A Self-Adaptive Hybridized Differential Evolution Naked mole-rat Algorithm for Engineering Optimization Problems

Rohit Salgotra[a,*], Urvinder Singh[a], Gurdeep Singh[a], Nitin Mittal[b], Amir H Gandomi[c]

[a]*Dept. of ECE, Thapar Institute of Engineering & Technology, Patiala, India*
[b]*Dept. of ECE, Chandigarh University, Mohali, India*
[c]*Faculty of Engineering & Information Technology, University of Technology Sydney, NSW 2007, Australia*

Abstract

This paper presents a self-adaptive hybrid variant of differential evolution (DE) algorithm and naked mole-rat algorithm (NMRA), namely SaDN. The algorithm is altogether a new version, designed to overcome the local optima stagnation and poor exploration properties of DE and NMRA respectively. The new algorithm has been designed by incorporating DE into the worker phase of NMRA while keeping all the major parameters of both the algorithms intact. In order to make the algorithm self-adaptive, seven different mutation strategies have been explored for different parameters, and it was found that Lévy based scaling factor and sigmoidal mating factor are the best parameters. Apart from these parameters, adaptive properties have been introduced to all other parameters so that no user-based initialization of parameters is required. For performance evaluation, the proposed SaDN is tested on CEC 2005, CEC 2014 and CEC 2019 benchmark problems and comparison is performed for variable population size and higher dimension sizes. From the experimental results, it has been found that the proposed SaDN performs better with respect to other major state-of-the-art algorithms from the literature. Apart from that, SaDN is subjected to three engineering design problems and compared with other algorithms. Numerical results demonstrate that SaDN shows better performance and is statistically significant in terms of Wilcoxon's rank-sum test, Freidman's test and computational complexity. **The link for source-code will be added after publishing.**

Keywords: Differential evolution, Naked mole-rat algorithm, Hybrid algorithms, Self-adaptivity, numerical optimization, engineering design problems.

*Corresponding author
Email addresses: r.03dec@gmail.com (Rohit Salgotra), gandomi@uts.edu.au (Amir H Gandomi)

1. Introduction

Over the past three decades, the nature inspired algorithms have continuously dominated every sphere of benchmarks to real world problems. The algorithms have become the backbone of optimization research and have been formulated based on the natural behaviour of certain animal species found in nature. These algorithms have added advantage of better search capabilities, lower number of switching parameters and less requirement of problem based tuning. Thus have been found to dominate every sphere, where classical optimization techniques fail to provide significant results. These algorithms include cuckoo search algorithm (CS) [1], whale optimization algorithm (WOA) [2], salp swarm algorithm (SSA) [3],[4], grasshopper optimization algorithm (GOA) [5], equilibrium optimizer (EO) [6], flower pollination algorithm (FPA) [7], bi-directional evolutionary structural optimization (BESO) [8], discrete artificial fish swarm (DAFS) algorithm [9], drone squadron optimization (DSO) [10], newton metaheuristic algorithm (NMA) [11], subtractive algorithm (SA) [12], Pity Beetle Algorithm (PBA) [13] and others [14], [15], [16]. These are the most recently introduced algorithms in optimization research and are being used in almost every domain of research. That can be better understood from the original domain research that these algorithms have been applied to business management, mathematics, medical images, robotics, artificial immune systems, antenna arrays, image segmentation, Stress minimization problem and others. The major reason for this is that these algorithms have added advantage of faster convergence, fewer chances of local optima stagnation, are highly challenging and have shown significant contribution with respect of their counterparts. Also, some hybrid optimization techniques have been studied to improve the performance of an algorithm such as hybrid optimization based on harmony search (HS) and mine blast algorithm (MBA) namely mine blast harmony search (MBHS) (MBA for exploration and HS for exploitation enhancement) [17], hybridization of iso-geometrical analysis (GIGA) and an improved multi-objective particle swarm optimization algorithm (IMOPSO) for shape and size optimization of variable-thickness bi-directional functionally graded plates (2D-FGPs) with multi-objective optimization [18].

Naked mole-rat algorithm (NMRA) is one such algorithm introduced in the recent past [19] and is based on swarm intelligence theory. By swarm intelligence, we mean that the species under consideration lives in large groups of population and perform their routine tasks in collaboration with each other. The naked-mole rats also live in groups of 50 minimum to 295 maximum number of members and follow a worker-breeder relationship to perform their tasks efficiently. By worker-breeder relationship, the total population is divided into two sub-parts that is workers and breeders.

