMSaDE. This algorithm improves "DE/rand/2" mutation strategy. Alswaitti et al. [13] proposed a novel variance-based crossover strategy for improving the convergence rate. Ramadas et al. [14] proposed a new mutation strategy and named it as FSDE. FSDE added two parameters for controlling mutation viz., variable and constant parameter. However, there still exist the problem of finding the right control parameter values and in most of the cases, it still depends on past experiences.

Particle swarm optimization algorithm is also considered to be the important and effective evolutionary algorithm. This algorithm was proposed by Kennedy and Eberhart [15] and depends on individual best, pbest and global best solution, gbest for guiding the search for global optimal solutions. This algorithm is known for its faster convergence speed, less initialization parameters and ease of implementation in complex optimization problems [16–18]. However, the main demerit of PSO is falling into local optima during early evolution stage in comparison to other evolutionary algorithms and suffering from lack of population diversity that often leads to premature convergence during the later evolution stage [19, 20].

To overcome the demerits of both DE and PSO algorithm, efforts have been made to maintain a better balance between exploration and exploitation capability of both the algorithms by hybridizing them, which is a growing area of research and its objective is to mitigate the weakness of individual algorithms. To balance exploration and exploitation capabilities of DE and PSO, a hybrid variant named HCPSODE was proposed by Lin et al. [21]. In this approach, a chaotic map with greater Lyapunov exponent and a new nonlinear approach for reducing inertia weight is introduced. Wang et al. [22] proposed another DE and PSO hybridization strategy that evade the suboptimal solutions proposed in previous approaches. In this approach, a collective mutation strategy is developed for maintaining the diversity and convergence speed. Wang et al. [23] introduced another hybridization approach to maintain the DE global exploration and local exploitation. In this approach, a DE mutation and crossover strategy is improved by adding PSO mutation with no additional resource required. Pérez-González et al. [24] proposed a greenhouse model based on PSO and DE. Ahmadianfar et al. [25] proposed another hybrid variant of DE and PSO with multi strategy that helps in exploring the local and global search capabilities. Dash et al. [26] proposed a hybrid DEPSO for designing optimal FIR filter. In this approach, the DE control parameters are combined with fine-tuned parameters of PSO. Even though these algorithms are based on ensemble approaches of DE and PSO that improve the exploration and exploitation capabilities of both the algorithms, but they require more resources to compute, fine-tuning of complex parameters and may lead to premature convergence.

In this study, an effort has been made to solve the above discussed problems by maintaining a better balance between exploration and exploitation capabilities of both the algorithm by proposing an alternative hybridized variant of Differential evolution and PSO. In our proposed approach, we are using $DE/rand/2$ mutation strategy and therefore enlarge its search space possibly to the extent that helps in finding more encouraging results and thus avoid premature convergence. During the subsequent phase of evolution process, this value of sigmoid function reduces with the increase of number of iterations. In this scenario, there is a greater probability of operating PSO mutation strategy and thus this sigmoid function helps in improving the precision and convergence speed. Although DE strategy can also perform mutation and it help in convergence speed and precision, but this sigmoid function will ensure high precision and convergence speed of our proposed algorithm. In addition to this, parameters are archived and compared in each iteration during the evaluation process so that parameters are tracked and updated. This mutual cooperation between the self-adaptive mutation approach and parameter archival strategy enhances the performance of our proposed hybrid approach.

This proposed approach is different from past efforts as in previous approaches the original $DE/rand/1$ mutation strategy is used with complex rules added on several features of hybrid structure of algorithm that results in complex hybrid schemes. Whereas in our approach $DE/rand/2$ strategy is used with sigmoid function to improve its convergence speed and accuracy. This makes our algorithm more simple, easy to implement and easily extensible. Another difference from previous work is in choosing the mutation strategy. In previous approaches, there has a strong randomness in an evolution process, which results in lack of requisite guidance. On the other hand, the proposed DEPSO approach has better guidance as $DE/rand/2$ mutation strategy is used in early stage to expand and explore in all regions and PSO is used in later stage to search the promising regions found during the initial stages and thus improving the local exploitation capability.

The proposed approach is tested on 50 and 25-dimensional test functions and compared with conventional DE, PSO and one recently proposed variant of Differential evolution algorithm Forced Strategy Differential Evolution

used for data clustering (FSDE) [14] algorithm We have also compared our approach with another hybrid variant of differential evolution and PSO (HDEPSO) [26] Further we have also compared our algorithm with 3 well known variants Self-Adapting Control Parameters in Differential Evolution jDE [24], self-adaptive differential evolution algorithm (SaDE) [27] and JADE: Adaptive Differential Evolution with Optional External Archive [11]. Similar to our work, in all these variants a new mutation strategy is developed based on tuning the control parameters. We have chosen 15 benchmark test functions to evaluate the performance of our proposed approach. We have also performed Friedman's test to validate the statistical performance of our proposed algorithm.

This paper is organized as follows: DE and PSO are briefly described in Sect. 2 and 3 respectively. Our proposed method is detailed out in Sect. 4. Section 5 represent the experimental setting. Section 6 shows the results and discussion section. Finally, Sect. 7 concludes the outcome of this paper.

2 Related work

DE is a well-known evolutionary algorithm and has attracted various researchers in different domain due to its simplicity and efficiency. The values of control parameters determine the efficiency of differential evolution algorithm to find the optimal solution for a given problem. There are various modifications which have been done in classical DE to improve its efficiency. Two different schemes of population initialization approaches have been proposed by Ali et al. [28]. A cluster-based population initialization approach has been proposed by Poikolainen et al. [29]. In this approach, three successive steps of pre-processing were introduced in order to improve the efficiency of DE. Another population initialization approach has been proposed by Sun et al. [30], that was based on novel fluctuant population approach in which population size is adjusted during each run.

Researchers have also worked on mutation strategy of differential evolution algorithm in order to improve its efficiency. Wang et al. [12] proposed another variant of Differential evolution algorithm by adaptive a new mutation strategy and named it as IMSaDE. This algorithm improves "DE/rand/2" mutation strategy. Brest et al. [31] proposed a Self-Adapting Control Parameters in Differential Evolution jDE as a new variant of DE. Qin et al. [27] proposed a self-adaptive differential evolution algorithm (SaDE). In this approach four different mutation strategies of DE ($DE/rand/1, DE/rand-to-best/2, DE/rand/2$ and$DE/current-to-rand/1$) are used a candidate pool strategy used for each individual as per the success rate in generating better child generation. Zhang et al. [11] proposed JADE: Adaptive Differential Evolution with Optional External Archive as a novel mutation strategy asldquoDE/current − to − *p*bestrdquo. In this approach the parameters are adjusted by archival and adaptive strategy. Shao et al. [32] proposed an improved utilization strategy of population information. In this approach best and worst vectors generated during evolution phase are calculated. These vectors are utilized during crossover and mutation strategy to replace the parent vector in order to improve the performance of DE. Annepu et al. [33] proposed a self-adaptive approach to improve the convergence speed based on mutation factor and cross-over probability. Another approach proposed by Zhang et al. [34] in which a novel mutation approach as inter-vehicle strategy and novel single point crossover and route sensitive selection approach is proposed to improve the efficiency.

Hybridization of evolutionary algorithms are superior to standalone algorithm as they have the capability of overcoming the weaknesses of individual algorithms without losing their advantages [35]. Sun et al. [36] hybridizes DE with estimation of distribution algorithm (EDA) in which global parameters values of EDA and DE are utilized to give optimized solutions. Wang et al. [37] proposed a hybrid approach-based DE with Nelder–Mead (NM) simplex search. In this approach NM local search ability is combined with evolutionary search of DE to get the robust and effective results. Guo et al. [38] proposed an enhanced self-adaptive differential evolution (ESADE). In this approach DE is hybridized with Simulated annealing during the selection operation. This hybridization improves the global search ability of DE. The experimental results show the improved performance of ESADE over other algorithms in comparison. Keshk et al. [39] proposed another variant of DE combined with Hidden markov model. In this approach, for the given population HMM is used to compress the information and utilized the model for adjusting DE parameters. The comparison results show that DE-HMM hybridization algorithm is giving better performance as compared with other algorithms in comparison.

