Journal of Bionic Engineering (2022) 19:1140–1160
https://doi.org/10.1007/s42235-022-00190-4

RESEARCH ARTICLE

Comparative Performance Analysis of Differential Evolution Variants on Engineering Design Problems

Sanjoy Chakraborty[1,2] · Apu Kumar Saha[3] · Sushmita Sharma[3] · Saroj Kumar Sahoo[3] · Gautam Pal[4]

Received: 29 August 2021 / Revised: 21 March 2022 / Accepted: 23 March 2022 / Published online: 13 June 2022

Abstract
Because of their superior problem-solving ability, nature-inspired optimization algorithms are being regularly used in solving complex real-world optimization problems. Engineering academics have recently focused on meta-heuristic algorithms to solve various optimization challenges. Among the state-of-the-art algorithms, Differential Evolution (DE) is one of the most successful algorithms and is frequently used to solve various industrial problems. Over the previous 2 decades, DE has been heavily modified to improve its capabilities. Several DE variations secured positions in IEEE CEC competitions, establishing their efficacy. However, to our knowledge, there has never been a comparison of performance across various CEC-winning DE versions, which could aid in determining which is the most successful. In this study, the performance of DE and its eight other IEEE CEC competition-winning variants are compared. First, the algorithms have evaluated IEEE CEC 2019 and 2020 bound-constrained functions, and the performances have been compared. One unconstrained problem from IEEE CEC 2011 problem suite and five other constrained mechanical engineering design problems, out of which four issues have been taken from IEEE CEC 2020 non-convex constrained optimization suite, have been solved to compare the performances. Statistical analyses like Friedman's test and Wilcoxon's test are executed to verify the algorithm's ability statistically. Performance analysis exposes that none of the DE variants can solve all the problems efficiently. Performance of SHADE and ELSHADE-SPACMA are considerable among the methods used for comparison to solve such mechanical design problems.

Keywords Differential evolution · Metaheuristics · IEEE CEC · Mechanical design problem

1 Introduction

Optimization is a sort of decision-making and one of the essential quantitative techniques in decision-making machinery. Under specific predetermined conditions, decisions must be made to optimize one or more objectives. Most real-world problems may be stated in optimization models, including numerous criteria and goals [1]. Optimization techniques are based on biology, artificial intelligence, nature, and scientific areas such as physics, chemistry, etc. [2]. The scope of research with optimization is vast. Optimization methods can solve problems from linear programming, integer programming, quadratic programming, non-convex optimization, engineering, science, and economics, etc. Solving these problems becomes more complicated when the nature of the problem cannot be known in advance. Without knowing the nature of the problem, it is also tough to select a proper method for finding the solution [1].

Furthermore, an equilibrium between exploration and exploitation, also known as global and local search, is critical in any optimization approach. An algorithm must maintain a healthy balance between exploration and exploitation to be effective [3]. According to the theory of "No Free Lunch" (NFL) [4], theorems, no algorithm can solve problems of all types with equal efficiency. Even the same algorithm may give different solutions based on different

✉ Apu Kumar Saha
apusaha.nita@gmail.com

1 Department of Computer Science and Engineering, National Institute of Technology Agartala, Agartala, Tripura 799046, India

2 Department of Computer Science and Engineering, Iswar Chandra Vidyasagar College, Belonia, Tripura 799155, India

3 Department of Mathematics, National Institute of Technology Agartala, Agartala, Tripura 799046, India

4 Department of Computer Science and Engineering, Tripura Institute of Technology, Narsingarh, Tripura 799015, India

parameter values. This has led to the development of new algorithms and their modified forms by researchers worldwide. The last few decades have seen a surge in developing algorithms classified as meta-heuristics.

Meta-heuristic methods being different from deterministic methods can search the global solution without gradient information of the optimization issue [5]. The Differential Evolution (DE) technique is a part of evolutionary programming. It is designed by R. Storn and K. Price [6] to optimize issues over the continuous field. In DE, the worth of every variable is a real number. DE utilizes mutation and selection during the search to guide the pursuit toward the potential zone. The standard DE method comprises four essential phases—initialization of population, calculation of donor vector, the crossover between donor vector and target vector to form a trial vector, selection of target vector for the next generation from trial vector, and target vector of the present generation. The last three stages of DE execute as a circle for ensuing DE generations until a termination criterion is triggered.

1.1 Advantages and Disadvantages of DE

DE is a powerful and valuable global optimizer. DE is a population-based method that belongs to the evolutionary algorithm's category. In DE, siblings are formed by disrupting the arrangements with a scaled distinction between two randomly selected individuals from the population. It distinguishes DE from other evolutionary methods. DE employs a process of coordinated substitution. "On the off occasion that the trial vector is superior to the parent solution, it is chosen." In contrast to a few other evolutionary calculation methods, DE is a simple method that can be implemented with a few lines of code in any standard programming language. Also, DE requires not many control parameters (3 to be exact: the scale factor, the crossover rate, and the population size) a component that makes it simple to use for the experts", as per Das et al. [7]. The authors likewise referenced that "no other single search paradigm has been able to secure competitive ranking in nearly all the CEC competitions on a single objective, constrained, dynamic, large-scale, multi-objective, and multimodal optimization problems." DE is exceptionally adequate to researchers and specialists because of its reliability and high performance; Neri et al. [8]. The writers of the similar article likewise called attention to the explanation for the massive achievement of DE is a suggested self-variation delimited in the design of the actual DE technique. As solutions are spread inside the pursuit space, the algorithm should be explorative in its beginning phase. In the later stage of the optimization process, exploitation is fundamental. DE is profoundly explorative toward the start of the process, and bit by bit, it becomes exploitative.

Regardless of these benefits, DE has a couple of burdens. It tends to be composed that the search process is indeed negotiated if promising arrangements are not discovered in hardly any explorative moves. The proficient working of DE relies upon the three control parameters referenced previously. The population size is related to the possible moves. A small population can have a predetermined number of developments, and an enormous population has numerous exercises. If the population is small, that may lead to premature convergence, Eiben et al. [9]. The scaling factor and crossover rate value play a crucial role in the algorithm's well-functioning. Still, the selection of these values is a tedious task. The problem of parameter setting can be typical while solving real-life optimization problems with larger dimensions; the risk of stagnation increases in DE with increased dimensionality, Zamuda et al. [10]. Not only the dimensionality problem, but DE is also inefficient in noisy optimization problems. Standard DE can fail in handling the noisy fitness function, Krink et al. [11]. Based on the above facts, the scientific community is aware that although DE is a decent algorithm, there is extensive scope for updating the algorithmic structure.

1.2 Types of Modifications on DE

Several works on DE have been done over the previous 2 decades, and new modified forms of DE have been proposed using the methodologies which are mentioned in following subsections. Also, a pictorial representation of different DE variants using different strategies is shown in Fig. 1.

1.2.1 Modification in Population Initialization

In population-based search methods, the initial population generated through the system produced arbitrary numbers that generally follow a uniform distribution. However, this is a straightforward strategy for instating the population; specialists saw that altering the introduction interaction may help in improving the effectiveness of the method. Therefore, an assortment of initialization techniques has been proposed in the literature. For the most part, these changes depend on one or the other contracting the search space before all else itself to empower quicker convergence or, again, depend on isolating the population into more modest subgroups of populations that can simultaneously tune the population adaptively.

1.2.2 Alteration in Mutation Strategy

The mutation phase of DE is the most important, since it introduces a new individual into the population. The nature of the problem determines the choice of mutation strategy. The amplification factor's strategy determines the

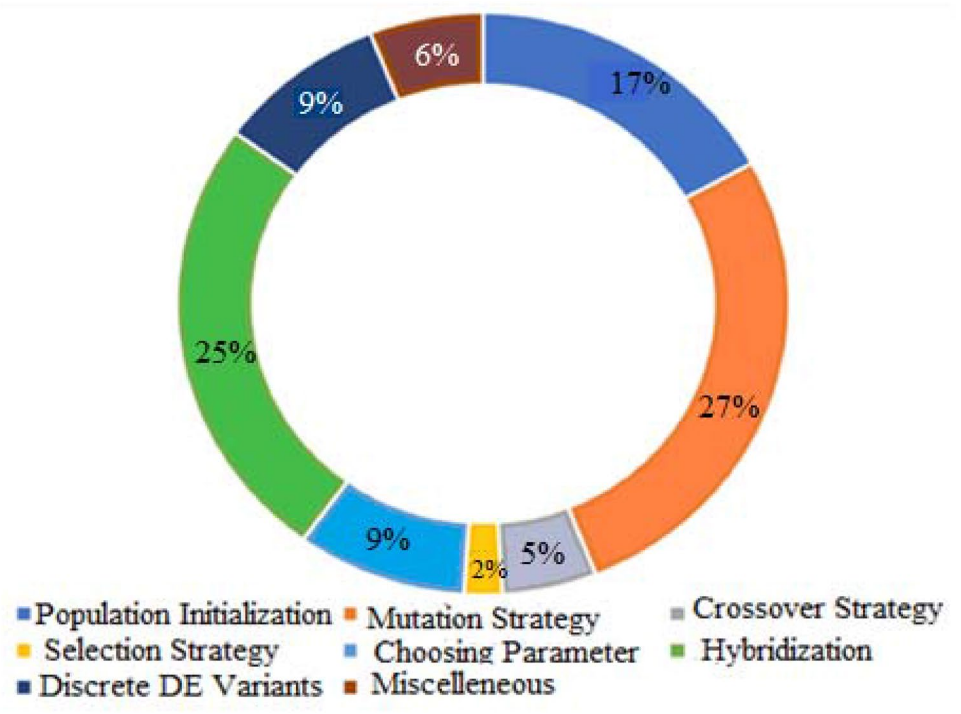

Fig. 1 Percentage of different DE variants developed using various strategies

population's diversity. The researchers put a lot more work into designing algorithms by changing current methods, merging many techniques in one algorithm, and adaptively determining the strategy.

1.2.3 Variation in Crossover Strategy

The trial vector is created from the donor and target vector using the crossover approach. Exponential crossover is used in the original DE. Later on, the binomial kind of crossover gained prominence. Since DE's debut, scholars have proposed several crossover approaches.

1.2.4 Change in the Selection Strategy

DE has a novel selection mechanism that isolates it from other methods. Even though alterations proposed in the selection mechanism are restricted to a couple of papers, analysts have shown that reasonable changes can also help improve the method's efficacy.

1.2.5 Variation in Choosing the Parameter

Parameters are the essential elements of an evolutionary process. Canonical DE uses three basic parameters, mutation factor, crossover rate, and population size. The selection of parameters can be deterministic, adaptive, or self-adaptive. In deterministic parameter selection, values are modified using some deterministic rule after a fixed number of generations is elapses. Adaptive parameter selection is when parameters are changed according to feedback from the search process. During self-adaptation, parameters encoded into the chromosomes, the better value of these parameters produces better offspring, propagating to the next generation.

1.2.6 Hybridization

Sometimes the effectiveness of an algorithm may be increased in terms of convergence speed, computational complexity, ability to get out from local optima, identifying the stagnation, etc., by utilizing the working procedure of one or more other algorithms. Therefore, hybridization is the technique of merging two or more algorithms to design a robust method.