The workers are meant for performing labour, whereas breeders are found to provide assistance to the queen for mating and other tasks [19]. These breeders are selected from the whole population of the worker pool and ultimately the best breeder mates with the queen. If a particular breeder is not able to cooperate or whose fitness reduces, that breeder is sent back to the worker pool, and in order to equalize the effect, the best performing worker is pushed towards the breeder part. Thus there is every chance for a worker to become a breeder and for a breeder to be shifted towards the worker phase. All this is followed either by interactions or experience. The workers assist each other to perform various tasks such as maintenance, defence, construction and provisioning. The breeders also follow this pattern by cooperating with each other, as only a single breeder mates with the queen [19].

Based on the above discussed patterns of naked mole-rats, the NMRA algorithm was proposed. This algorithm is a new addition to the field of nature inspired computing and has been found to provide reliable results. The algorithm uses the concepts of mating patterns of breeders with respect to the queen to search for a possible solution to the problem under consideration. Here simple concepts of shifting of the best worker towards breeders phase and best breeder drifting closer toward the queen are followed to devise new solutions. However, the algorithm does not use the queen as the potential solution but continuously aims at finding the best breeder who will mate with the queen. The best breeder is thus found, is considered as the prospective solution of the problem under consideration. The process of shifting the workers towards the breeder phase is controlled by breeder probability, which helps the algorithm in switching between the breeder phase and the worker phase. Let us briefly discuss both of these phases. The worker phase is controlled by two random mole-rats in the close proximity of each of other, whereas the breeder phase is controlled by the current best mole-rat and some random breeder in the close proximity of the current best solution. Here it should be noted that the breeder and workers are governed by simple random scaling factor commonly referred to as a mating factor. Thus overall, the algorithm aims to achieve a simpler yet robust implementation to achieve a better optimal solution.

In terms of local and global search, the worker phase of NMRA corresponds to exploration and breeder phase corresponds to the exploitation operation. The exploration operation is governed by two random solutions, whereas exploitation operation has one random solution, which is close to the current best solution. Though NMRA is a significant addition to the field of nature inspired algorithms, it also suffers from certain problems. The algorithm follows a simpler structure and is easier to implement, but as the problem complexity increases, the algorithm is prone to certain disadvantages. The algorithm has an overall inadequate exploration due to less randomization in the solution quality. Aforementioned is because in the exploration part, the random solution generated are very close to each other and hence only certain sections of the search space are explored rather than the whole of the search space. Overall, the exploration part is very weak and may lead to local optima stagnation problems. The exploitation operation, on the other hand, is very simple and is found to provide reliable local search capability.

So, it can be said that the NMRA is prone to local optima stagnation due to poor exploration and overall poor worker phase. Thus, in order to provide efficient performance in terms of better exploration, the NMRA needs to be enhanced. In present work, the worker phase is enhanced by using the concepts of various equations of DE, including DE/rand/1, DE/best/1, DE/current-to-best/1, DE/best/2 and DE/rand/2 strategies [20]. All of these equations are discussed in the consecutive sections and are randomly selected based on a certain probability. In a simple DE algorithm, these equations are referred to as mutation equations and are followed by the crossover operation. Here also, the crossover operation of the basic DE is followed in the worker phase based on certain random crossover probability. This helps the algorithm is finding a better solution by the added advantage of better exploration capabilities of the DE based search equations. The basic structure of the new proposed algorithm is the same as that of the original NMRA, with DE algorithm based equations incorporated in the worker phase. Apart from the enhancements in the exploration and exploitation properties, self-adaptive parameters are added in the proposed SaDN algorithm. The self-adaptive properties are added in such a way that no user-based tuning of parameters is required. Here it should be kept in mind that the exploration operation is performed by DE based worker phase, exploitation by using NMRA breeder phase and self-adaptive characteristic is implemented by using seven different kinds of mutation inertia weights.

The proposed SaDN mainly consists of three major parameters namely scaling factor of worker phase, the randomized mating factor (λ) of the breeder phase and the crossover rate (CR) of DE algorithm. In present work, mutation scaling factors are adapted by using five different mutation operators (Lévy mutation, Cauchy mutation, neighbourhood based mutation, trigonometric mutation and diversity mutation); seven different mating factors (exponential decreasing, linearly decreasing, uniform random, sigmoidal decreasing, oscillating, simulated annealing and logarithmic decreasing randomization); and an adaptive CR is used. The best parameters thus selected from these randomized parameters are Lévy based mutation and sigmoidal decreasing mating factor. The final parameter which helps in switching between the exploration and exploitation (worker to breeder switching) is the breeding probability (bp), and five different variations for this parameter are analysed. It has been found that lower the value of bp, higher is the working capability of the algorithm. Thus after careful investigation, in present work, $bp = 0.05$ has been used. Overall, the proposed SaDN algorithm has the added advantage of both DE and NMRA algorithms and is expected to provide reliable results.