There are many hybrid variants which combines DE with PSO. Tian et al. [40] proposed a hybrid variant of DE with PSO that utilizes mutation and crossover operators of DE and present a novel mutation strategy by replacing PSO's velocity and position parameter. Also, a random neighbor individual is selected for crossover. In this approach, the operator selection is specified for the whole evolutionary cycle. Wang et al. [22] proposed hybrid variant of DE and PSO that ignores the suboptimal solutions proposed in previous approaches. In this approach, sub-population are generated from initial population by both DE and PSO and population with best particles are constituted as the initial population to perform DE operation. As for each iteration, 2 sub-population are generated that increases the space

complexity of this algorithm. Dash et al. [26] presented an approach that combines DE with PSO (HDEPSO) in which the best features of both the algorithms are utilized. However, this approach requires more resources to compute and specific number of parameters are fine-tuned which may lead of increase in average execution time of algorithm.

3 Differential evolution algorithm (DE)

Storn and Price [1, 2] first developed Differential evolution algorithm. This algorithm is utilized for solving various continuous and discrete optimization problems associated in different domains. Being a good optimization technique, this work is also inspired by their basic approach. Further, in this section a detailed description about basic terminologies of DE is presented.

3.1 Initiation

Let us suppose that a population NP with X_{id}^{I} as the candidate solution in ith index of generation I. where $i = 1, 2, \ldots, NP$. Differential evolution algorithm mainly depends on mutation, crossover and selection operators. We will discuss on these three operators in subsequent sections.

3.2 Mutation

This is a unique operator which makes DE diverse in comparison to other EAs. This operator is used to generate the trial vector V_{id}^{I} which is used to generate offspring. So, for generating V_{id}^{I},mutation is applied w.r.t each individual vector. In DE this strategy is commonly represented as $DE/x/y/z$ where x and y are the elementary and variance vector respectively. z is the crossover arrangement. The DE mutation approaches are implemented as below.

$$
\begin{aligned}
&DE/rand/1: \\
&\quad V_{id}^{I+1} = X_{r1d}^{I} + F.\left(X_{r2d}^{I} - X_{r3d}^{I}\right) \\
&DE/best/1: \\
&\quad V_{id}^{I+1} = X_{best}^{I} + F.\left(X_{r1d}^{I} - X_{r2d}^{I}\right) \\
&DE/currenttobest/2: \\
&\quad V_{id}^{I+1} = X_{r1d}^{I} + F.\left(X_{best}^{I} - X_{r1d}^{I}\right) + F.\left(X_{r1d}^{I} - X_{r2d}^{I}\right) \\
&DE/best/2: \\
&\quad V_{id}^{I+1} = X_{best}^{I} + F.\left(X_{r1d}^{I} - X_{r2d}^{I}\right) + F.\left(X_{r3d}^{I} - X_{r4d}^{I}\right) \\
&DE/rand/2: \\
&\quad V_{id}^{I+1} = X_{r1d}^{I} + F.\left(X_{r2d}^{I} - X_{r3d}^{I}\right) + F.\left(X_{r4d}^{I} - X_{r5d}^{I}\right)
\end{aligned}
\tag{1}
$$

where F is the positive constant known as scaling factor in the range [0, 2], mutant vector is represented as V_{id}^{I}, mutually exclusive randomly selected integers are represented as $X_{r1d}^{I}, X_{r2d}^{I}, X_{r3d}^{I}, X_{r4d}^{I}, X_{r5d}^{I}$. The random values r_1, r_2, r_3, r_4, r_5 are chooses such that $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$.The best individual selected values is represented as X_{best}^{I} for the generation I.

3.3 Crossover

Post mutation, mutation vector V_{id}^{I} and parent vectors X_{id}^{I} are engaged to produce offspring also known as trail vector S_{id}^{I}. This process is called as crossover. Crossover is implemented as below.

$$
S_{id}^{I} = \begin{cases} V_{id}^{I} \; if \; rand \leq Cr \\ X_{id}^{I} \; otherwise \end{cases} \tag{2}
$$

where Cr is the crossover rate. The value of Cr is in range [0, 1]. $rand$ is a random number and value is in range [0, 1]. This random number guarantee that the preliminary vector S_{id}^{I} should at the most get one member from mutation and this helps in avoiding the stagnation during evolutionary process. Although during the evolutionary process, few trail vector's element due to the impact of mutation process may get deviated from the possible solutions space. So, to reset the infeasible solution.

$$
S_{id}^{I} = \begin{cases} X_{min} + rand(0,1) * \left(X_{max} - X_{min}\right) \; if \; S_{id}^{I} \notin \left[X_{max}, X_{min}\right] \\ X_{id}^{I} \; otherwise \end{cases} \tag{3}
$$

3.4 Selection

Selection is the process of determining the parent and offspring for the next generation. This process is also applied to find out which individual will take part in mutation process. In this process, DE employ greedy selection approach for evaluation of trail vectors. This process can be implemented by using below equation.

$$
X_{id}^{I+1} = \begin{cases} S_{id}^{I} \; if \; fun\left(S_{id}^{I}\right) \leq fun\left(X_{id}^{I}\right) \\ X_{id}^{I} \; otherwise \end{cases} \tag{4}
$$

From the above equation, if the fitness value of exploratory vector S_{id}^{I} is lesser than or equal toX_{id}^{I}, then S_{id}^{I} will replace X_{id}^{I}. The above steps is repeated until a stopping criteria is met.

4 Particle swarm optimization (PSO)

Kennedy and Eberhart [15] first proposed PSO in 1995. PSO is stochastic search process based on velocity-position scheme. In PSO, each individual particle can be represented as a point in dimension D of search space. From the below equation, ith particle position is denoted as.

$$X_i = X_{i1}, X_{i2} \ldots, X_{iD} \tag{5}$$

And velocity is signified as:

$$Y_i = Y_{i1}, Y_{i2}, \ldots Y_{iD} \tag{6}$$

In an evolution process each particle calculate its individual best position also known as *pbest* and global best position known as *gbest* and accordingly update its own position and velocity as per the equation below.

$$Y_{id}^{t+1} = Y_{id}^{t} + c_1 * r_1 * \left(pbest - X_{id}^{t}\right) + c_2 * r_2 * \left(gbest - X_{id}^{t}\right) \tag{7}$$

$$X_{id}^{t+1} = X_{id}^{t} + Y_{id}^{t} \tag{8}$$

where $c_1 and c_2$, are the learning coefficients. $r_1 and r_2$, are in the range [0, 1] as a regular random number. t and $t + 1$ are the iterations in search process. $d \in D$ is the dimension d in search space. The velocity Y_{id}^{t} is within the velocity limits as $Y_{id}^{t} \in \left[-Y_{max}, Y_{max}\right]$

5 The proposed algorithms

We have introduced a novel hybridization approach by utilizing the merits of both Differential evolution (DE) and Particle swarm optimization (PSO) algorithm. Thus, accuracy and efficiency in optimization process can be improved by hybridizing local search ability of PSO and global search ability of DE. By combining the benefits of both the algorithm, a new mutations strategy is proposed, and the technique is named as DEPSO. We have used binary particle swarm optimization (BPSO) where the velocity with the value of $X_{id}, pbest$ and *gbest* is updated within range of [0, 1].

In our approach the main operator is DE, and PSO is used to improve the search capability. This helps to speed up the search and increase the accuracy of our algorithm. This method uses the DE mutation strategy in early stage to expand and explore in all regions and PSO is used in later stage to search the regions found during the initial stage and thus avoid the local optima and improve the local exploitation capability. As an outcome, this procedure has programmed stability among global and local searching.

In our approach, we have used a unique selection probability function for DE and PSO mutation strategies.

$$sig = \frac{1}{1 + 2e^{-(I_{max}/I)^{\sigma}}} \tag{9}$$

In this equation, I_{max} is the maximum number of generation and I is the current generation set for the evolution process.σ is the positive constant value set during the evaluation process. The detail of the sigmoid function and values are set experimentally in Sect. 6.1.1. The value of sigmoid function is large during the initial phase of evolution and probability of rand () is also large. In our proposed approach, we are using DE mutation strategy and therefore enlarge its search space possibly to the extent that to in finding more encouraging results and thus to evade the early convergence. During the subsequent phase of evolution process, as the number of iterations increases, this value of sigmoid function reduces. In this scenario, there is a greater probability of operating PSO mutation strategy and thus this sigmoid function helps in improving the precision and convergence speed. Although DE strategy can also perform mutation and it help in convergence speed and accuracy, but this sigmoid function will ensure high precision and convergence speed of our proposed algorithm.

The proposed method initially starts with choosing and initializing the parameters of both DE and PSO. Randomly generated control parameters like F, Cr and inertia weight and sigmoid function is calculated. As a next step, perform mutation operation based on the sigmoid function. $DE/rand/2$ mutation strategy is used for calculation. PSO mutation strategy is also used if rand value is greater than or equal to rand function. In the next step crossover is applied and corresponding trial vector is chosen based on crossover rate. Finally, selection is applied, and global best and individual best is selected, and corresponding control parameters are updated. The step by step explanation of our proposed approach is explained below. Selection of mutation strategy based on sigmoid function is done to speed up the convergence and maintain the population diversity. Pseudo code for our proposed algorithm is represented below.