1.3 Few Works on Verification of Performance and Real-World Problem Solving

A reasonable number of surveys using the basic meta-heuristic algorithms or their variants have been carried out. Nama et al. [12] studied the performance of Harmony Search Algorithm (HSA), Teaching–Learning Based Optimization (TLBO), and Particle Swarm Optimization (PSO) for finding the total active earth force on the back of a retaining wall. Yildiz et al. [5] solved six mechanical engineering problems utilizing ten meta-heuristics and presented a comparative study on their effectiveness. Kashani et al. [13] evaluated the uses of PSO on geo-specialized issues and finally offered a comparative study solving three geotechnical engineering problems using the PSO variants. Foroutan et al. [14]

proposed the green hybrid traction power supply substation model and investigated the performance of a few recent optimization algorithms on the problem. The effect of various nature-inspired algorithms on intrusion detection problems is tested by Thakur & Kumar [15]. Effectiveness of some recently designed algorithms and few DE variants tested on economic optimization of cooling tower problem by Patel et al. [16]. Performance of 12 meta-heuristics on wind farm layout optimization problem studied by Kunakote et al. [17]. Nama et al. [18] modified the Backtracking Search algorithm (BSA), incorporating a new adaptive control parameter, and used the modified method to determine active earth pressure on retaining wall supporting c-Φ backfill using the pseudo-dynamic method. Demirci and Yıldız [19] designed a novel hybrid approach, referred to as Hybrid Gradient Analysis (HGA) which is introduced for the evaluation of both convex and concave constraint functions in Reliability-Based Design Optimization (RBDO). Yıldız et al. [20] developed Henry Gas Solubility Optimization (HGSO) algorithm and solved the shape optimization of a vehicle brake pedal. Champasak et al. [21] designed a new self-adaptive meta-heuristic based on decomposition and solved unmanned aerial vehicle (UAV) problem with six objective functions. Sharma et al. [22] integrated the mutualism phase of the SOS algorithm with BOA and optimized some engineering design problems. Yildiz et al. [23] used the Butterfly Optimization Algorithm (BOA) to optimize coupling with a bolted rim problem. They also used it to solve the shape optimization of a vehicle suspension arm. Nama et al. [24] introduced a new variant of Symbiotic Organisms Search (SOS) algorithm with self-adaptive benefit factors and modified the mutualism phase. The authors used the new algorithm to solve five real-world problems. Yıldız et al. [25] used Equilibrium Optimization Algorithm (EOA) to solve a structural design optimization problem for a vehicle seat bracket. Yıldız et al. [26] used the Sine Cosine Algorithm (SCA) to solve the shape optimization of a vehicle clutch lever. Panagant et al. [27] used Seagull Optimization Algorithm (SOA) to solve the shape optimization of a vehicle bracket. The design problem is to find structural shape while minimizing structural mass and meeting a stress constraint. Dhiman et al. [28] introduced the evolutionary multi-objective version of the Seagull Optimization Algorithm (SOA), entitled Evolutionary Multi-objective Seagull Optimization Algorithm (EMoSOA). Twenty-four benchmark functions and four real-world engineering design problems are validated using the proposed algorithm. Chakraborty et al. [29] designed a modified WOA and applied it to optimize real-world problems from civil and mechanical engineering disciplines. Yıldız et al. [30] developed a new approach based on the Grasshopper Optimization Algorithm and Nelder–Mead Algorithm to optimize robot gripper problem with a fast and accurate solution. Additionally, vehicle side crash design, multi-clutch disk, and manufacturing optimization problems were also solved with the developed method. Sharma et al. [31] designed a balanced variant of BOA incorporating mutualism and parasitism phases of SOS with the basic BOA. Image segmentation problem with multilevel thresholding approach was solved using the new algorithm. Chakraborty et al. [32] modified the Whale Optimization Algorithm (WOA) and segmented the COVID-19 X-ray images to diagnose the disease easily.

Though all the CEC-winning algorithms are highly efficient in solving optimization problems, comparing performance among these efficient algorithms can be an attractive effort. With this motivation in this study, DE and its eight CEC-winning variants, SaDE, jDE, SHADE, LSHADE, LSHADE-EpSin LSHADE-cnEpSin, LSHADE-SPACMA, and ELSHADE-SPACMA, are selected for the experiment. Table 1 displays the rank of the algorithms and the year of holding the position. First, CEC 2019 and CEC 2020 bound-constrained suite is evaluated using the chosen methods, and then, a total of six real-world problems are solved. Among the problems selected, one is an unconstrained problem from CEC 2011 problem suite and five other constrained mechanical engineering design problems. Four mechanical engineering problems are taken from CEC 2020 non-convex constrained optimization suite. Statistically, the performance of the algorithms is analyzed using non-parametric tests like Friedman's test and Wilcoxon test.

The rest of the paper is organized as follows: Sect. 2 summarizes the algorithms employed here for comparison. Evaluated IEEE CEC 2019 and CEC 2020 results are tabulated, and a discussion on the results is given in Sect. 3. Section 4 represents a brief discussion of the real-world problems employed and a discussion on the results evaluated by the algorithms. Statistical analysis of the evaluated numerical data is carried out in Sect. 5. Discussion on the evaluated run time of the real-world problems is given in Sect. 6. Section 7, finally, concludes the study with future extensions.

Table 1 List of DE variants employed

DE variants	Year of competition	Rank
SaDE [33]	2005	3rd
jDE [34]	2009	1st
SHADE [35]	2013	4th
LSHADE [36]	2014	1st
LSHADE-Epsin [37]	2016	1st
LSHADE-cnEpsin [38]	2017	2nd
LSHADE-SPACMA [39]	2017	3rd
ELSHADE-SPACMA [40]	2018	3rd

2 Brief Description of DE and Its Variants Employed

A brief description of DE and its CEC-winning variants preferred here for comparison is given in this section.

2.1 Differential Evolution (DE) [6]

Differential Evolution is a population-based, stochastic optimization algorithm. It is used for solving the nonlinear optimization problem. It is a parallel direct search technique that uses population size (N_p) parameter vectors as a population for each generation. Here, weighted difference vector between two population members is added to a third member to produce a new parameter vector. The resultant vector, having a lower objective function value than a predetermined population member, is selected. This selected vector will replace the vector with which it has been compared in the next generation. The performance of DE depends on the proper selection of the trial vector generation technique and corresponding control parameter values. Finding of most suitable method and associated parameter settings in a trial-and-error approach involves more computational costs. Different approaches may need to couple with different parameter settings in various stages of evolution to get the best performance. DE has three steps in each generation: mutation, crossover, and selection. In the mutation phase, each individual in the population produces a respective mutation vector based on some strategy. The mutation vector and target vector exchange internal components during crossover to create a resultant vector. The selection step decides which vectors enter the next generation using greedy binary selection between the target and trial vectors. $DE/rand/1$ is the mutation strategy used in basic DE. Later on, this strategy is altered using diverse concepts to modify the DE algorithm. The well-known mutation strategies used in DE algorithms are given below

$rnd/1/bml$:

$$\boldsymbol{p}'^{(k)} = \boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd1}} + F.\left(\boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd2}} - \boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd3}}\right); \tag{1}$$

$rnd/2/bml$:

$$\boldsymbol{p}'^{(k)} = \boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd1}} + F.\left(\boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd2}} - \boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd3}}\right) + F.\left(\boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd4}} - \boldsymbol{p}^{\boldsymbol{k}}_{\mathbf{rnd5}}\right); \tag{2}$$

$Best/1/bml$:

$$\boldsymbol{p}'^{(k)} = \boldsymbol{p}^{k}_{\mathbf{best}} + F.\left(\boldsymbol{p}^{k}_{\mathbf{rnd1}} - \boldsymbol{p}^{k}_{\mathbf{rnd2}}\right); \tag{3}$$

$Best/2/bml$:

$$\boldsymbol{p}'^{(k)} = \boldsymbol{p}^{k}_{\mathbf{best}} + F.\left(\boldsymbol{p}^{k}_{\mathbf{rnd1}} - \boldsymbol{p}^{k}_{\mathbf{rnd2}}\right) + F.\left(\boldsymbol{p}^{k}_{\mathbf{rnd3}} - \boldsymbol{p}^{k}_{\mathbf{rnd4}}\right); \tag{4}$$

$Current - to - best/1/bml$:

$$\boldsymbol{p}'^{(k)} = \boldsymbol{p}^{k} + F.\left(\boldsymbol{p}^{k}_{\mathbf{best}} - \boldsymbol{p}^{k}_{\mathbf{rnd1}}\right) + F.\left(\boldsymbol{p}^{k}_{\mathbf{rnd2}} - \boldsymbol{p}^{k}_{\mathbf{rnd3}}\right); \tag{5}$$

$Current - to - pbest/1/bml$:

$$\boldsymbol{p}'^{(k)} = \boldsymbol{p}^{k} + F.\left(\boldsymbol{p}^{k}_{\mathbf{best}} - \boldsymbol{p}^{k}\right) + F.\left(\boldsymbol{p}^{k}_{\mathbf{rnd1}} - \boldsymbol{p}^{k}_{\mathbf{rnd2}}\right). \tag{6}$$

In the above equations, p^k is the kth solution of the population (P) and $\boldsymbol{p}'^{(k)}$ is the donor vector. Crossover types used in DE can be binomial or exponential. Here, bml signifies the binomial crossover.

2.2 Differential Evolution Algorithm with Strategy Adaptation (SaDE) [33]

Trial vector generation methods and related control parameters are slowly self-adapted by learning from their past experiences in the Self-adaptive DE (SaDE) technique to produce the best solutions. It keeps a candidate pool of many powerful trial vector generation techniques. A particular method is selected from the candidate pool in the evolution process for each target vector in the present population. The collection contains four mutation strategies, namely, $\varepsilon DE/rand/1/bin, DE/rand - to - best/2/bin, DE/rand/2/bin$, and $DE/current - to - rand/1\varepsilon$. A technique is chosen based on past probability learned of producing favorable solutions, and this is applied to perform mutation operation. To generate a trial vector in the SaDE algorithm, control parameters are assigned probabilistically to each target vector in the present population. The probabilities are slowly learned from the experience to produce better solutions. Here, the parameter resembled a normal distribution with a mean value of 0.5 and a standard deviation of 0.3. A batch of values is sampled randomly and used in each target vector in the present population. In this way, for small values, ' exploitation' and large values' exploration' are maintained throughout the evolution process.