Apart from the introduction, the rest of the article is organized into five more sections. Here section 2 details about the basics of NMRA algorithm and all generalizations regarding the algorithm are presented in this section. In section 3, the various mutation weights and mating factors have been discussed in detail. This section deals with the overview of all the major parametric adaptations employed in the proposed SaDN algorithm. In section 4, the new SaDN algorithm is proposed. This section highlights the major drawbacks of the existing algorithm, the requirement of the proposal, the motivation behind the proposal and the various steps that have been followed to implement the algorithm. The section also provides computational complexity of the proposed algorithm with respect to the basic NMRA. In section 5, numerical results are presented. These results are extensively performed with respect to some major algorithms such as JADE [21], SHADE [22], LSHADE-SPACMA [6], CMA-ES [6] and others [23]. The benchmark functions used are CEC 2005 [24], CEC 2014 [25] and CEC 2019 [26]. Both these datasets are highly challenging and consist of unimodal, multimodal, fixed dimension, hybrid and composite functions. Apart from numerical benchmarks, the algorithm has been applied for optimization of engineering design problems. The problems used are highly complex, and a comparison of the proposed algorithm is performed with respect to other newly introduced models in the literature. More discussion about the same is presented in the consecutive subsections. Apart from the experimental results, statistical testing based on Wilcoxon's rank-sum test and Freidman's tests [27] have been performed to prove the superior performance of the proposed SaDN algorithm. A detailed discussion on the proposed algorithm, a summary of results, some insightful implications and drawbacks are also presented in the same section. In the final section 6, the concluding remarks and future recommendations are presented. The outline of the article is given in Figure 1.

2. Naked mole-rat algorithm (NMRA)

The naked mole-rats patterns follow worker breeder relations and their synergy on who will mate with the queen. The best worker among the worker pool gets the chance to become a breeder, and similarly, the best breeder becomes the mating partner for the female queen. The NMRA algorithm

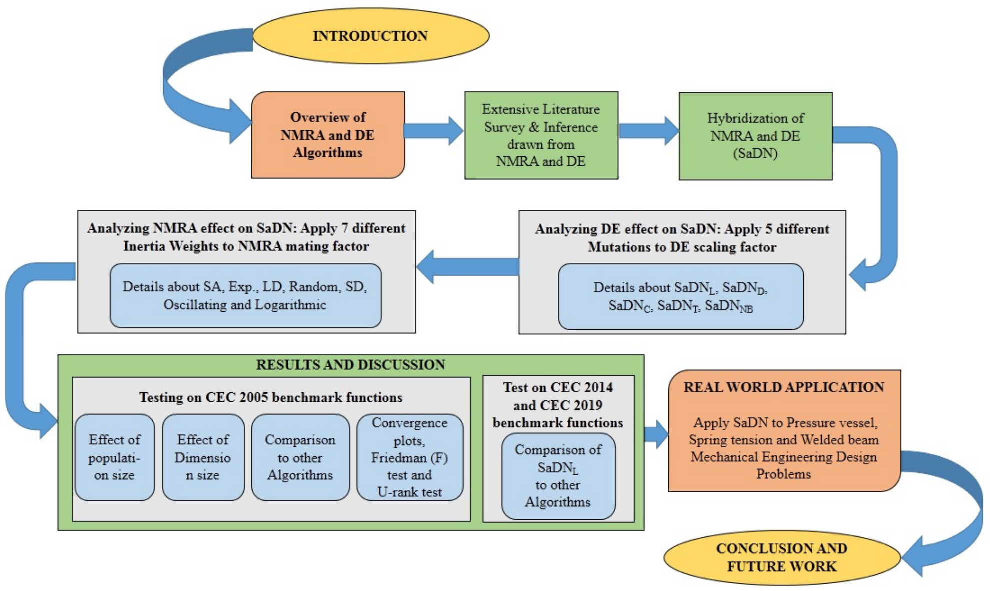
INTRODUCTION
Overview of NMRA and DE Algorithms
Extensive Literature Survey & Inference drawn from NMRA and DE
Hybridization of NMRA and DE (SaDN)
Analyzing DE effect on SaDN: Apply 5 different Mutations to DE scaling factor
Details about SaDN$_L$, SaDN$_D$, SaDN$_C$, SaDN$_T$, SaDN$_{NB}$
Analyzing NMRA effect on SaDN: Apply 7 different Inertia Weights to NMRA mating factor
Details about SA, Exp., LD, Random, SD, Oscillating and Logarithmic
RESULTS AND DISCUSSION
Testing on CEC 2005 benchmark functions
Effect of populati-on size
Effect of Dimensio n size
Comparison to other Algorithms
Convergence plots, Friedman (F) test and U-rank test
Test on CEC 2014 and CEC 2019 benchmark functions
Comparison of SaDN$_L$ to other Algorithms
REAL WORLD APPLICATION
Apply SaDN to Pressure vessel, Spring tension and Welded beam Mechanical Engineering Design Problems
CONCLUSION AND FUTURE WORK