Step 1: Set the NP, Iteration, generation. Randomly generate population

Step 2: Randomly generate control params like F, Cr, inertia weights

Step 3: while number of iterations reaches max

For i = 1:NP

%Compute sigmoid function using Eq. (9)

$\boldsymbol{sig} = \frac{1}{1+2e^{-(I_{max}/I)^{\sigma}}}$

% Mutation

Choose the random number using rand [0, 1]

If rand ≤ sig

% $\boldsymbol{DE/rand/2}$ mutation strategy

$\boldsymbol{V_{id}^{I+1} = X_{r1d}^{I} + F.\left(X_{r2d}^{I} - X_{r3d}^{I}\right) + F.\left(X_{r4d}^{I} - X_{r5d}^{I}\right)}$

Else

% PSO mutation strategy

$\boldsymbol{Y_{id}^{t+1} = Y_{id}^{t} + c_1 * r_1 * \left(pbest - X_{id}^{t}\right) + c_2 * r_2 * \left(gbest - X_{id}^{t}\right)}$

End If

Step 5: % Crossover

For i1 = 1: Ds

$\boldsymbol{S_{id}^{I} = \begin{cases} V_{id}^{I} & if\ rand \le Cr \\ X_{id}^{I} & otherwise \end{cases}}$

End For

Step 6: % Selection

$\boldsymbol{X_{id}^{I+1} = \begin{cases} S_{id}^{I} & if\ fun\left(S_{id}^{I}\right) \le fun\left(X_{id}^{I}\right) \\ X_{id}^{I} & otherwise \end{cases}}$

End For

End While

Step 7: Repeat steps 3 to 6 until any stopping conditions is achieved

6 Experimental results

6.1 Experimental settings

In order to evaluate the performance of our proposed approach, 10 standard benchmark functions listed in Appendix are utilized in our experimentation. Further 11 standard benchmark function are used to compare our proposed algorithm with 4 well known variants of Differential evolution algorithm. The experiments are instigated using MATLAB with 2.11 GHz, Intel ® Core™ i7-8650U and 16 GB of RAM system configuration. Various parameters which are used to produce results are shown in Table 1.

We have performed the experiment on our proposed technique and compared the results with classical DE [1], PSO [15], ABC [41] and FSDE [14], HDEPSO [26] algorithm. To perform our experiment, we have used 25 and 50 as dimensions and population size as 100. Since both the algorithms are stochastic in nature so we have kept maximum number of evaluations as 5,000,000 and 1000 as maximum number of iterations. The FSDE is using VTR function with value as $1.e^{-015}$. The scaling factor and crossover rate chosen as 0.6 and 0.8 respectively. The velocity limits are $2, -2$ and inertia weight are 0.99. In the evolution stage setting up the right parameters is very important as performance is dependent on these control parameters, however choosing right parameters is a difficult process. In our approach values are adjusted in real time by monitoring the evolutionary process of each individual. During the process of evolution if any individual stagnates then the values need to be readjusted. In literature there is no guideline to set the values of control parameters, instead the random selection of values by trial and error is considered as the better option [42].

6.1.1 Selection of mutation strategy

In the Eq. 9 the selection probability for mutation strategy and value of constant values used is studied experimentally. The positive value of σ plays an important role in calculating the sigmoid function. The Fig. 1 shows the probability of choosing the DE and PSO strategies. In the figure, to better understand the evolution process and impact of σ during the evolution process, the entire evolution process is divided into initial and final stage. We can also observe that with the increase of σ the chances of selection of DE mutation strategy increases and probability of selection of PSO mutation strategy becomes smaller. From the figure we can observe that during the initial stage of evolution the chances of choosing DE mutation strategy increase with $\sigma \geq 1.3$. This ensure the population diversity and helps our proposed algorithm to search the maximum region. As DEPSO should also ensure in improving the local optima and thus convergence speed or accuracy, therefore the participation of PSO in mutation operation should be increased. From the figure we can observe that for final or later stage of evolution the value of $\sigma \leq 1.5$. Therefore, for our experiment to balance the population diversity and convergence speed we have chosen the value of σ as 1.4

6.2 Result analysis

The proposed algorithm is evaluated on 10 test benchmark function and then the results are compared with classical DE, PSO, ABC, recently proposed variant of Differential evolution -FSDE and hybrid HDEPSO. The results are tabulated in Tables 2, 3, 4 and 5. To validate the results, we have also conducted Friedman's statistical test on our proposed algorithm DEPSO. The rank and the Friedman's test results are shown in Table 6.

Table 1 Different experimental parameters with their setting values

Parameter	Setting value
No of Iteration	1000
Pop size (NP)	100
Inertia weight damping ratio (w)	.99
C_1	2.05
C_2	2.05
Velocity limits	$2, -2$
Scale Factor (F)	0.6
Crossover rate (Cr)	0.8
Value to reach (VTR)	$1.e^{-015}$
Dimension (D)	25, 50

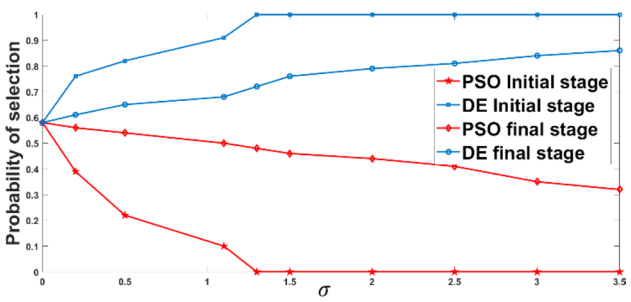

Fig. 1 Probability of selection of mutation strategy

We have performed our experiment on 25-dimensional test functions and results are shown in Tables 2 and 4. When we compare the best values of all the algorithms, our proposed algorithm is showing better results on 7 test functions whereas FSDE is showing better results on only 3 test functions. We have captured the mean and standard deviation results in Table 4. We have also performed Wilcoxon's rank sum test on the obtained results and indicated performance comparison is added in the Table 4. We have also performed Friedman's test for confirming the validity of our results. The results obtained from our experiment shows that when compared with DE, DEPSO is performing better on 9, worst on 1 function. DE-PSO when compared with PSO, our proposed algorithm is showing better results on 7, worst on 2 and similar results on 1 test functions. DEPSO when compared with ABC, our algorithm is giving better results on 8 and worst result on 2 test functions. When compared with FSDE, our proposed algorithm is showing better results on 7 test functions, worst on 2 and similar on 1 test functions. Comparing HDEPSO with our algorithm shows that our approach is giving better results on all the test functions. From the results we can see that our proposed algorithm is performing better, and this may be due to hybridization of exploration and exploitation of DE and PSO in an effective manner. In our algorithm there is an equilibrium of global exploration and local exploitation during early and later stages of evolution, therefore showing higher convergence speed and robustness.

Similarly, All the algorithms are tested on 50 dimensional functions and the results of best values are shown in Table 3. As shown in Table 3, our proposed algorithm is showing better results on 7 test functions and PSO is showing better results on 2 test functions whereas FSDE is showing better results only on 1 test functions. The mean and standard deviations are shown in Table 5. We have applied Wilcoxon's rank sum test on the obtained results and performance of our proposed algorithm is compared and indicated in $+$, $-$ and $\approx$ as the DE-PSO showing worst, better and similar performance results. We have also applied Friedman's non-parametric test and results are shown in Table 6. This test is performed to showcase the validity of our results. From the results we can see that DEPSO when compared with DE is showing better results on 8 functions, showing worst results on 2 functions. Similarly, DEPSO when compared with PSO is showing better results on 7 functions, worst results on 1 and similar performance on 2 results. DEPSO when compared with ABC, our algorithm is showing better results on 11 results, worst results on 3 and similar results on 1 function. Now when compared with FSDE, our proposed algorithm is

Table 2 Performance values achieved by DE, PSO, ABC, FSDE, HDEPSO and DEPSO on 10 test functions with 25D