2.3 Self-adapting Control Parameters in Differential Evolution (jDE) [34]

It is one of the most efficient DE variants. jDE uses a self-adaptive control technique to change the control parameters. The control parameters are adjusted with the Evolution. Here, user needs not to assume the appropriate values, which are problem-dependent. This technique changes the control parameters F and c^r during the run. The third control parameter N_p, the number of members in a population is not changed during the run. The control parameters F and c^r are adjusted during the evolution process, and both are applied at each level. The upper and lower value of F is 0.1 and 0.9, respectively. c^r takes a value between (0,1). F and c^r are evaluated using the following equations:

$$F^k_{g+1} = \begin{cases} F_l + rnd * F_u \; if \; rnd < \varnothing^1 \\ F^k_g \; Otherwise \end{cases} \tag{7}$$

$$C^{r(k)}_{g+1} = \begin{cases} rnd \; if \; rnd < \varnothing^2 \\ C^{r(k)}_g \; Otherwise \end{cases}. \tag{8}$$

Value of $\varnothing^1$ and $\varnothing^2$ remain fixed, and it is 0.1. Superior values of these control parameters generate better individuals. This individual produces offspring and propagates these better parameter values. In this approach, multiple runs are not needed to adjust control parameters. Self-adaptive DE is more independent than DE.

2.4 Success-History-Based Parameter Adaptation for Differential Evolution (SHADE) [35]

SHADE is a kind of adaptive DE. The origin of this algorithm is JADE and uses the $current - to - pbest/1$ mutation strategy, an external archive, and adaptively controls the parameter values F and c^r. It uses a historical memory of successful control parameter settings to guide the selection of future control parameter values. It uses a historical memory $M_cr\&M_F$ that stores set of c^r & F values which achieved a better result in the past. The SHADE approach maintains a historical memory with H entries for both DE control parameters. New c^r & F pair is assessed directly by sampling the parameter space close to one of these stored pairs as per the following equations:

$$c^r_i = \begin{cases} 0, if \; M_{c^r,rnd_i} = \perp \\ rndn_i\left(M_{c^r,rnd_i}, 0.1\right), otherwise \end{cases}, \tag{9}$$

$$F_i = rndc_i(\left(M_{F,rnd_i}, 0.1\right). \tag{10}$$

In any case, if the value of c^r_i goes outside [0,1], replaced by the value 0 or 1, which is near the generated value. If the value of $F_i > 1$, it is transformed to 1, and when $F_i \le 0$, eqn. (x) is executed repeatedly to generate a legal value. The parameter p, which is used to adjust the greediness of the *current-to-best/1* mutation strategy, is set for each solution in the population using the equation

$$p^{\mathrm{i}} = rnd\left[p^{\min}, 0.2\right]. \tag{11}$$

2.5 Improving the Search Performance of SHADE Using Linear Population Size Reduction (LSHADE) [36]

L-SHADE is the extended version of SHADE with Linear Population Size Reduction (LPSR). It is a simple deterministic population resizing method that continuously reduces population size according to a linear function. LPSR method is a simplified, particular case of SVPS, which reduces the population linearly as a function of the number of fitness evaluations and requires only one parameter (initial population sizes). In LSHADE, minimum size of the population is defined in advance. Reduction of population increases the algorithm's convergence speed and decreases the computational complexity. Excluding the population reduction process, the search steps of LSHADE are precisely the same as SHADE. Population reduction is accomplished in the algorithm using the formula

$$N_{p,g+1} = \left[\mathrm{round}\left(\frac{N^{\min}_p - N^{\mathrm{initl}}_p}{maxnfes}\right) * nfe + N^{\mathrm{initl}}_p\right]. \tag{12}$$

Whenever $N_{p,g+1} < N_{p,g}$, then $N_{p,g} - N_{p,g+1}$; the number of population is deducted from the population.

2.6 An Ensemble Sinusoidal Parameter Adaptation Incorporated with L-SHADE (LSHADE-EpSin) [37]

In the LSHADE-EpSin approach, a different parameter adaptation technique is used to select control parameters to perform better than the L-SHADE algorithm. Like LSHADE, it also uses $DE/current - to - pbest/1$ mutation strategy. The proposed algorithm uses a new ensemble sinusoidal approach to adapt the DE algorithm's scaling factor automatically. This ensemble technique combines two sinusoidal formulas: (i) a non-adaptive sinusoidal decreasing adjustment and (ii) an adaptive history-based sinusoidal increasing adjustment. These two strategies were used to generate scaling factor for each solution of the population during the first half of the iteration process using the following equations:

$$F^i_g = \frac{1}{2} * \left(\sin(2\pi * \mathrm{freqn} * g + \pi) * \frac{g_{\max} - g}{g_{\max}} + 1\right) \tag{13}$$

$$F^i_g = \frac{1}{2} * \left(\sin\left(2\pi * \mathrm{freqn}_{i,g} * g + \pi\right) * \frac{g}{g_{\max}} + 1\right); \tag{14}$$

freqn in Eq. (13) represents the frequency of sinusoidal function and $\mathrm{freqn_i}$ in Eq. (14) is an adaptive frequency estimated using an adaptive scheme. At every generation $\mathrm{freqn}_{i,g}$ is calculated using a Cauchy distribution. During the second half of the search process, F & c^r is estimated using Cauchy and normal distributions, respectively. This sinusoidal ensemble approach aims to find an effective balance between exploiting the already found best solutions and exploring non-visited regions. The Gaussian walks-based local search method is used to increase the exploitation ability of the process. In this method, a self-adaptive system is used to adapt the control settings during the search. A new ensemble of adaptive version sinusoidal techniques is used to adjust the scaling factor values automatically.

2.7 Ensemble Sinusoidal Differential Covariance Matrix Adaptation with Euclidean Neighborhood (LSHADE-cnEpSin) [38]

This algorithm is an enhanced version of LSHADE-EpSin. The modification is based on two main changes, like an ensemble of sinusoidal approaches based on performance adaptation and covariance matrix learning for the crossover operator. Non-adaptive sinusoidal decreasing adjustment and an adaptive sinusoidal increasing adjustment are used to adapt the scaling factor. In this method, a performance adaptation mechanism based on previous success selects the sinusoidal waves. To set up a satisfactory coordinate system and improve LSHADE-EpSin, covariance matrix learning with the Euclidean neighborhood is used for the crossover operator. This will tackle issues with high correlation among the variables. In this technique, the individuals are first sorted according to their function values, and the best individual, P, is marked. After that, the Euclidean distance is calculated between P and every other individual in the population. Finally, the individuals are sorted based on their Euclidean distance.

2.8 LSHADE with Semi-parameter Adaptation Hybrid with CMA-ES (LSHADE-SPACMA) [39]

It is a hybrid variant of LSHADE-SPA and modified CMA-ES. LSHADE-SPA was proposed with a semi-parameter adaptation scheme to generate scaling factor. Two different strategies were adopted for estimation of F & c^r. During the first half of the iteration process, F & c^r are evaluated using the formulas given below

$$F^i = 0.45 + 0.1 * rnd \tag{15}$$

$$C^{r(i)} = rndn\big(M_c r(i), 0.1\big); \tag{16}$$

$M_c r(i)$ is the position selected arbitrarily from the pool of successful mean values. rnd, here, is a random number between (0,1). During the second half of the search process, the estimation process of C^r does not change, and scaling factor is selected using Cauchy distribution as per the following equation:

$$F^i = rndc\big(M_{F^i}, \partial\big), \tag{17}$$

∂ here denotes the standard deviation. After every iteration, Lehmer mean is evaluated using successful F^i values and one position of M_F is updated. LSHADE-SPA and modified CMA-ES are hybridized and named LSHADE-SPACMA. The modified CMA-ES goes through the crossover operation to enhance the exploration capability. In this approach, both techniques work parallelly on the same population, and more solutions from the population will be allocated slowly to the algorithm with better performance.

2.9 Enhanced LSHADE-SPACMA Algorithm (ELSHADE-SPACMA) [40]

An Enhanced LSHADE-SPACMA technique is the improved variant of LSHADE-SPACMA. In this approach, the p value of the $DE/current-to-pbest/1$ strategy responsible for the greediness of the mutation strategy is made dynamic. The larger value of p enhances the exploration, and smaller values will enhance the exploitation. p is calculated using the following equation:

$$p_{g+1} = \left[\text{round}\left(\frac{p_{\min} - p_{\text{initl}}}{maxnfes}\right) * nfe + p_{\text{initl}}\right]; \tag{18}$$

p_{initl} and $p_{\min}$ symbolizes the initial and minimum value of the variable. Again, a directed mutation strategy has been added within the hybridization framework to improve the performance. Dual enhancement is done for better performance improvement of LSHADE-SPACMA. First, 50% of each LSHADE-SPACMA and AGDE algorithm is integrated into a hybridized framework. In this process, the population is shared alternatively among the algorithms. The second adjustment is the evaluation of p according to the behavior of ELSHADE-SPACMA. Value of p starts with a bigger value to increase exploration and reduces linearly later on. This technique improves the explorative capability at the end of the search.

3 Experimental Results and Discussion on IEEE CEC 2019 and CEC 2020 Functions

The selected algorithms are first tested with IEEE CEC 2019 [41] function suite. The suite contains ten complicated multimodal functions. Then, the IEEE CEC 2020 [42] bound-constrained functions are evaluated. This function set contains unimodal, basic, hybrid, and composite categories functions. In this paper, the function set is assessed using dimension 20. Appendix-I represents the function sets used for the comparison. The termination criteria are set as D*10,000 function evaluations for all the comparison algorithms during the IEEE function evaluation. The parameters of all the chosen algorithms were kept the same as advised in the respective original study. The results are collected as the average value of 30 independent runs. The experiment's device configuration consists of MATLAB version R2015a with an Intel i3 processor, 8 GB DDR-4 RAM, and operating system Windows10.

3.1 Comparison Using IEEE CEC 2019 Functions

DE and its variants were used to examine the results, as shown in Table 2. The table shows the average (avg), standard deviation (sd), and best (bst) values calculated by the algorithms for each of the functions. "NA" represents a function that an algorithm cannot evaluate. An equivalent value denoted by the symbol ≈. SHADE, LSHADE-SPACMA, LSHADE-cnEpSin, and ELSHADE-SPACMA outperform comparison algorithms on 3, 1, 1, and 3 functions, respectively, according to Table 2. SHADE calculates the minimal optimal value on F2, F3, and F5. LSHADE-SPACMA and LSHADE-cnEpSin determine the ideal value for functions F4 and F9. ELSHADE-SPACMA generates the minimum value for functions F7, F8, and F10. The function F1 cannot be evaluated using the LSHADE-cnEpSin algorithm. On functions F1 and F6, just a few algorithms produce identical optimal values. SHADE and ELSHADE-SPACMA were created as the best-performing algorithms. DE and jDE appeared as the worst algorithms, since they generated maximum value on four functions.

3.2 Comparison Using IEEE CEC 2020 Functions

DE and its variants were used to examine the results, as shown in Table 3. The same optimal value evaluated by many algorithms is an equivalent value denoted by the sign ≈. Table 3 show that DE, SHADE, LSHADE-EpSin, LSHADE-SPACMA, and LSHADE-cnEpSin outperform comparison algorithms on the 2, 3, 1, 1, and 1 function, respectively. DE determines the best value for the functions F15 and F20. SHADE calculates the minimal optimal value on F12, F14, and F19. LSHADE-EpSin, LSHADE-SPACMA, and LSHADE-cnEpSin, respectively, measure the best value for functions F13, F16, and F17. On functions F11 and F18, just a few algorithms produce identical optimal values. SHADE has been identified as the algorithm with the most significant results. On CEC 2020 functions, DE and LSHADE-EpSin appeared as the worst algorithms, since both of them generated maximum value on three functions.