F	DE	PSO	ABC	FSDE	HDEPSO	DEPSO
f1 (Sphere)	7.60E−111	3.81E−283	8.73E−25	3.28E−140	3.73E−163	5.42E−307
f2(Rosenbrock)	7.10E−30	4.93E−32	4.52E−13	3.58E−32	4.37E−11	4.93E−32
f3(Ackley)	6.22E−15	7.73E−14	1.59E−11	4.44E−15	7.99E−15	9.77E−18
f4(Griewank)	8.85E−02	7.40E−03	1.07E−01	1.48E−02	3.82E−04	7.40E−05
f5(Quartik)	1.17E−189	5.93E−323	9.56E−30	3.25E−40	7.72E−05	4.94e−324
f6(Schwefel_2_21)	1.02E−12	4.62E−03	1.27E−05	9.91E−06	2.39E−44	1.87E−114
f7(Michalewicz)	−9.96E+00	−7.40E+00	−4.12E+00	−9.69E+00	−1.9962	−9.85E+00
f8(Rastrigin)	1.99E+00	7.76E+01	1.33E+01	2.15E−02	2.1E+01	9.95E+00
f9(Zakharov)	7.57E−05	8.57E−03	8.28E−03	2.07E−02	5.4E−04	6.12E−06
f10(Powell)	7.78E−08	8.26E−08	1.80E−05	1.49E+01	1.39E−05	4.32E−14

Table 3 Performance values achieved by DE, PSO, ABC, FSDE, HDEPSO and DEPSO on 10 test functions with 50D

F	DE	PSO	ABC	FSDE	HDEPSO	DEPSO
f1 (Sphere)	2.15E−02	1.13E−51	1.01E+01	1.87E−39	1.77E−52	1.82E−55
f2(Rosenbrock)	4.52E−13	4.93E−12	2.33E−16	4.44E−20	2.77E−13	6.72E−19
f3(Ackley)	2.98E+00	2.75E−14	2.53E−01	9.68E−14	7.99E−15	2.66E−15
f4(Griewank)	6.34E−02	2.22E−15	4.02E−01	1.03E+00	5.72E−10	8.55E−16
f5(Quartik)	1.00E−134	1.83e−322	1.05E−70	4.59E−03	3.56E−04	0.00E+00
f6(Schwefel_2_21)	2.21E+00	4.33E−03	6.02E+00	8.33E+00	5.04E−01	9.23E−03
f7(Michalewicz)	−3.15E+01	−2.16E+01	−7.73E+00	−27.1994	−1.9679	−2.16E+01
f8(Rastrigin)	5.97E+02	6.57E+01	3.15E+02	5.54E+01	7.45E+03	7.36E+01
f9(Zakharov)	4.89E+02	2.26E−02	1.05E+03	2.35E+00	4.56E−02	2.13E−03
f10(Powell)	7.71E−10	3.47E−09	2.16E−05	5.40E+01	2.27E−05	2.13E−11

Table 4 Mean and SD for DE, PSO, ABC, FSDE, HDEPSO and DEPSO on 10 functions on 25 D

Function	DE Mean (std dev)	PSO Mean (std dev)	ABC Mean (Std Dev)	FSDE Mean (std dev)	HDEPSO Mean (std dev)	DEPSO Mean (std dev)
f1 (Sphere)	3.14E−06 (4.54E+03)	2.57E−134 (8.12E−134)	3.42E−22 (1.08E−21)	4.95E−06 (4.09E+03)	4.70E−161 (0.00E+00)	2.99E−148 (9.46E−148)
f2 (Rosenbrock)	3.76E+07 (2.02E+07)	0.00E+00 (0.00E+00)	8.35E−13 (2.64E−12)	1.32E+01 (8.84E+06)	9.04E−11 (1.51E−10)	1.20E−14 (3.79E−14)
f3 (Ackley)	1.99E+01 (2.66E−01)	1.93E−14 (6.12E−14)	7.11E−12 (2.25E−11)	4.70E−04 (4.38E+00)	7.64E−15 (1.12E−15)	2.51E−15 (7.94E−15)
f4 (Griewank)	8.62E−03 (2.73E−02)	7.42E−03 (2.35E−02)	1.40E−02 (4.44E−02)	1.02E+01 (3.35E+00)	1.02E+01 (3.35E+00)	4.96E−03 (1.57E−02)
f5 (Quartik)	1.06E+01 (9.47E+00)	0.00E+00 (0.00E+00)	6.23E−27 (1.97E−26)	8.98E−03 (3.29E+00)	3.75E−04 (1.83E−04)	0.00E+00 (0.00E+00)
f6 (Schwefel_2_21)	1.93E+03 (3.48E+03)	5.08E+03 (7.46E+02)	7.68E−06 (2.43E−05)	3.00E+03 (2.26E+03)	3.00E+03 (2.26E+03)	1.03E+04 (8.34E+00)
f7 (Michalewicz)	−8.94E−01 (1.26E+00)	3.72E−01 (1.18E+00)	9.34E−02 (2.95E−01)	−8.74E−01 (1.24E+00)	−8.74E−01 (1.24E+00)	8.57E−02 (2.71E−01)
f8 (Rastrigin)	2.58E+02 (2.79E+01)	2.61E+00 (8.26E+00)	7.39E−01 (2.34E+00)	2.43E+01 (5.05E+01)	2.43E+01 (5.05E+01)	2.38E+00 (7.54E+00)
f9 (Zakharov)	3.88E−09 (1.05E−02)	2.33E−04 (7.63E−04)	1.43E−02 (4.51E−02)	7.10E−09 (1.27E−02)	7.10E−09 (1.27E−02)	4.71E−19 (1.49E−18)
f10 (Powell)	2.06E+03 (1.14E+03)	6.21E−08 (1.96E−07)	6.70E−04 (2.12E−03)	2.97E−04 (5.67E+02)	1.37E−05 (2.11E−05)	6.24E−06 (1.97E−05)
−	9	7	7	7	8	
≈	0	1	0	1	2	
+	1	2	3	2	0	

showing better results on 7 results, worst results on 1 and similar results on 2 function. So, we are seeing similar results as with 25D test functions, our proposed algorithm is showing better results as compared to other algorithms on both 50 and 25-dimensional test functions. Compared with HDEPSO, our proposed algorithm is showing better results on all the test functions. FSDE is showing slightly better results on 3 benchmark function on 25D and on 50D it is showing only on 1 test function. This is because of their weighted mutation strategy but it is not performing on the rest as the mutation strategy converges slowly. However, our proposed solution is giving better results on 7 test functions on 25D and on 9 on 50D. This is also because of the sigmoid function that we have added to update the position value also help in increasing the velocity and in turn the search performance. Another reason is the updated hybrid mutation strategy of DE and PSO that helps algorithm to achieve population diversity as well as improve the convergence speed.

Figure 2 and 3 shows the best value comparison graphs plotted for DE, PSO, ABC, FSDE, HDEPSO and DEPSO on different benchmark functions having dimension 25 and 50. Y-axis represents the best cost and X-axis represents the number of iterations. From graphs as well, it can be concluded that our algorithms curve is efficient on all the benchmark function. Our proposed algorithm is performing well and superior to other compared algorithm on all the benchmarking problems in terms of convergence rate. From the figures it can also be concluded that the DEPSO has good competency of escaping from local optimal. Plotted curves also show that the function values of the DEPSO rapidly decreases as the number of iterations increases and the mutation operator accelerate the search and help the algorithm to obtain optimal or near optimal solution.

We have also performed statistical non-parametric Friedman test on our proposed approach DEPSO to validate our results. From the results it is evident that our proposed algorithm is showing statistically better results as compared with other algorithms. Table 6 shows the results obtained from Friedman test.

Differential evolution algorithm maintains the constant parameters for the evolution however for each individual dynamic parameter can achieve optimized performance for each stage. Every optimization problem has different attributes and so it is difficult to achieve best parameter values therefore a strategy based on selection of dynamic selection of parameters will improve the performance of differential evolution algorithm. In our proposed approach the values of each individual can be adjusted based on the real time evolutionary status. Therefore, individual evaluation is stagnant

Table 5 Mean and SD for DE, PSO, ABC, FSDE, HDEPSO and DEPSO on 10 functions on 50 D

Function	DE Mean (Std Dev)	PSO Mean (Std Dev)	ABC Mean (Std Dev)	FSDE Mean (Std Dev)	HDEPSO Mean (Std Dev)	DEPSO Mean (Std Dev)
f1 (Sphere)	1.76E−01 (5.56E−01)	1.60E−50 (5.06E−50)	4.65E−03 (1.47E−02)	9.08E+00 (4.26E−01)	2.01E−53 (1.92E−53)	5.66E−56 (1.79E−55)
f2(Rosenbrock)	4.69E−14 (1.48E−13)	0.00E+00 (0.00E+00)	7.81E−12 (2.47E−11)	0.00E+00 (0.00E+00)	2.20E−11 (6.59E−11)	2.55E−23 (8.07E−23)
f3(Ackley)	1.60E−01 (5.06E−01)	6.59E−12 (2.08E−11)	1.02E−02 (3.24E−02)	9.54E+05 (7.01E+05)	7.99E−15 (1.66E−14)	1.16E−15 (3.67E−15)
f4(Griewank)	5.69E−03 (1.80E−02)	6.95E−03 (2.20E−02)	1.98E−02 (6.25E−02)	2.84E+01 (2.83E+00)	5.72E−02 (2.74E−01)	6.78E−03 (1.78E−02)
f5(Quartik)	4.61E−127 (1.46E−126)	0.00E+00 (0.00E+00)	5.70E−71 (1.80E−70)	0.00E+00 (0.00E+00)	5.58E−04 (3.81E−04)	0.00E+00 (0.00E+00)
f6(Schwefel_2_21)	8.11E−02 (2.56E−01)	9.48E−03 (3.00E−02)	1.32E−01 (4.17E−01)	2.97E+00 (2.10E−01)	2.14E−01 (1.22E−01)	4.90E−03 (1.55E−02)
f7(Michalewicz)	3.79E−01 (1.20E+00)	7.04E−01 (2.23E+00)	2.15E−01 (6.80E−01)	2.54E−01 (1.27E+02)	−1.97E+00 (6.08E−01)	4.71E−01 (1.49E+00)
f8(Rastrigin)	3.33E+00 (1.05E+01)	7.13E+00 (2.26E+01)	6.10E+00 (1.93E+01)	3.21E+01 (3.67E−02)	6.54E+03 (4.15E+02)	9.61E+00 (3.04E+01)
f9(Zakharov)	2.97E+01 (9.39E+01)	3.55E−03 (1.12E−02)	1.28E+02 (4.04E+02)	2.32E+01 (4.23E−01)	2.56E−02 (1.45E−01)	2.82E−04 (8.93E−03)
f10(Powell)	1.02E−07 (3.23E−07)	6.13E−08 (1.94E−07)	2.88E−06 (9.12E−06)	2.88E+02 (1.29E+01)	2.56E−05 2.95E−05	1.33E−08 (4.19E−08)
−	8	7	8	7	8	
≈	0	2	0	2	2	
+	2	1	2	1	0	

Table 6 Test Statistics using Friedman's test

R2	785
k	2
n	13
Q	42.76923077
df	1
p	6.15934E−11
alpha	0.05
sig	yes

if during the evolutionary process if any individual fails to produce better results in successive generations and so the values should be readjusted by random values by the following strategy.

$$F_{i,I+1} = \begin{cases} F_{i,I} \; if \; ES_i < ES_{max} \\ F_{min} + \left(F_{max} - F_{min}\right) * rand(1, D) \; otherwise \end{cases} \tag{10}$$

$$Cr_{i,I+1} = \begin{cases} Cr_{i,I} \; if ES_i < ES_{max} \; if \; ES_i < ES_{max} \\ Cr_{min} + \left(Cr_{max} - Cr_{min}\right) * rand(1, D) \; otherwise \end{cases} \tag{11}$$

where $F_{i,I}$ is the scaling factor $and Cr_{i,I}$ is the crossover rate for X_{id}^{I} for generation $I.valuesof F_{min} and F_{max}$ are 0.5 and 1.0. Values of $Cr_{max} and Cr_{min}$ are 0.8 and 1.0 respectively. Values of rand (1, D) is in the random values in range [1, D] where D is the size (samples, 2). ES_i is the temp value of individual generation and ES_{max} is the max or best values obtained from that generation. We update best value only in case of success to save time. if competitor is better than the best one ever, we will update the new best value as the temp value.

In order to handle the problem related to Population diversity and premature convergence, our proposed strategy uses mutation operation to generate new offspring and then each offspring uses crossover with particle/particle as mentioned in Eq. (3) to generate new offspring. These newly generated particles are created with updated position. During the iteration process this new particle update their personal best as compared with all the other newly created offspring.

6.2.1 Comparison with PSO, jDE, SaDE and JADE

We have further compared our DEPSO algorithm with Canonical PSO [43], jDE [31], SaDE [27]] and JADE [11]. The parameter settings for our proposed algorithm is same as

mentioned in Table 1 for impartial comparison. For the algorithms in comparison, parameters settings are same as mentioned in the original papers. For this experiment we have used 11 well known benchmark functions. These benchmark functions are same as mentioned in these original variants research papers. The results are calculated by running the algorithm over 50 independent runs for D = 30 and 100. The results obtained are tabulated in Table 7 and 8.

For 30D problems, the performance for JADE is better in comparison to jDE, SaDE and PSO. The jDE and SaDE gets 2 best solutions due to the slow convergence issue. PSO is performing worst and is not giving any best solution or second-best solution on any of the benchmarking functions. This is due to the premature convergence issue with the PSO. JADE is giving better results because it shows high reliability, fast convergence and diversity improvement in mutation strategy. On the other hand, our proposed algorithm DEPSO is giving better results as compared to other algorithms and giving best solution on 9 benchmarking functions. This is because our proposed algorithm is having good diversity due to its adaptive mutation strategy based on sigmoid function.

Similarly, our proposed algorithms DEPSO is giving better results on 100D benchmarking functions in comparison to jDE, SaDE, JADE and PSO. This is due to the mutation strategy based on sigmoid function that provides more diversity and high convergency speed. Other algorithms except JADE are giving no better solution when compared with DEPSO algorithm. JADE with archive is giving better solution on only 2 benchmarking functions when compared with our proposed algorithm. But JADE is giving better results on jDE, SaDE and PSO algorithm. This is due to the adaptive parameter control mutation strategy that introduces population diversity in JADE.

To summarize, our proposed approach is showing better results than other algorithms in comparison both in terms of dimensionality and convergence speed. The main reason for giving better results are: PSO strategy for alternate mutation and adaptive parameter selection approach for DE parameters is giving more exploration ability to our proposed approach. Additionally, the adaptive crossover, scaling parameters selection and a sigmoid function are utilized to transform the velocity of PSO and to handle the multiplicity of population and convergence speed of our proposed approach (Table 9).

6.2.2 Comparison on high dimensional datasets

We have also evaluated the performance of our proposed algorithm on high dimensional problems (D = 500,1000) [44]. For this experiment we have used $NP = 5000$, the number of iterations is 5000 and 8 test functions that includes unimodal and multimodal functions. The mean and standard deviation are calculated and summarized in Sect. 8. Experimental outcome indicates that our proposed algorithm is outperformed when compared with classical DE and PSO. This is again due to the

Table 7 Mean and SD for PSO, SaDE, jDE, JADE and DEPSO on 11 functions on 30 D

Function	Gen	PSO	SaDE	jDE	JADE w/o archive	JADE with archive	DEPSO
Sphere	1500	9.6E − 42 (2.7E − 41)	4.5E − 20 (6.9E − 20)	2.5E − 28 (3.5E − 28)	1.8E − 60 (8.4E − 60)	1.3E − 54 (9.2E − 54)	9.96E − 307 (0.00E + 00)
schwefel 2.22	2000	9.3E − 21 (6.3E − 20)	1.9E − 14 (1.05E − 14)	1.5E − 23 (1.0E − 23)	1.8E − 25 (8.8E − 25)	3.9E − 22 (2.7E − 21)	4.39E − 172 (0.00E + 00)
schwefel 1.2	5000	2.5E − 19 (3.9E − 19)	9.0E − 37 (5.43E − 36)	5.2E − 14 (1.1E − 13)	5.7E − 61 (2.7E − 60)	6.0E − 87 (1.9E − 86)	0.00E + 00 (0.00E + 00)
schwefel 2.21	5000	4.4E − 14 (9.3E − 14)	7.4E − 11 (1.82E − 10)	1.4E − 15 (1.0E − 15)	8.2E − 24 (4.0E − 23)	4.3E − 66 (1.2E − 65)	7.65E − 78 (5.677E − 76)
rosenbrock	3000	2.5E + 01 (3.2E + 01)	2.1E + 01 (7.8E + 00)	1.3E + 01 (1.4E + 01)	8.0E − 02 (5.6E − 01)	3.2E − 01 (1.1E + 00)	2.07E − 18 (1.94E − 15)
Step	1500	8.0E − 02 (2.7E − 01)	0.0E + 00 (0.0E + 00)	0.0E + 00 (0.0E + 00)	0.0E + 00 (0.0E + 00)	0.0E + 00 (0.0E + 00)	5.19E − 01 (1.67E − 01)
Quartic	3000	2.5E − 03 (1.4E − 03)	4.8E − 03 (1.2E − 03)	3.3E − 03 (8.5E − 04)	6.4E − 04 (2.5E − 04)	6.8E − 04 (2.5E − 04)	5.97E − 05 (3.45E − 05)
schwefel 2.26	9000	2.4E + 03 (6.7E + 02)	4.7E + 00 (3.3E + 01)	0.0E + 00 (0.0E + 00)	0.0E + 00 (0.0E + 00)	7.1E + 00 (2.8E + 01)	4.23E + 03 (6.05E + 02)
rastrigin	1000	5.2E + 01 (1.6E + 01)	1.2E − 03 (6.5E − 04)	1.5E − 04 (2.0E − 04)	1.0E − 04 (6.0E − 05)	1.4E − 04 (6.5E − 05)	0.00E + 00 (0.00E + 00)
Ackley	2000	4.6E − 01 (6.6E − 01)	4.3E − 14 (2.6E − 14)	4.7E − 15 (9.6E − 16)	4.4E − 15 (0.0E + 00)	4.4E − 15 (0.0E + 00)	7.99E − 15 (1.72E − 15)
griewank	3000	1.1E − 02 (1.6E − 02)	0.0E + 00 (0.0E + 00)	0.0E + 00 (0.0E + 00)	0.0E + 00 (0.0E + 00)	2.0E − 04 (1.4E − 03)	2.78E − 09 (1.65E − 08)