4 Description of Real-World Problems Employed and Discussion on the Results

Total six engineering design problems are solved using the algorithms employed. Among the issues selected, one is unconstrained, and the rest are constrained. The termination criteria are set as D*10,000 function evaluations for all the comparison algorithms during the evaluation. The parameters of all the chosen algorithms were kept the same as advised in the respective original study. While solving constrained problems, the death penalty method [43] is used.

This simple and popular strategy simply rejects the population's unfeasible answers. There will never be any unfeasible solutions in the population in this circumstance. According to this process, if any constraint is violated, a penalty value is assigned, and later sum of all the penalty values is added or subtracted with the objective function. This strategy should perform well if a possible search space is convex or a reasonable portion of the entire search space. However, when the problem is extremely limited, the algorithm will waste a lot of time identifying a small number of viable solutions. Furthermore, just evaluating points in the viable portion of the search space precludes superior solutions from being found.

4.1 Parameter Estimation for Frequency-Modulated Sound Waves Problem

This problem is collected from CEC 2011 real-world optimization problem suit [44]. Description of the problem and analysis of the results assessed are given below:

4.1.1 Problem Definition

The frequency-modulated (FM) sound system is necessary for the present-day musical frameworks. The goal here is to produce a sound like the perfect sound naturally, and the highlights are extricated utilizing dissimilarity of highlights among integrated and goal sounds. This cycle of highlight extraction is followed, except if both integrated and target sounds become comparative. The issue is exceptionally convoluted and with multi-modular capacity having epistasis. The optimal value of objective function $f(X^*) = 0$. A considerable number of researchers have already solved the problem. The mathematical expression for assessed sound $y(t)$ and wanted sound $y_0(t)$ waves are given by

$$y(t) = a_1.\sin(\omega_1.t.\theta + a_2.\sin\left(\omega_2.t.\theta + a_3.\sin\left(\omega_3.t.\theta\right)\right)) \quad (19)$$

$$y_0(t) = (1.0).\sin(5.0.t.\theta - 1.5.\sin(4.8.t.\theta + 2.0.\sin(4.9.t.\theta))). \quad (20)$$

Table 2 Comparison of evaluated results on IEEE CEC 2019 function suite

Function-id		DE	jDE	SaDE	SHADE	LSHADE	LSHADE-EpSin	LSHADE-SPACMA	LSHADE-cnEpSin	ELSHADE-SPACMA
F1	avg	1	1.03	1.58	**1≈**	**1≈**	**1≈**	1.02e+09	NA	**1≈**
	sd	3.03e−03	8.82e−02	1.78	0	0	0	8.43e+08	NA	0
	bst	1	1	1	1	1	1	8.77e+01	NA	1
F2	avg	9.53	4.16e+01	3.28e+01	**4**	7.26	5	3.22e+04	6.38	1.33e+01
	sd	9.09	4.16e+01	3.72e+01	1.6	6.18	0	6.53e+03	3.84	1.47e+01
	bst	2.02	2.81	3.02	1.38	2.5	5	1.61e+04	2.62	1.97
F3	avg	1.92	1.53	1.3	**1.19**	1.24	1.32	1.3	1.34	1.31
	sd	1.55	5.29e−01	1.84e−01	1.98e−01	2.02e−01	1.73e−01	1.84e−01	1.55e−01	1.75e−01
	bst	1	1	1	1	1	1	1	1	1
F4	avg	7.26	3.91	4.46	3.89	4.52	3.42	**3.39**	3.72	4.45
	sd	3.66	1.97	1.94	9.90e−01	1.07	9.67e−01	8.52e−01	1.25	2.02
	bst	3.98	1	2	2	2	1	1	1.99	2
F5	avg	1.03	1.02	1.01	1.01	1.01	**1.01**	1.01	1.01	1.03
	sd	3.03e−02	1.45e−02	1.08e−02	6.14e−03	1.27e−02	1.02e−02	7.49e−03	1.04e−02	2.66e−02
	bst	1	1	1	1	1	1	1	1	1
F6	avg	**1≈**	1.38	**1≈**	**1≈**	1.02	1	1	1.02	**1≈**
	sd	0	4.58e−01	0	0	8.51e−02	6.42e−07	2.64e−03	8.51e−02	0
	bst	1	1	1	1	1	1	1	1	1
F7	avg	4.93e+02	2.11e+02	2.60e+02	1.37e+02	1.74e+02	1.72e+02	1.47e+02	1.44e+02	**1.19e+02**
	sd	4.17e+02	1.60e+02	1.39e+02	9.67e+01	1.14e+02	1.04e+02	1.19e+02	1.15e+02	9.80e+01
	bst	1.19	1.36	4.76	4.81	5.15	1.23	1.46	1.19	1.31
F8	avg	2.97	3.33	2.98	2.72	2.82	2.37	2.38	2.35	**2.35**
	sd	6.60e−01	2.46e−01	3.42e−01	2.87e−01	3.75e−01	4.44e−01	3.50e−01	4.85e−01	4.31e−01
	bst	1.49	2.69	2	2	1.85	1.32	1.73	1.39	1.15
F9	avg	1.10	1.13	1.13	1.09	1.08	1.05	1.04	**1.04**	1.04
	sd	4.60e−02	2.39e−02	2.49e−02	1.42e−02	2.64e−02	2.22e−02	1.82e−02	1.50e−02	2.36e−02
	bst	1.01	1.07	1.08	1.06	1.02	1.01	1.01	1.01	1.01
F10	avg	1.64e+01	2.10e+01	1.50e+01	1.46e+01	1.73e+01	1.44e+01	1.32e+01	1.46e+01	**1.04e+01**
	sd	8.38	5.34e−01	9.26	8.75	7.51	8.82	9.69	9.16	8.38
	bst	1	1.82e+01	1	1.25	1.38	1.09	1	1	1

Bold numbers in the table indicate superior values

Table 3 Comparison of evaluated results on IEEE CEC 2020 function suite

Function-id		DE	jDE	SaDE	SHADE	LSHADE	LSHADE-EpSin	LSHADE-SPACMA	LSHADE-cnEpSin	ELSHADE-SPACMA
F11	avg	100	100	100	100	100	8.73e+02	**100≈**	100	**100≈**
	sd	2.64e−15	2.64e−15	1.17e−09	9.15e−15	1.18e−14	1.35e+03	0	4.51e−14	0
	bst	100	100	100	100	100	1.01e+02	100	100	100
F12	avg	2.36e+03	1.55e+03	1.28e+03	**1.12e+03**	1.14e+03	1.21e+03	1.17e+03	1.23e+03	1.13e+03
	sd	1.24e+03	1.49e+02	1.12e+02	3.17e+01	5.33e+01	7.93e+01	7.28e+01	7.02e+01	4.87e+01
	bst	1.11e+03	1.30e+03	1.11e+03	1.10e+03	1.10e+03	1.11e+03	1.10e+03	1.11e+03	1.10e+03
F13	avg	7.35e+02	7.30e+02	7.23e+02	7.23e+02	7.23e+02	**7.08e+02**	7.24e+02	7.26e+02	7.23e+02
	sd	1.52e+01	3.05	1.08	7.93e−01	9.49e−01	2.75	2.32	5.75	8.13e−01
	bst	7.24e+02	7.23e+02	7.21e+02	7.21e+02	7.21e+02	7.02e+02	7.21e+02	7.10e+02	7.22e+02
F14	avg	1.90e+03	1.90e+03	1.90e+03	**1.90e+03**	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03
	sd	2.41	2.89e−01	6.31e−01	1.10e−01	2.55e−01	4.12e−01	2.01e−01	1.70e−01	7.56e−01
	bst	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03
F15	avg	**1.79e+03**	2.10e+03	2.64e+03	2.19e+03	2.34e+03	1.92e+05	2.55e+03	1.93e+03	2.23e+03
	sd	8.98e+01	2.87e+02	1.37e+03	2.10e+02	2.99e+02	1.01e+05	3.13e+02	100	2.37e+02
	bst	1.70e+03	1.72e+02	1.78e+03	1.78e+03	1.92e+03	3.73e+04	2.18e+03	1.75e+03	1.86e+03
F16	avg	1.74e+03	1.88e+03	1.63e+03	1.68e+03	1.68e+03	1.68e+03	**1.60e+03**	2.05e+03	1.62e+03
	sd	9.25e−13	4.63e−13	4.63e−13	4.63e−13	1.16e−12	6.94e−13	4.63e−13	9.25e−13	6.94e−13
	bst	1.74e+03	1.88e+03	1.63e+03	1.68e+03	1.68e+03	1.68e+03	1.60e+03	2.05e+03	1.62e+03
F17	avg	2.14e+03	2.16e+03	2.23e+03	2.27e+03	2.28e+03	1.07e+05	2.38e+03	**2.14e+03**	2.28e+03
	sd	5.12e+01	6.93e+01	9.72e+01	9.22e+01	1.05e+02	6.56e+04	2.00e+02	3.44e+01	1.20e+02
	bst	2.10e+03	2.10e+03	2.11e+03	2.10e+03	2.10e+03	1.59e+04	2.14e+03	2.10e+03	2.11e+03
F18	avg	2.30e+03	2.30e+03	2.30e+03	**2.30e+03≈**	2.35e+03	2.30e+03	2.30e+03	2.30e+03	**2.30e+03≈**
	sd	2.06e−01	8.44e−14	3.43e−01	0	2.63e+02	1.31e−12	6.81e−01	6.49e−13	0
	bst	2.30e+03	2.30e+03	2.30e+03	2.30e+03	2.30e+03	2.30e+03	2.30e+03	2.30e+03	2.30e+03
F19	avg	2.81e+03	2.82e+03	2.81e+03	**2.81e+03**	2.81e+03	2.84e+03	2.81e+03	2.83e+03	2.82e+03
	sd	1.21e+01	5.03	7.93	3.43	4.25	7.81	5.06	6.36e+03	8.14
	bst	2.76e+03	2.81e+03	2.80e+03	2.80e+03	2.80e+03	2.82e+03	2.80e+03	2.50e+03	2.81e+03
F20	avg	**2.91e+03**	2.91e+03	2.94e+03	2.92e+03	2.91e+03	2.91e+03	2.92e+03	**2.91e+03**	2.91e+03
	sd	3.67e−02	1.69e+01	2.69e+01	1.29e+01	2.27	9.15e−01	1.58e+01	7.63e−02	8.91e−01
	bst	2.91e+03	2.91e+03	2.91e+03	2.91e+03	2.91e+03	2.91e+03	2.91e+03	2.91e+03	2.91e+03

Bold numbers in the table indicate superior values

In this problem, $\theta = \frac{2\Pi}{100}$ and the parameters lie within the range [− 6.4, 6.35]. The fitness function is defined as the summation of square errors between the assessed sound and the wanted sound. The objective function of the problem is defined as

$$f(X) = \sum_{t=0}^{100} \left(y(t) - y_0(t)\right)^2. \tag{21}$$

A diagram of the problem is given in Fig. 2a.