Table 8 Mean and SD for PSO, SaDE, jDE, JADE and DEPSO on 11 functions on 100 D

Function	Gen	PSO	SaDE	jDE	JADE w/o archive	JADE with archive	DEPSO
Sphere	2000	6.0E−11 (2.5E−10)	2.9E−08 (3.2E−08)	5.0E−15 (1.7E−15)	1.2E−48 (1.5E−48)	5.4E−67 (1.6E−66)	3.26E−120 (4.83E−119)
Schwefel 2.22	3000	2.8E−04 (1.3E−03)	1.7E−05 (3.8E−06)	4.1E−15 (1.1E−15)	1.2E−26 (2.0E−26)	2.2E−37 (2.5E−37)	0.00E+00 (0.00E+00)
Schwefel 1.2	8000	1.2E+02 (6.7E+01)	2.4E−13 (5.2E−13)	5.4E−02 (2.7E−02)	1.2E−26 (2.0E−26)	2.2E−37 (2.5E−37)	1.77E−46 (3.80E−46)
Schwefel 2.21	15,000	4.9E+01 (2.5E+01)	1.1E+00 (4.0E−01)	3.1E−09 (5.9E−10)	1.9E−02 (1.5E−02)	3.2E−71 (8.3E−71)	1.03E−97 (8.76E−85)
Rosenbrock	6000	9.2E−03 (2.6E−03)	9.4E+01 (4.0E–01)	7.2E+01 1.1E+01)	5.6E–01 (1.4E+00)	4.0E–01 (1.2E+00)	1.82E−16 (4.19E−14)
Step	1500	4.7E+01 (7.9E+01)	0.0E+00 (0.0E+00)	0.0E+00 (0.0E+00)	1.6E−01 (3.7E−01)	0.0E+00 (0.0E+00)	9.16E−04 (3.92E+00)
Quartic	6000	1.3E+02 (4.8E+01)	1.0E−02 (4.9E−03)	8.1E−03 (9.0E−04)	1.1E−03 (2.1E−04)	7.8E−04 (1.4E−04)	9.33E−05 (4.56E−05)
Schwefel 2.26	9000	9.4E+03 1.2E+03)	1.1E−10 0.0E+00)	1.1E−10 (0.0E+00)	1.1E−10 (0.0E+00)	1.1E−10 (0.0E+00)	1.82E+04 (2.57E+03)
Rastrigin	3000	3.4E+02 (4.4E+01)	9.1E−03 (1.8E−03)	2.1E−04 (2.1E−04)	1.9E−01 (3.8E−02)	2.0E−01 (3.7E−02)	0.00E+00 (0.00E+00)
Ackley	3000	2.6E+00 (6.8E−01)	2.1E−07 (1.0E−07)	9.9E−14 (2.0E−14)	8.9E−15 (2.1E−15)	8.0E−15 (0.0E+00)	5.95E−14 (8.48E−14)
Griewank	3000	8.8E−02 (2.5E−01)	8.6E−13 (8.2E−13)	0.0E+00 0.0E+00)	3.9E−04 (2.0E−03)	1.5E−04 (1.0E−03)	3.22E−15 (4.00E−15)

co-operation strategy between DE and PSO which is working efficiently on high dimensional problems.

We have also calculated the average run time of each algorithm to demonstrate the computational complexity of our proposed algorithm. Table 10 shows the average run time of DE, PSO, ABC, FSDE, HDEPSO and our proposed DEPSO approach on 10 benchmark functions. From the table we can see that our approach has the better time complexity in comparison to other approaches. We can also confirm from the various experiments including higher dimensional data, that our approach is computationally faster and effective.

7 Conclusion

Differential evolution algorithm is dependent on its mutation and crossover approach. To improve this, a hybrid variant of DE and PSO is proposed in this paper. Initially our proposed algorithm starts with DE mutation strategy to improve the exploration capability and in the later stage, mutation approach of PSO is applied. This will help in increasing the convergence speed of our proposed approach. We have also used dynamic parameter selection approach, and this help DE-PSO to handle multiple optimization problems effectively.

The proposed DE-PSO algorithm is compared with conventional DE, PSO, ABC, FSDE and HDEPSO algorithm on 25, 50-dimensional test functions. The results show that our proposed algorithm is giving better results in comparison to other algorithms. We have further compared our DEPSO algorithm with Canonical PSO, jDE, SaDE and JADE on 30 and 100-dimensional test functions. Our proposed approach is showing better results than other algorithms in comparison both in terms of dimensionality and convergence speed. To further establish the robustness and effectiveness of our proposed algorithm, we have also evaluated the performance of our proposed algorithm on high dimensional problems (D = 500, 1000). We have also performed Friedman's test to statistically shows the performance of our proposed algorithm. We have also calculated the average run time of each algorithm to demonstrate the computational complexity of our proposed algorithm.

In the future work, we are planning to implement this algorithm in feature selection to improve the accuracy of clustering algorithm. Apply this algorithm on motion estimation to develop video coding application is also another area of exploration.

Table 9 Mean and SD for PSO, DE and DEPSO on Unimodal and Multimodal functions with D-500 and 1000

Function	Dim(D)	DE Mean (Std Dev)	PSO Mean (Std Dev)	DEPSO Mean (Std Dev)
Sphere	500	5.23E+03	3.53E+03	6.92E+02
	1000	(8.71E+03)	(1.71E+03)	(3.39E+03)
		6.27E+03	2.67E+03	8.47E+02
		(9.16E+03)	(1.16E+03)	(4.08E+03)
Schwefel's 2.22	500	5.00E−01	2.00E−01	3.51E−03
	1000	(2.14E−03)	(4.14E−03)	(8.19E−06)
		5.00E−01	5.00E−01	5.00E−01
		(1.58E−02)	(1.58E−02)	(1.58E−02)
Schwefel's 2.21	500	3.96E−01	9.96E−02	5.00E−03
	1000	(1.18E−01)	(1.34E−01)	(1.13E−06)
		9.10E−03	1.10E−03	3.19E−04
		(8.33E−03)	(3.13E−03)	(2.32E−03)
Schwefel	500	−5.06E+03	−4.06E+03	−2.10E+03
	1000	(0.0E+00)	(0.0E+00)	(0.0E+00)
		−8.51E+03	−6.51E+03	−1.15E+04
		(0.0E+00)	(0.0E+00)	(0.0E+00)
Rastrigin	500	1.85E+02	5.85E+02	5.75E+01
	1000	(3.81E+01)	(1.83E+01)	(6.09E+01)
		2.00E+02	1.00E+02	4.79E+01
		(3.29E+01)	(2.29E+01)	(5.54E+01)
Ackley	500	1.05E+01	5.01E+01	1.30E+00
	1000	(5.31E+00)	(1.51E+00)	(3.57E+00)
		1.17E+01	7.71E+01	1.26E+00
		(4.86E+00)	(6.84E+00)	(3.65E+00)
Griewank	500	4.20E+02	2.24E+02	1.57E+01
	1000	(2.03E+02)	(3.02E+02)	(7.66E+01)
		5.09E+02	2.09E+02	1.88E+01
		(2.03E+02)	(1.03E+02)	(8.37E+01)
Rosenbrock	500	8.88E+06	4.84E+06	7.87E+05
	1000	(1.90E+07)	(2.19E+07)	(6.75E+06)
		8.65E+06	5.61E+06	6.14E+05
		(1.79E+07)	(1.79E+06)	(5.83E+06)

Appendix

Formula	Ranges	Optimal
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-10, 10]$	0
$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	$[\pi 32.32]$	0
$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x}{\sqrt{i}}\right) + 1$	$[-600, 600]$	0
$f_5 = \sum_{i=1}^{n} ix^4 + random(0, 1)$	$[-1.28, 1.28]$	0
$f_6(x) = \max\{\lvert x_i\rvert, 1 \le x_i \le D\}$	$[-100, 100]$	−4.18
$f_7(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0
$+(\sum_{i=1}^{n} 0.5ix_i)^2$	$[-5, 10]$	0
$f_8(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$		
$f_9(x) = \sum_{i=1}^{n}\sin(x_i) sin^{20}\left(\frac{ix_i^2}{\pi}\right)$	$[0, \pi]$	9.66015
$f_{10}(x) = \sum_{i=1}^{n/4}\begin{bmatrix}(x_{4i-3} \\ +10x_{4i-2})^2 \\ +5(x_{4i-1})^2 \\ +(x_{4i-2} - 2x_{4i-1})^4 \\ +10(x_{4i-3} - x_{4i})^4\end{bmatrix}$	$[-4.5]$	0

Table 10 Average execution time on benchmark functions

Function	DE	PSO	ABC	FSDE	HDEPSO	DEPSO
f1 (Sphere)	3.9632	3.9234	10	6.5	4.65	3.7943
f2(Rosenbrock)	11.8317	7.9096	7.9113	12.4852	8.34	7.9002
f3(Ackley)	18.9	12.638	12.4852	11.8317	12.476	5.58
f4(Griewank)	6.2748	6.2409	6.8626	8.79	7.546	5.3823
f5(Quartik)	9.84	6.8	7.9002	9.3	8.342	6.2
f6(Schwefel_2_21)	9.9478	10.267	10.0569	7.3456	9.456	6.64
f7(Michalewicz)	6.8	6.74	6.876	5.9432	7.567	5.2
f8(Rastrigin)	9.84	7.1071	6.8626	6.45	8.3456	6.1338
f9(Zakharov)	8.7228	8.1109	8.7228	9.6	7.435	6.26
f10(Powell)	6.45	10.679	10.5832	6.94	8.435	6.34

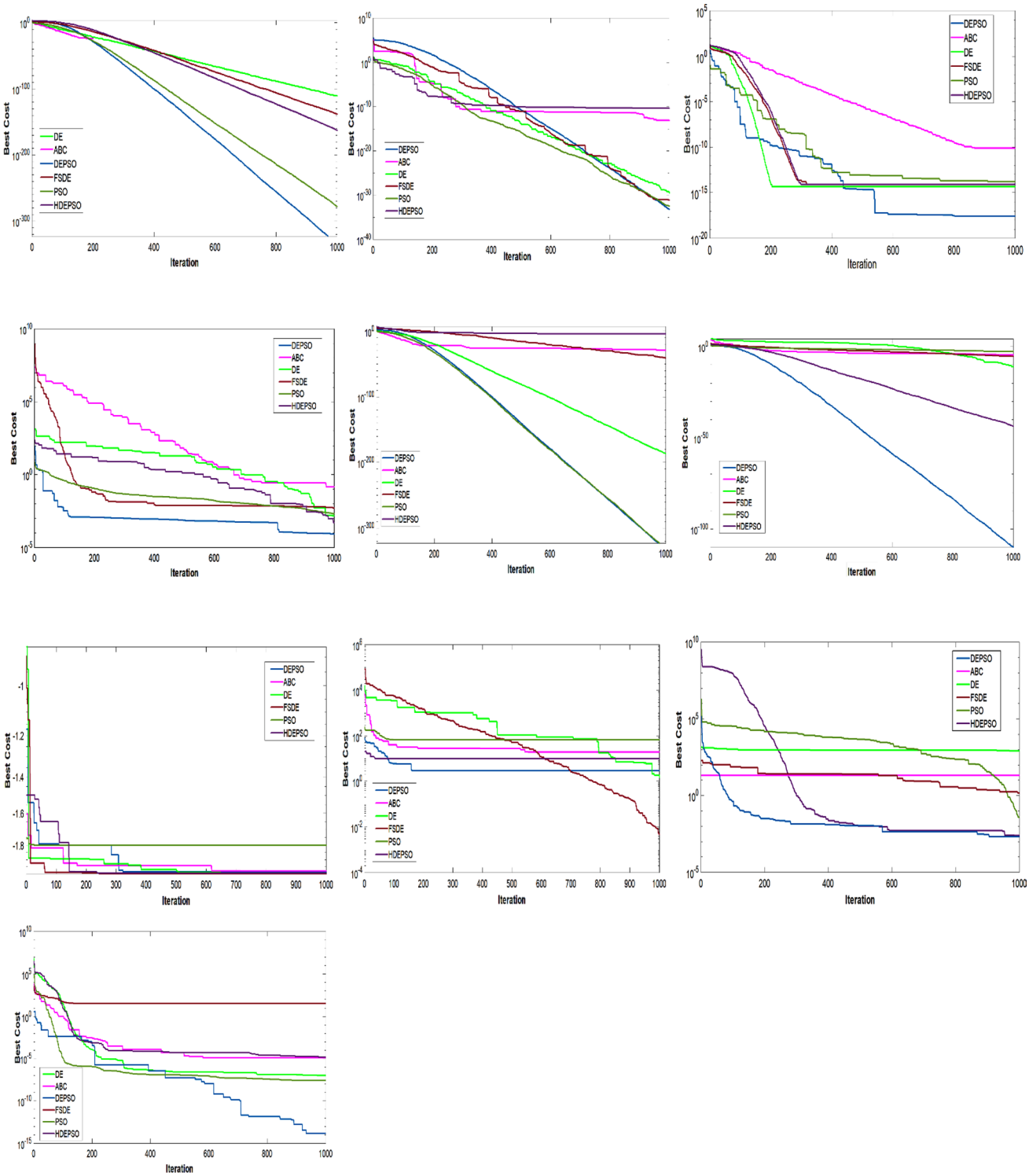

Fig. 2 Best value comparison graph of Sphere, Rosenbrock, Ackley, Griewank, Quartik, Schwefel_2_21, Michalewicz, Rastrigin, Zakharov, Powell on 25 D