4.1.2 Analysis of Result

Table 4 shows the results of the algorithms used to calculate them. Table 4 reveals that most of the algorithms can determine the best ideal value. DE identifies the problem's minimal average, standard deviation, and best among all the comparison algorithms. The other two algorithms on the list, SaDE, and LSHADE, are ranked second and third. LSHADE-SPACMA calculates the problem's worst mean value.

4.2 Car Side Impact Design Problem

This problem was initially proposed by Gu et al. [45]. A diagram of the problem is given in Fig. 2b. Description of the problem and discussion on the evaluated results are given below:

4.2.1 Problem Definition

The car is exposed to a side impact on the foundation of the European Enhanced Vehicle Safety Committee (EEVC) procedures. The objective is to minimize the car's total weight using eleven mixed variables while maintaining safety performance according to the standard. These variables represent the thickness and material of critical parts of the vehicle. The 8th and 9th variables are discrete, and these are material design variables, while the rest are continuous and represent thickness design variables.

The symbols $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}$ are used here to represent the variables thickness of B-pillar inner, the thickness of B-pillar reinforcement, the thickness of floor side inner, the thickness of cross members, the thickness of door beam, the thickness of door beltline reinforcement, the thickness of roof rail, the material of B-pillar inner, the material of floor side inner, barrier height, and barrier hitting position, respectively. The problem is subjected to ten inequality constraints. The car side impact design is considered a confirmed case of a mechanical optimization problem with mixed discrete and continuous design variables. This problem can be mathematically described as

$$\boldsymbol{a} = \left\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\right\}. \tag{22}$$

Objective function

$$\begin{aligned} Minf(\boldsymbol{a}) = 1.98 + 4.90a_1 + 6.67a_2 + 6.98a_3 \\ + 4.01a_4 + 1.78a_5 + 2.73a_7, \end{aligned} \tag{23}$$

subject to:

$$\begin{aligned} h_1(\boldsymbol{a}) = 1.16 - 0.3717a_2a_4 - 0.00931a_2a_{10} \\ - 0.484a_3a_9 + 0.01343a_6a_{10} \le 1 \end{aligned} \tag{24}$$

$$\begin{aligned} h_2(\boldsymbol{a}) = 0.261 - 0.0159a_1a_2 - 0.188a_1a_8 - 0.019a_2a_7 \\ + 0.0144a_3a_5 + 0.0008757a_5a_{10} + 0.080405a_6a_9 \\ + 0.00139a_8a_{11} + 0.00001575a_{10}a_{11} \le 0.32, \end{aligned} \tag{25}$$

$$\begin{aligned} h_3(\boldsymbol{a}) = 0.214 + 0.00817a_5 - 0.131a_1a_8 \\ - 0.0704a_1a_9 + 0.03099a_2a_6 - \end{aligned}$$

$$\begin{aligned} 0.018a_2a_7 + 0.0208a_3a_8 + 0.121a_3a_9 \\ - 0.00364a_5a_6 + 0.0007715a_5a_{10} \\ - 0.0005354a_6a_{10} + 0.00121a_8a_{11} \le 0.32, \end{aligned} \tag{26}$$

$$h_4(\boldsymbol{a}) = 0.074 - 0.061a_2 - 0.163a_3a_8 + 0.001232a_3a_{10} - 0.166a_7a_9$$

$$+0.227a_2^2 \le 0.32, \tag{27}$$

$$\begin{aligned} h_5(\boldsymbol{a}) = 28.98 + 3.818a_3 - 4.2a_1a_2 + 0.0207a_5a_{10} \\ + 6.63a_6a_9 - 7.7a_7a_8 + 0.32a_9a_{10} \le 32, \end{aligned} \tag{28}$$

$$\begin{aligned} h_6(\boldsymbol{a}) = 33.86 + 2.95a_3 + 0.1792a_{10} - 5.05a_1a_2 \\ - 11.0a_2a_8 - 0.0215a_5a_{10} - 9.98a_7a_8 \\ + 22.0a_8a_9 \le 32, \end{aligned} \tag{29}$$

$$h_7(\boldsymbol{a}) = 46.36 - 9.9a_2 - 12.9a_1a_8 + 0.1107a_3a_{10} \le 32, \tag{30}$$

$$\begin{aligned} h_8(\boldsymbol{a}) = 4.72 - 0.5a_4 - 0.19a_2a_3 - 0.0122a_4a_{10} \\ + 0.009325a_6a_{10} + 0.000191a_{11}^2 \le 4, \end{aligned} \tag{31}$$

$$\begin{aligned} h_9(\boldsymbol{a}) = 10.58 - 0.674a_1a_2 - 1.95a_2a_8 + 0.02054a_3a_{10} \\ - 0.0198a_4a_{10} + 0.028a_6a_{10} \le 9.9, \end{aligned} \tag{32}$$

$$\begin{aligned} h_{10}(\boldsymbol{a}) = 16.45 - 0.489a_3a_7 - 0.843a_5a_6 + 0.0432a_9a_{10} \\ - 0.0556a_9a_{11} - 0.000786a_{11}^2 \le 15.7, \end{aligned} \tag{33}$$

where

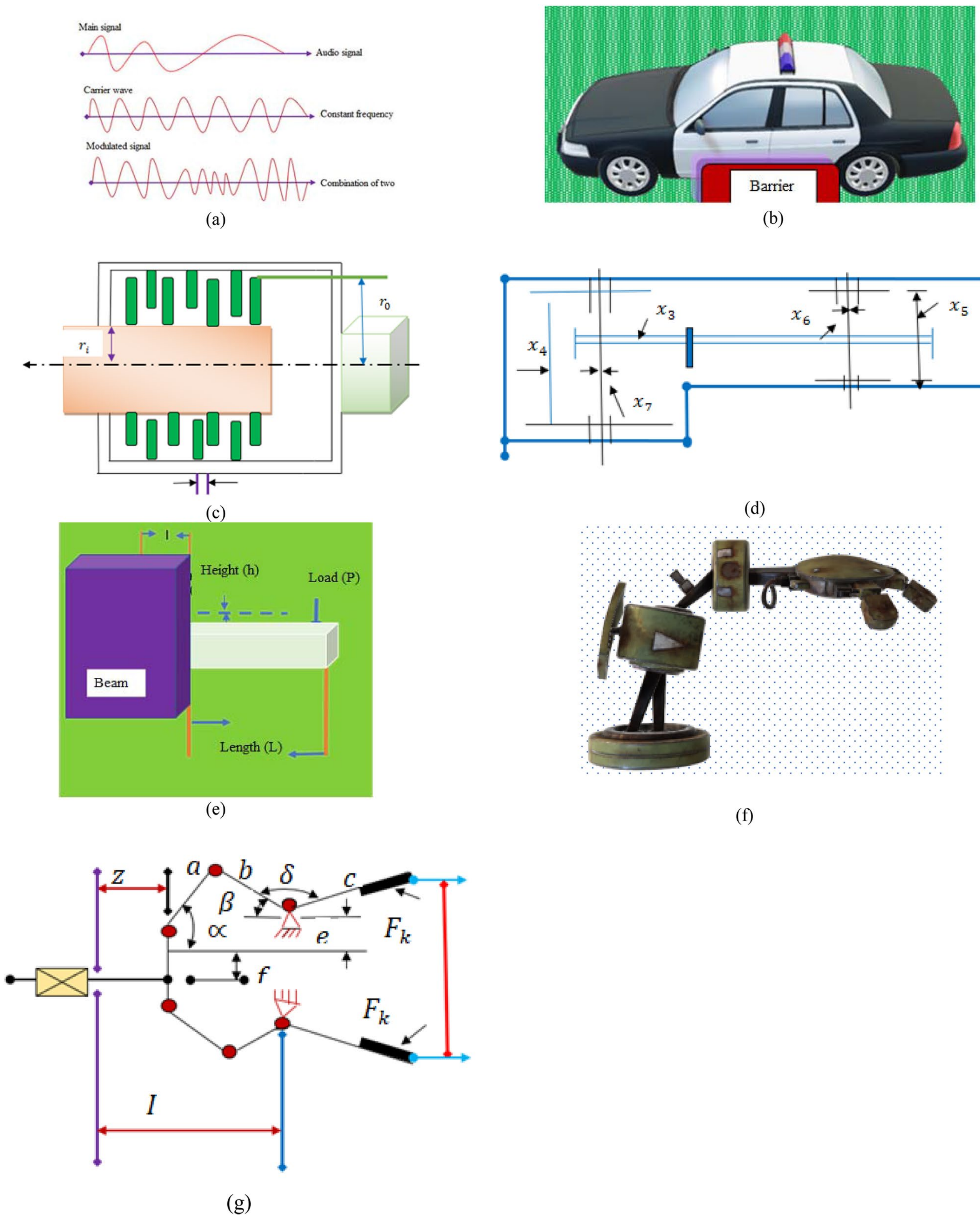

Fig. 2 Diagram of the real-world problems

Table 4 Comparison of results on the parameter estimation for frequency-modulated sound waves problem

Algorithms	Mean	Sd	Best	Evaluation time (seconds)	Optimal value
DE	**0**	**0**	**0**	1.8978e+02	0
jDE	9.37021e−01	2.51343	7.11378e−17	2.2937e+02	
SaDE	1.36053e−03	6.46447e−03	**0**	4.1959e+02	
SHADE	2.39056	3.22852	**0**	**1.7953e+02**	
LSHADE	1.18685	2.48267	**0**	2.1719e+02	
LSHADE-EpSin	3.90314	4.61850	9.45145e−04	2.2417e+02	
LSHADE-SPACMA	1.35825e+01	5.77936	**0**	2.3020e+02	
LSHADE-cnEpSin	8.58850	4.64689	1.40708	2.4162e+02	
ELSHADE-SPACMA	4.90410	5.64013	**0**	2.2765e+02	

Bold numbers in the table indicate superior values

Table 5 Comparison of results on the car side impact design problem

Algorithms	Mean	Sd	Best	Evaluation time (seconds)	Optimal value
DE	2.28429e+01	**5.35960e−15**	2.28429e+01	4.2466e+02	2.189634e+01
jDE	2.28429e+01	1.94790e−10	2.28429e+01	3.6223e+02	
SaDE	2.28429e+01	3.91906e−08	2.28429e+01	7.3100e+02	
SHADE	2.28429e+01	7.52199e−15	2.28429e+01	**2.7731e+02**	
LSHADE	2.28429e+01	6.11801e−15	2.28429e+01	3.4284e+02	
LSHADE-EpSin	**2.18963e+01**	2.84921	**1.55150e+01**	3.8264e+02	
LSHADE-SPACMA	2.28429e+01	6.29335e−15	2.28429e−15	4.0453e+02	
LSHADE-cnEpSin	2.26907e+01	1.61958	1.65571e+01	3.8240e+02	
ELSHADE-SPACMA	2.28429e+01	6.11801e−15	2.28429e+01	3.4728e+02	

Bold numbers in the table indicate superior values

$$0.5 \le a_i \le 1.5, i = 1, 2, 3, 4, 5, 6, 7, a_8, a_9 \in (0.192, 0.345),$$

$$-30 \le a_{10}, a_{11} \le 30.$$

4.2.2 Analysis of Results

Table 5 shows the results of the comparison algorithms' calculations. Table 5 reveals that LSHADE-EpSin is capable of finding the smallest best value. On the problem, LSHADE-cnEpSin determines the second minimum optimal value. LSHADE-EpSin determines the issue's minimal average value and the best value among all the comparison methods. Other algorithms, however, found a lower standard deviation value. Different algorithms rated 2nd and 3rd are DE and LSHADE and ELSHADE-SPACMA. SaDE calculates the problem's worst mean value.

4.3 Multiple Disk Clutch Brake Design Problem

It comes from the category of a mechanical engineering design problem. Figure 2c represents the diagram of the problem. The problem is described in [29]. Details of the problem and analysis of the results assessed by different algorithms are given below:

4.3.1 Problem Definition

The minimization of the mass of multiple disk clutch brake is the prime objective of the problem. This can be achieved by optimizing five decision variables, namely inner radius (x_1), outer radius (x_2), disk thickness (x_3), the force of actuators (x_4), and the number of frictional surfaces (x_5). It is a constrained problem and contains eight nonlinear constraints. The mathematical formulation of the problem is given below.

Objective function

$$\text{Min} f(x) = \pi\left(x_2^2 - x_1^2\right)x_3\left(x_5 + 1\right)\rho. \tag{34}$$

Constraints

$$h_1(x) = -p_{\max} + p_{rz} \le 0, \tag{35}$$

$$h_2(x) = p_{rz}V_{sr} - V_{sr,\max}p_{\max} \le 0, \tag{36}$$

$$h_3(x) = \Delta R + x_1 - x_2 \le 0, \tag{37}$$

$$h_4(x) = -L_{\max} + (x_5 + 1)\left(x_3 + \partial\right) \le 0, \tag{38}$$

$$h_5(x) = sM_s - M_h \le 0, \tag{39}$$

$$h_6(x) = T \ge 0, \tag{40}$$

$$h_7(x) = -V_{sr,\max} + V_{sr} \le 0, \tag{41}$$

$$h_8(x) = T - T_{\max} \le 0, \tag{42}$$

where

$$M_h = \frac{2}{3}\mu x_4 x_5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2},$$

$$\omega = \frac{\Pi n}{30}\frac{\text{rad}}{s},$$

$$A = \pi(x_2^2 - x_1^2),$$
$$p_{rz} = \frac{x_4}{A},$$
$$V_{sr} = \frac{\Pi R_{sr} n}{30},$$

$$R_{sr} = \frac{2}{3}\frac{x_2^3 - x_1^3}{x_2^2 x_1^2},$$

$$T = \frac{I_z \omega}{M_h + M_f},$$

$$\Delta R = 20, L_{\max} = 30, \mu = 0.6,$$

$$V_{sr,\max} = 10, \partial = 0.5, s = 1.5,$$

$$T_{\max} = 15, n = 250, I_z = 55$$

$$M_s = 40, M_f = 3, \text{and} p_{\max} = 1.$$

Variable range

$$60 \le x_1 \le 80, 90 \le x_2 \le 110, 1 \le x_3 \le 3, 0 \le x_4 \le 1000, 2 \le x_5 \le 9.$$

4.3.2 Analysis of Result

Table 6 shows the results provided by the comparison algorithms. Table 6 shows that, except for LSHADE-SPACMA and LSHADE-cnEpSin, all algorithms discover the same value as average ideal value, standard deviation value, and best value. On this problem, LSHADE-cnEpSin is the worst performer of the compared algorithms.

4.4 Weight Minimization of a Speed Reducer Problem

The issue is related to the Mechanical Engineering discipline. A diagram of the problem is given in Fig. 2d.

4.4.1 Problem Definition

The notched is used as an independent element to reduce or increase the speed, and a firm covering is used to enclose them. When this unit is used to minimize any device's speed, it is called a speed reducer. Reducer is widely used in turbines, rolling mills to reduce speed. The parameters required to optimize this problem are the gear's *b*-face width, *z*-number of pinning teeth, m-teeth module, l_1-length of the shaft-1

Table 6 Comparison of results on the multiple disk clutch brake design problem

Algorithms	Mean	Sd	Best	Evaluation time (seconds)	Optimal value
DE	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	1.57862e+02	2.352424e–01
jDE	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	2.02500e+02	
SaDE	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	3.69844e+02	
SHADE	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	**1.53062e+02**	
LSHADE	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	1.84641e+02	
LSHADE-EpSin	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	1.91125e+02	
LSHADE-SPACMA	2.42347e–01	2.76493e–02	2.35242e–01	1.92797e+02	
LSHADE-cnEpSin	4.84572e–01	1.05304e–01	3.11188e–01	2.00047e+02	
ELSHADE-SPACMA	**2.35242e–01**	**1.41150e–16**	**2.35242e–01**	1.96078e+02	

Bold numbers in the table indicate superior values

between bearing, l_2-length of the shaft-2 between bearing, d_1- diameter of shaft-1, and d_2- diameter of shaft-2. Here, the problem's design variables are represented by the position vectors of the algorithm in the following manner:

Face width of the gear $(b) = x_1$

Teeth module $(m) = x_2$

Number of pinning teeth $(z) = x_3$

Length of shaft-1 between bearing $(l_1)_= x_4$

Length of shaft-2 between bearing $(l_2)_= x_5$

Diameter of shaft-1 $(d_1) = x_6$

Diameter of shaft-1 $(d_2) = x_7$.

The problem is found in [29]. The mathematical representation of the problem is

Objective function

$$\begin{aligned}\text{Min.}f(x) =&0.7854x_2^2x_1\left(14.9334x_3 + 3.3333x_3^2 - 43.0934\right)\\&+ 0.7854\left(x_4x_6^2 + x_5x_7^2\right) + 7.477\left(x_7^3 + x_6^3\right)\\&- 1.508\left(x_7^2 + x_6^2\right)\end{aligned} \tag{43}$$

$$h_1(x) = x_1x_2^2x_3 + 27 \le 0, \tag{44}$$

$$h_2(x) = -(1)x_2^2x_3^2 + 397.5 \le 0, \tag{45}$$

$$h_3(x) = -x_2x_2^4x_3x_4^{-3} + 1.93 \le 0, \tag{46}$$

$$h_4(x) = -x_2x_7^4x_3x_5^{-3} + 1.93 \le 0, \tag{47}$$

$$h_5(x) = 10x_6^{-3}\sqrt{16\cdot 91\times 10^6 + \left(745x_4x_2^{-1}x_3^{-1}\right)^2} - 1100 \le 0, \tag{48}$$

$$h_6(x) = 10x_7^{-3}\sqrt{157.5\times 10^6 + \left(745x_5x_2^{-1}x_3^{-1}\right)^2} - 850 \le 0, \tag{49}$$

$$h_7(x) = x_2x_3 - 40 \le 0, \tag{50}$$

$$h_8(x) = -x_1x_2^{-1} + 5 \le 0, \tag{51}$$

$$h_9(x) = x_1x_2^{-1} - 12 \le 0, \tag{52}$$

$$h_{10}(x) = 1.5x_6 - x_4 + 1.9 \le 0, \tag{53}$$

$$h_{11}(x) = 1.1x_7 - x_5 + 1.9 \le 0. \tag{54}$$

Variable range

$$2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28,$$

$$2.9 \le x_6 \le 3.9, 5 \le x_7 \le 5.5, 7.3 \le x_4, x_5 \le 8.3.$$

4.4.2 Analysis of Result

Table 7 shows the estimated results from the comparison algorithms. Examining Table 7 shows that all algorithms, except LSHADE-cnEpSin and LSHADE-EPSin, find the same value as average ideal value, standard deviation value, and best value as average optimal value, standard deviation value, and best value. LSHADE-cnEpSin can determine the optimal value, which is lower than the value found in the literature. The average and best values determined by LSHADE-cnEpSin are lower than those determined by the other algorithms used in this study. LSHADE-EpSin is the worst contrast algorithms in terms of average and standard deviation values.

4.5 Welded Beam Design Problem

This is a design problem from the mechanical engineering discipline. Figure 2e shows a diagram of the problem.

Table 7 Comparison of results on the weight minimization of a speed reducer problem

Algorithms	Mean	Sd	Best	Evaluation time (seconds)	Optimal value
DE	2.99442e+03	**0**	2.99442e+03	2.12390e+02	2.9944e + 03
jDE	2.99442e+03	**0**	2.99442e+03	2.52094e+02	
SaDE	2.99442e+03	**0**	2.99442e+03	4.93062e+02	
SHADE	2.99442e+03	**0**	2.99442e+03	**1.924844e+02**	
LSHADE	2.99442e+03	**0**	2.99442e+03	2.28797e+02	
LSHADE-EpSin	3.01072e+03	8.92779e+01	2.99442e+03	2.47641e+02	
LSHADE-SPACMA	2.99442e+03	**0**	2.99442e+03	2.53031e+02	
LSHADE-cnEpSin	**2.82398e+03**	1.13335e+01	**2.80147e+03**	2.66656e+02	
ELSHADE-SPACMA	2.99442e+03	**0**	2.99442e+03	2.62484e+02	

Bold numbers in the table indicate superior values

4.5.1 Problem Definition

The minimization of the fabrication cost of the welded beam is the main objective of the problem. The problem comprises five constraints and four variables. Limitations are shear stress (τ), bending stress within the beam (θ), buckling load (P_c), and end deflection of the beam (δ). The variables to be optimized here are the height of the bar (t), the thickness of the weld (h), the thickness of the bar (b), and the length of the clamped bar (l).h, l, t, and b in the problem are represented by x_1, x_2, x_3, and x_4. The problem is described in the Mechanical engineering category of the literature [29]. The mathematical formulation of the problem is.

Objective function

$$\text{Min.f}(x) = 0.04811x_3x_4\{x_2 + 14\} + 1010471x_1^2x_2 \tag{55}$$

subject to.

$h_1(x) = x_1 - x_4 \le 0$, (56)

$$h_2(x) = \delta(x) - \delta_{\max} \le 0, \tag{57}$$

$$h_3(x) = P \le P_c(x), \tag{58}$$

$$h_4(x) = \tau_{\max} \ge \tau(x), \tag{59}$$

$$h_5(x) = \sigma(x) - \sigma_{\max} \le 0, \tag{60}$$

where

$$\text{R} = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \text{J} = 2\frac{x_2^2}{4} + \left(\frac{2}{\left(\left(\frac{x_1+x_3}{2}\right)\right)\sqrt{2}x_1x_2}\right),$$

$$\sigma(x) = \frac{6\text{PL}}{x_4x_3^2}, \delta(x) = \frac{6\text{PL}^3}{\text{E}x_3^2x_4}, \text{P}_c(x) = \frac{4.013\text{E}x_3x_4^3}{6\text{L}^2}\left(1 - \frac{x_3}{2\text{L}}\sqrt{\frac{\text{E}}{4\text{G}}}\right),$$

$$L = 14, P = 6000, E = 30 \times 1^6, G = 12 \times 10^6$$

$$\sigma_{\max} = 30,000, \delta_{\max} = 0.25, \tau_{\max} = 13600,$$

where

$0.1 \le x_3, x_2 \le 10, 0.125 \le x_1 \le 2, 0.1 \le x_4 \le 2.$

4.5.2 Analysis of Result

Results calculated by the comparison algorithms are presented in Table 8. Table 8 reveals that the average value and the best value evaluated by most employed algorithms are the same. LSHADE-cnEpSin is not suitable for this problem. Though other algorithms except LSHADE-EpSin and LSHADE-cnEpSin generate the same average value and best value, their standard deviation value differs. The minimum standard deviation value designates the algorithm's robustness in finding the solution. SaDE, DE, and LSHADE-SPACMA are ranked as 1st, 2nd, and 3rd algorithms based on generated standard deviation values. The performance of LSHADE-EpSin is recorded as the worst on this problem.