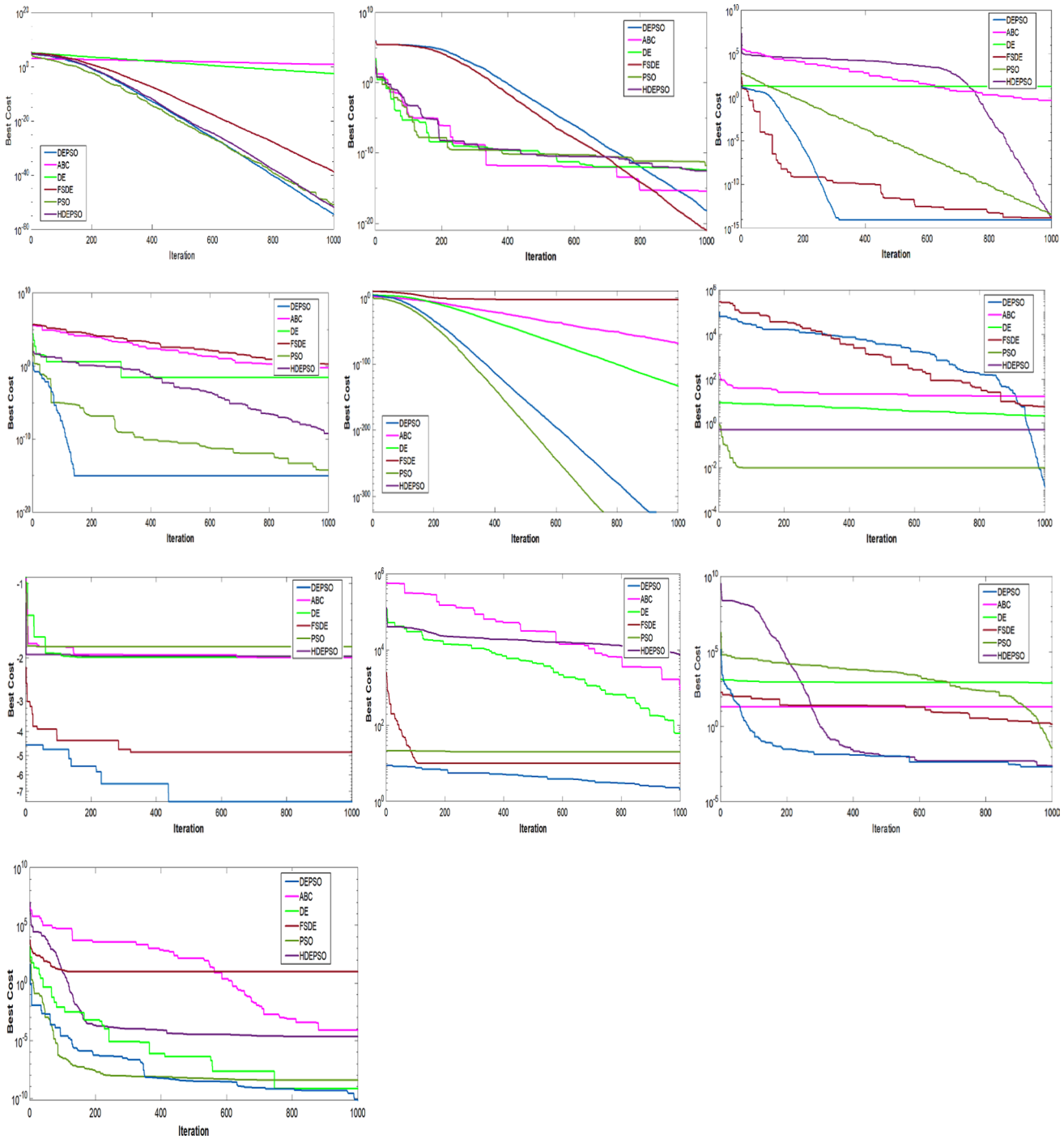

Fig. 3 Best value comparison graph of Sphere, Rosenbrock, Ackley, Griewank, Quartik, Schwefel_2_21, Michalewicz, Rastrigin, Zakharov, Powell on 50 D

References

1. Storn R, Price K (1995) Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Int Comput Sci Inst Technol Rep TR-95–012
2. Storn R, Price K (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
3. Ayala HVH, Santos FMD, Mariani VC, Coelho LDS (2019) Image thresholding segmentation based on a novel beta differential evolution approach. Expert Syst Appl 42(4):2136–2142

4. Cervantes-Sanchez F, Cruz-Aceves I, Hernandez-Aguirre A, Solorio-Meza S, Cordova-Fraga T, Aviña-Cervantes JG (2018) Coronary artery segmentation in X-ray angiograms using gabor filters and differential evolution. Appl Radiat Isot 138:18–24
5. Hou Y, Zhao L, Lu H (2018) Fuzzy neural network optimization and network traffic forecasting based on improved differential evolution. Future Gen Comput Syst 81:425–432
6. Wang T, Liu C, Wang L, Ma B, Gu X (2018) Evolution modeling with multi-scale smoothing for action recognition. J Vis Commun Image Represent 55:778–788
7. Civicioglu P, Besdok E (2019) Bernstain-search differential evolution algorithm for numerical function optimization. Expert Syst Appl 138:112831
8. Zhang Q, Zou D, Duan N, Shen X (2019) An adaptive differential evolutionary algorithm incorporating multiple mutation strategies for the economic load dispatch problem. Appl Soft Comput 78:641–669
9. Qin A, Huang V, Suganthan P (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolu Comput 13(2):398–417
10. Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput 9(6):448–462
11. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
12. Wang S, Li Y, Yang H, Liu H (2018) Self-adaptive differential evolution algorithm with improved mutation strategy. Soft Comput 22(10):3433–3447
13. Alswaitti M, Albughdadi M, Isa NAM (2019) Variance-based differential evolution algorithm with an optional crossover for data clustering. Appl Soft Comput 80:1–17
14. Ramadas M, Abraham A, Kumar S (2019) FSDE-Forced Strategy Differential Evolution used for data clustering. Journal of King Saud University - Computer and Information Sciences 31:52–61
15. Kennedy J and Eberhart R (1995) Particle swarm optimization in IEEE international conference on neural networks
16. Prajapati A, Chhabra JK (2018) A particle swarm optimization-based heuristic for software module. Arab J Sci Eng 43:7083–7094
17. Junxiang L, Jianqiao C (2019) Solving time-variant reliability-based design optimization by PSO-t-IRS: a methodology incorporating a particle swarm optimization algorithm and an enhanced instantaneous response surface. Reliab Eng Syst Saf 191:106580
18. Matos J, Faria RP, Nogueira IB, Loureiro JM, Ribeiro AM (2019) Optimization strategies for chiral separation by true moving bed chromatography using particles swarm optimization (PSO) and new parallel PSO variant. Comput Chem Eng 123:344–356
19. Xie XF, Zhang WJ, Yang ZL (2002) A dissipative particle swarm optimization. Congr Evolu Comput 2:1456–1461
20. Clerc M, Kennedy J (2002) The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans on Evolu Comput 6(1):58–73
21. Lin G, Zhang J, Liu Z (2016) Hybrid particle swarm optimization with differential evolution for numerical and engineering optimization. Int J Autom Comput 15(1):103–114
22. Wang H, Zuo LL, Liu J, Yi WJ, Niu B (2018) Ensemble particle swarm optimization and differential evolution with alternative mutation method. Nat Comput 11655:1–1
23. Wanga S, Li Y, Yang H (2019) Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. Appl Soft Comput J 81:105496
24. Pérez-González A, Begovich-Mendoza O, Ruiz-León J (2018) Modeling of a greenhouse prototype using PSO and differential evolution algorithms based on a real-time LabViewTM application. Appl Soft Comput 62:86–100
25. Ahmadianfar I, Khajeha Z, Asghari-Pari S-A, Chu X (2019) Developing optimal policies for reservoir systems using a multi-strategy optimization algorithm. Appl Soft Comput 80:888–903
26. Dash J, Dam B, Swain R (2019) Design and implementation of sharp edge FIR filters using hybrid differential evolution particle swarm optimization. AEU - International Journal of Electronics and Communications 114:344–356
27. Qin A, Suganthan P (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of IEEE congress on evolutionary computation, IEEE. Edinburgh, Scotland, UK
28. Ali M, Pant M, Abraham A (2013) Unconventional initialization methods for differential evolution. Appl Math Comput 219(9):4474–4494
29. Poikolainen I, Neri F, Caraffini F (2015) Cluster-based population initialization for differential evolution frameworks. Inf Sci 297:216–235
30. Sun G, Xu G, Gao R, Liu J (2019) A fluctuant population strategy for differential evolution. Evol Intell
31. 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
32. Shao C, Cai Y, Fu S, Li J, Luo W (2018) An enhanced utilization mechanism of population information for differential evolution. Evol Intell
33. Annepu V, Rajesh A (2019) Implementation of self adaptive mutation factor and cross-over probability based differential evolution algorithm for node localization in wireless sensor networks. Evol Intell 12:469–478
34. Zhang X, Zhang X (2020) A set-based differential evolution algorithm for QoS-oriented and cost-effective ridesharing. Appl Soft Comput 96:106618
35. Xin B, Chen J, Zhang J, Fang H, Peng ZH (2012) Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. IEEE Trans Syst Man Cybern Syst 42(5):744–767
36. Sun J, Zhang Q, Tsang EPK (2005) DE/EDA: a new evolutionary algorithm for global optimization. Inf Sci 169(3–4):249–262
37. Wang L, Ye Xu, Lingpo Li (2011) Parameter identification of chaotic systems by hybrid Nelder-Mead simplex search and differential evolution algorithm. Expert Systems with Applications 38(4):3238–3245
38. Guo H, Li Y, Li J, Sun H, Wang D, Chen X (2014) Differential evolution improved with self-adaptive control parameters based on simulated annealing. Swarm Evol Comput 19:52–67
39. Keshk M, Singh H, Abbass H (2018) Automatic estimation of differential evolution parameters using hidden markov models. Evol Intell 10:77–93
40. Tian G, Ren Y, Zhou M (2016) Dual-objective scheduling of rescue vehicles to distinguish forest fires via differential evolution and particle swarm optimization combined algorithm. IEEE Trans Intell Trans Syst 18(11):3009–3021
41. Karaboga D (2010) Artificial bee colony algorithm". Scholarpedia. Swarm Evol Comput 5(3):6915
42. Nasimul N, Danushka B, Hitoshi I (2006) An adaptive differential evolution algorithm. In IEEE Transaction on. Evolutionary Computation
43. Trelea I (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85(6):317–325
44. Liu Y, Yao X, Zhao Q, Higuchi T (2001) Scaling up fast evolutionary programming with cooperative coevolution. South Korea, No.01TH8546), Seoul