4.6 Robot Gripper Problem

This problem was first modeled by Osyczka et al. [46]. It is a design problem from the discipline Mechanical Engineering. Diagram of the problem are given in Fig. 2f, g.

4.6.1 Problem Definition

This problem has seven decision variables denoted by $x = [a, b, c, e, f, I, \delta]$ where, a, b, c, e, f, I, are dimensions of the gripper, and δ is the angle between c and b shown in Fig. 2g. The objective function of the problem is the absolute value of the difference between the maximum and minimum

Table 8 Comparison of results on the welded beam design problem

Algorithms	Mean	Sd	Best	Evaluation time (seconds)	Optimal value
DE	1.67021	1.93398e–16	**1.67021**	1.26031e+02	1.67021e+00
jDE	1.67021	1.10100e–08	**1.67021**	1.53297e+02	
SaDE	**1.67021**	**1.88951e–16**	**1.67021**	2.72844e+02	
SHADE	1.67021	2.25840e–16	**1.67021**	**1.18000e+02**	
LSHADE	1.67021	2.18182e–16	**1.67021**	1.36906e+02	
LSHADE-EpSin	1.69763	1.66363e–02	1.67473	1.54141e+02	
LSHADE-SPACMA	1.67021	1.97744e–16	**1.67021**	1.53266e+02	
LSHADE-cnEpSin	NA	NA	NA	NA	
ELSHADE-SPACMA	1.67021	2.14251e–16	**1.67021**	1.51109e+02	

Bold numbers in the table indicate superior values

force generated by the robot. The mathematical formulation of the robot gripper problem is shown below:

$$f(x) = \left|\max F_K(x,z) - \min F_K(x,z)\right| \tag{61}$$

subject to

$$g_1(x) = y\left(x, z_{\max}\right) - Y_{\min} \le 0 \tag{62}$$

$$g_2(x) = y\left(x, z_{\max}\right) \le 0 \tag{63}$$

$$g_3(x) = Y_{\max} - y(x,0) \le 0 \tag{64}$$

$$g_4(x) = y(x,0) - Y_G \le 0 \tag{65}$$

$$g_5(x) = I^2 + e^2 - (a+b)^2 \le 0 \tag{66}$$

$$g_6(x) = b^2 - (a-e)^2 - \left(I - z_{\max}\right)^2 \le 0 \tag{67}$$

$$g_7(x) = z_{\max} - I \le 0, \tag{68}$$

where

$$\alpha = \mathrm{ArcCos}\left(\frac{g^2 + a^2 - b^2}{g \times 2b}\right) + \varphi; g^2 - e^2 = (z - I)^2; \varphi = \mathrm{ArcTan}\left(\frac{e}{1 - z}\right)$$

$$\beta = \mathrm{ArcCos}\left(\frac{g^2 + b^2 - a^2}{g \times 2b}\right) - \varphi; y(x,z) = 2(c + e + f \times \sin(\delta + \beta))$$

$$F_k = \left(\frac{\sin(\alpha + \beta) \times b \times p}{2c \times \cos\alpha}\right); Y_{\min} = 50; Y_{\max} = 100; Y_G = 150; z_{\max} = 100; p = 100$$

$$10 \le f, a, b \le 150; 0 \le e \le 50; 100 \le l \le 300; 100 \le c \le 200; 1 \le \delta \le 3.14.$$

4.6.2 Analysis of Result

The optimal value of this problem is 2.5287918415. Numerical results evaluated by the algorithms are given in Table 9. LSHADE records the average value close to the optimal value found in the literature. The standard deviation value calculated by SHADE is minimum among the algorithms. LSHADE-EpSin algorithm evaluates the best optimal value. LSHADE-SPACMA is not suitable for this problem. SHADE and ELSHADE-SPACMA are ranked as 2nd and 3rd, respectively, on this problem. LSHADE-cnEpSin is the worst algorithms in terms of results among the compared algorithms.

5 Statistical Analysis

Statistical test of the evaluated results is performed using Friedman's rank test of results on IEEE CEC 2019 and IEEE 2020 functions. The Wilcoxon signed-rank test is executed based on the algorithms evaluating real-world problems employed. Friedman's test and Wilcoxon signed-rank test fall into the non-parametric statistical hypothesis test category and compare two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ (i.e., it is a paired difference test). Tables 10 and 11 illustrate the outcomes of Friedman's test, and Table 12 demonstrates the results of the Wilcoxon signed-rank test. Analysis of the data in Table 10 exposes that SHADE, ELSHADE-SPACMA, and LSHADE-EpSin can be ranked as 1st, 2nd, and 3rd, respectively. Similarly, the analysis of Table 11 reveals that SHADE, LSHADE, and ELSHADE-SPACMA are the algorithms that

Table 9 Comparison of results on the robot gripper problem

Algorithms	Mean	Sd	Best	Evaluation time (seconds)	Optimal value
DE	2.62965	1.56533e–01	2.52681	8.49294e+03	2.52879e+00
jDE	0	0	0	9.75948e+03	
SaDE	2.56782	6.39773e–02	2.52681	9.02325e+03	
SHADE	2.52681	**3.19622e–14**	2.52681	8.69750e+03	
LSHADE	**2.52790**	5.92604e–03	2.52681	8.59072e+03	
LSHADE-EpSin	2.65886	1.62913e–01	**2.52739**	9.20495e+03	
LSHADE-SPACMA	NA	NA	NA	NA	
LSHADE-cnEpSin	2.82731	2.75806e–01	2.14118	8.92586e+03	
ELSHADE-SPACMA	2.52681	3.94040e–14	2.52681	**8.45197e+03**	

Bold numbers in the table indicate superior values

Table 10 Result of Friedman's rank test on IEEE CEC 2019 functions

Algorithm	Mean rank	Rank sum	Final rank	*P* value
DE	6.8	68	8	*P* value (0.000 < 0.01) indicates that Ho is rejected at 1% level of significance. i.e., there is a significant difference between the performances of different methods at a 1% level of significance
jDE	7.7	77	9	
SaDE	5.8	58	7	
SHADE	2.6	26	**1**	
LSHADE	5.3	53	6	
LSHADE-EpSin	3.5	35	**3**	
LSHADE-SPACMA	4.2	42	4	
LSHADE-cnEpSin	4.6	46	5	
ELSHADE-SPACMA	3.2	32	**2**	

Bold numbers in the table indicate superior values

Table 11 Result of Friedman's rank test on IEEE CEC 2020 functions

Algorithm	Mean rank	Rank sum	Final rank	*P* value
DE	4.9	49	5	*P* value (0.000 < 0.01) indicates that Ho is rejected at 1% level of significance. i.e., there is a significant difference between the performances of different methods at a 1% level of significance
jDE	6.1	61	9	
SaDE	5.65	56.5	7	
SHADE	3.5	35	**1**	
LSHADE	4.05	40.5	**2**	
LSHADE-EpSin	5.85	58.5	8	
LSHADE-SPACMA	4.95	49.5	6	
LSHADE-cnEpSin	4.7	47	4	
ELSHADE-SPACMA	4.3	43	**3**	

Bold numbers in the table indicate superior values

can be ranked as 1st, 2nd, and 3rd, respectively. Therefore, the performance of SHADE and ELSHADE-SPACMA are consistent with the IEEE functions employed in this study.

Table 12 displays the results of the Wilcoxon signed-rank test on real-world problems. It can be seen from the results that DE is the best algorithm in terms of the minimum rank obtained by the algorithm. ELSHADE-SPACMA and SaDE are the algorithms ranked as 2nd and 3rd. Evaluating the mean final rank, it can be concluded that SHADE is the best algorithm and ELSHADE-SPACMA is the 2nd best algorithm among the algorithms employed here for comparison.

6 Run-Time Analyses of the Algorithms in Real-World Problems

The total time taken by every algorithm to evaluate the mean, standard deviation, and best value within 30 independent runs on a particular problem is shown as the algorithm's run time. The run time of a problem depends on the algorithm's complexity, the number of dimensions, and the complexity of the problem. The six real-world issues evaluated in this study have 6, 11, 5, 7, 4, and 7 dimensions, respectively. Analyzing the runtimes being assessed in Tables 6, 7, 8 and 9 exposes that SHADE evaluates the first five out of six problems in less run time than other algorithms. ELSHADE-SPACMA optimizes the 6th problem in a minimum run time than the compared algorithms. The second problem (car side impact design) has the maximum dimension among the real-world problems employed. Thus, the evaluation time of this problem is more. Due to the problem complication evaluation time of the last problem (robot gripper) is greater than all the problems.

7 Conclusion

Numerous optimization algorithms are available in the literature. Among them, considering the efficacy of DE, the researchers have modified it extensively. Several variants of DE have won the CEC competitions. These CEC-winning variants are regarded as strong algorithms to solve

Table 12 Result of Wilcoxon signed-rank test on real-world problems

RLP	DE	jDE	SaDE	SHADE	LSHADE	LSHADE-EpSin	LSHADE-SPACMA	LSHADE-cnEpSin	ELSHADE-SPACMA
Frequency modulation	0.00E+00	9.37E–01	1.36E–03	2.39E+00	1.19E+00	3.90E+00	1.36E+01	8.59E+00	4.90E+00
Individual rank	1	3	2	5	4	6	9	8	7
Car Side impact design	2.28E+01	2.28E+01	2.28E+01	2.28E+01	2.28E+01	2.19E+01	2.28E+01	2.27E+01	2.28E+01
Individual Rank	5.5	5.5	9	5.5	5.5	1	5.5	2	5.5
Multiple disk clutch brake design	2.35E–01	2.35E–01	2.35E–01	2.35E–01	2.35E–01	2.35E–01	2.42E–01	4.85E–01	2.35E–01
Individual Rank	4	4	4	4	4	4	8	9	4
Weight minimization of a speed reducer	2.99E+03	2.99E+03	2.99E+03	2.99E+03	2.99E+03	3.01E+03	2.99E+03	2.82E+03	2.99E+03
Individual Rank	5	5	5	5	5	9	5	1	5
Welded beam design	1.67E+00	1.67E+00	1.67E+00	1.67E+00	1.67E+00	1.70E+00	1.67E+00	1.76E+00	1.67E+00
Individual Rank	3.5	7	3.5	3.5	3.5	8	3.5	9	3.5
Robot gripper problem	2.63E+00	0.00E+00	2.57E+00	2.53E+00	2.53E+00	2.66E+00	2.48E+01	2.83E+00	2.53E+00
Individual Rank	6	1	5	2.5	4	7	9	8	2.5
Rank sum	20	29.5	26	27	29	37	39	31.5	25
Final rank	**1**	6	**3**	4	5	8	9	7	**2**

Bold numbers in the table indicate superior values

complicated problems. Comparison of performance within the CEC competition-winning variants of DE has never been done as per our knowledge. In this study, performance of DE and its eight CEC competition-winning variants is studied, evaluating IEEE CEC 2019 function suite, IEEE CEC 2020 function suite, and one unconstrained and five constrained real-world engineering problems. The time taken by the algorithms for solving real-world problems is also assessed. Besides comparing the performance of the algorithms with numerical results, their performance is also analyzed statistically. Performance analysis exposes that SHADE is the best algorithm while solving CEC functions, and in five out of six real-world problems, it has been found the faster in calculating the result. On real-world problems, DE outperforms all its variants used here for comparison. ELSHADE-SPACMA emerged as a unique algorithm ranked 2nd or 3rd in every problem set. Evaluated optimal result in weight minimization of a speed reducer design problem is recorded minimum than the optimal value found in the literature. Finally, it can be concluded that among the compared algorithms, SHADE and ELSHADE-SPACMA can effectively solve complex problems and can be used to solve other problems from engineering and industry.

Data Availability Statements All data generated or analyzed during this study are included in this published article.

Declarations

Conflict of interest All authors of this article declare that there is no conflict of interest associated with this publication, and there has been no significant financial support for this work that could have influenced its outcome.

References

1. Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence, 90*, 103479.
2. Osman, I. H., & Kelly, J. P. (1996). Meta-Heuristics: An Overview. In I. H. Osman & J. P. Kelly (Eds.), *Meta-Heuristics* (pp. 1–21). Springer.

3. Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms. *ACM Computing Surveys, 45*, 1–33.
4. Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, 67–82.
5. Yildiz, A. R., Abderazek, H., & Mirjalili, S. (2019). A Comparative study of recent non-traditional methods for mechanical design optimization. *Archives of Computational Methods in Engineering, 27*, 1031–1048.
6. Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.
7. Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution—an updated survey. *Swarm and Evolutionary Computation, 27*, 1–30.
8. Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review, 33*, 61–106.
9. Eiben, A. E., & Smith, J. E. (2003). *to evolutionary computing* (Vol. 53, p. 18). Berlin: Springer.
10. Zamuda, A., Brest, J., Boskovic, B., & Zumer, V. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. *In IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, China, 2008, *3718–3725*.
11. Krink, T., Filipic, B., & Fogel, G. B. Noisy optimization problems-a particular challenge for differential evolution. *In Proceedings of the Congress on Evolutionary Computation*, Oregon, Portland, USA, 2004, *332–339*.
12. Nama, S., Saha, A. K., & Ghosh, S. (2015). Parameters optimization of geotechnical problem using different optimization algorithm. *Geotechnical and Geological Engineering, 33*, 1235–1253.
13. Kashani, A. R., Chiong, R., Mirjalili, S., & Gandomi, A. H. (2020). Particle swarm optimization variants for solving geotechnical problems: Review and comparative analysis. *Archives of Computational Methods in Engineering, 28*, 1871–1927.
14. Foroutan, F., Mousavi Gazafrudi, S. M., & Shokri-Ghaleh, H. (2020). A comparative study of recent optimization methods for optimal sizing of a green hybrid traction power supply substation. *Archives of Computational Methods in Engineering, 28*, 2351–2370.
15. Thakur, K., & Kumar, G. (2020). Nature inspired techniques and applications in intrusion detection systems: Recent progress and updated perspective. *Archives of Computational Methods in Engineering, 28*, 2897–2919.
16. Patel, V. K., Raja, B. D., Savsani, V. J., & Desai, N. B. (2021). Performance of recent optimization algorithms and its comparison to state-of-the-art differential evolution and its variants for the economic optimization of cooling tower. *Archives of Computational Methods in Engineering, 28*, 4523–4535.
17. Kunakote, T., Sabangban, N., Kumar, S., Tejani, G. G., Panagant, N., Pholdee, N., Bureerat, S., & Yildiz, A. R. (2022). Comparative performance of twelve metaheuristics for wind farm layout optimization. *Archives of Computational Methods in Engineering, 29*, 717–730.
18. Nama, S., Saha, A. K., & Ghosh, S. (2017). Improved backtracking search algorithm for pseudo dynamic active earth pressure on retaining wall supporting c-Φ backfill. *Applied Soft Computing, 52*, 885–897.
19. Demirci, E., & Yıldız, A. R. (2019). A new hybrid approach for reliability-based design optimization of structural components. *Materials Testing, 61*, 111–119.
20. Yıldız, B. S., Yıldız, A. R., Pholdee, N., Bureerat, S., Sait, S. M., & Patel, V. (2020). The henry gas solubility optimization algorithm for optimum structural design of automobile brake components. *Materials Testing, 62*, 261–264.
21. Champasak, P., Panagant, N., Pholdee, N., Bureerat, S., & Yildiz, A. R. (2020). Self-adaptive many-objective meta-heuristic based on decomposition for many-objective conceptual design of a fixed wing unmanned aerial vehicle. *Aerospace Science and Technology, 100*, 105783.
22. Sharma, S., & Saha, A. K. (2020). m-MBOA: A novel butterfly optimization algorithm enhanced with mutualism scheme. *Soft Computing, 24*, 4809–4827.
23. Yıldız, B. S., Yıldız, A. R., Albak, E. İ, Abderazek, H., Sait, S. M., & Bureerat, S. (2020). Butterfly optimization algorithm for optimum shape design of automobile suspension components. *Materials Testing, 62*, 365–370.
24. Nama, S., Saha, A. K., & Sharma, S. (2020). A novel improved symbiotic organisms search algorithm. *Computational Intelligence*. https://doi.org/10.1111/coin.12290
25. Yıldız, A. R., Özkaya, H., Yıldız, M., Bureerat, S., Yıldız, B. S., & Sait, S. M. (2020). The equilibrium optimization algorithm and the response surface-based metamodel for optimal structural design of vehicle components. *Materials Testing, 62*, 492–496.
26. Yıldız, A. B. S., Pholdee, N., Bureerat, S., Yıldız, A. R., & Sait, S. M. (2020). Sine-cosine optimization algorithm for the conceptual design of automobile components. *Materials Testing, 62*, 744–748.
27. Panagant, N., Pholdee, N., Bureerat, S., Kaen, K., Yıldız, A. R., & Sait, S. M. (2020). Seagull optimization algorithm for solving real-world design optimization problems. *Materials Testing, 62*, 640–644.
28. Dhiman, G., Singh, K. K., Slowik, A., Chang, V., Yildiz, A. R., Kaur, A., & Garg, M. (2021). EMoSOA: A new evolutionary multi-objective seagull optimization algorithm for global optimization. *International Journal of Machine Learning and Cybernetics, 12*, 571–596.
29. Chakraborty, S., Saha, A. K., Sharma, S., Mirjalili, S., & Chakraborty, R. (2021). A novel enhanced whale optimization algorithm for global optimization. *Computers & Industrial Engineering, 153*, 107086.
30. Yildiz, B. S., Pholdee, N., Bureerat, S., Yildiz, A. R., & Sait, S. M. (2021). Robust design of a robot gripper mechanism using new hybrid grasshopper optimization algorithm. *Expert Systems, 38*, e12666.
31. Sharma, S., Saha, A. K., Majumder, A., & Nama, S. (2021). MPBOA-A novel hybrid butterfly optimization algorithm with symbiosis organisms search for global optimization and image segmentation. *Multimedia Tools and Applications, 80*, 12035–12076.
32. Chakraborty, S., Saha, A. K., Nama, S., & Debnath, S. (2021). COVID-19 X-ray image segmentation by modified whale optimization algorithm with population reduction. *Computers in Biology and Medicine, 139*, 104984.
33. Qin, A. K., & Suganthan, P. N. Self-adaptive differential evolution algorithm for numerical optimization. *In IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, 2005, *1785–1791*.
34. 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. *In IEEE Transactions on Evolutionary Computation,* Vancouver, Canada, 2006, *646–657*
35. Tanabe, R., & Fukunaga, A. Success-history based parameter adaptation for differential evolution. *In IEEE Congress on Evolutionary Computation,* Cancun, Mexico, 2013, *71–78*.
36. Tanabe, R., & Fukunaga, A. S. Improving the search performance of SHADE using linear population size reduction. *In IEEE Congress on Evolutionary Computation (CEC),* Beijing, China, 2014, *1658–1665*.

37. Awad, N. H., Ali, M. Z., Suganthan, P. N., & Reynolds, R. G. An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. *In IEEE Congress on Evolutionary Computation (CEC),* Vancouver, Canada, 2016, *2958–2965.*
38. Awad, N. H., Ali, M. Z., & Suganthan, P. N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving CEC2017 benchmark problems. *In IEEE Congress on Evolutionary Computation (CEC),* Donostia-San Sebastián, Spain, 2017, *372-379*
39. Mohamed, A. W., Hadi, A. A., Fattouh, A. M., & Jambi, K. M. (2017). LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. *In IEEE Congress on Evolutionary Computation (CEC),* Donostia-San Sebastián, Spain, 2017, *145-152*
40. Hadi, A. A., Mohamed, A. W., & Jambi, K. M. (2018). Single-objective real-parameter optimization: Enhanced LSHADE-SPACMA algorithm. *Heuristics for Optimization and Learning, 906*, 103–121.
41. Price, K. V., Awad, N. H., Ali, M. Z., & Suganthan, P. N. (2018). Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization. *In Technical Report*, *Nanyang Technological University*, Singapore, Singapore , 2018, *1-21*.
42. Kadavy, T., Pluhacek, M., Viktorin, A., & Senkerik, R. SOMA-CL for competition on single objective bound constrained numerical optimization benchmark: a competition entry on single objective bound constrained numerical optimization at the genetic and evolutionary computation conference (GECCO) 2020. *In Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Prague, Czech Republic, 2020, *9–10*.
43. Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191*, 1245–1287.
44. Das, S., & Suganthan, P. N. (2010). Problem definitions and evaluation criteria for CEC competition on testing evolutionary algorithms on real world optimization problems. *Jadavpur University, Nanyang Technological University,* Kolkata, India, 2010, *341–359*.
45. Gu, L., Yang, R. J., Tho, C. H., Makowskit, M., Faruquet, O., & Li, Y. (2001). Optimization and robustness for crashworthiness of side impact. *International Journal of Vehicle Design, 26*, 348–360.
46. Osyczka, A., Krenich, S., & Karas, K. Optimum design of robot grippers using genetic algorithms. In *Proceedings of the Third World Congress of Structural and Multidisciplinary Optimization (WCSMO),* New York, USA, 1999, *241–243*.