Figure 1: Outline of the article

follows the mating patterns of the naked mole-rats. The algorithm is simple in structure and follows four basic rules as given by

- NMRs founds in a group of an average 70-80 members, and the maximum number can be up to 295.
- The whole population is divided into two parts: workers and breeders, which are led by a single queen.
- The workers have performed all the minor tasks, and the best performing worker has a chance to become a breeder. These breeders are meant for the mating process.
- The best breeder among all the members of breeder pool mates with the queen.

Here it should be noted that the NMRA algorithm searches for the best breeding mate rather than the queen and this behaviour is mathematically modelled as

Initialization of NMRs: Like all optimization algorithms, the first step is to initialize the NMR population randomly. Here n random members of NMR group are initialized within the range of $[1, 2, ..., n]$ for a D dimensional problem. Each NMR is initialized as

$$NMR_{i,j} = NMR_{min,j} + U(0,1) \times (NMR_{max,j} - NMR_{min,j}) \tag{1}$$

where i varies in the range of $[1, 2, 3....n]$, j in $[1, 2, 3....D]$, $NMR_{i,j}$ is the i^{th} solution for the j^{th} dimension, $NMR_{min,j}$ and $NMR_{max,j}$ are the lower and upper boundary conditions for the problem under consideration. The $U(0,1)$ is a uniform random number in the range of $[0,1]$. After initialization, the objective is to find breeders and workers and in turn, find the initial best solution. The next step is to perform the worker and the breeder phase.

Worker phase: This phase is meant for workers to improve their fitness over consecutive iterations and move towards the breeder phase. Here the fitness of the new worker NMR is evaluated, and if the new mating fitness is better than the fitness of the previous iteration, the new solution is memorized, and the old one is discarded. After all the workers complete the search phase, the new solution is retained. The general equation of the worker phase is given by

$$x_i^{t+1} = x_i^t + \lambda(x_j^t - x_k^t) \tag{2}$$

where, x_i^{th} is the solution of i^{th} worker for the t^{th} iteration, x_i^{t+1} is defined as a new solution, λ is the mating factor, x_j^t and x_k^t are random solutions from the worker's pool. The λ is the uniform randomized in the range of [0,1]. Note that the new solution x_i^t corresponds to each worker and they try to improve their fitness to become breeders.

Breeder Phase: The breeders are also updated with respect to iterations and follow a similar search pattern as done by workers. The breeder NMR's are updated using a certain mating probability also called as the breeding probability (bp), in the range of $[0,1]$. The new breeder is evaluated with respect to the overall best solution (y_{best}) and is updated consecutively. Some of the breeders are not able to update their positions and are subsequently pushed towards the worker phase. This theoretical formulation is mathematically given by

$$y_i^{t+1} = (1-\lambda)y_i^t + \lambda(y_{best} - y_i^t) \tag{3}$$

where, y_i^t is the i^{th} breeder or solution in t^{th} iteration, y_i^{t+1} corresponds to a new solution or breeder in the next iteration. λ decides the mating rate of breeder's with a queen. In this case, bp has an initial value of 0.5. This bp decides the extent of the overall worker phase and breeder phase. As the breeding probability increases, the exploration also increases and vice versa. Thus overall, we can say that bp is another parameter that played a crucial part in the NMRA.

Algorithm 1 Pseudo-code of NMRA algorithm

```
Begin
Define: population size ($\boldsymbol{n}$);
        stopping criteria; problem dimension ($D$)
if $i = 1 : t_{max}$ then
    For j=1:Workers
        Worker phase using Eq. 2
        Evaluate fitness of workers
        Find the best worker
    For j=1:Breeders
        Breeder phase using Eq. 3
        Evaluate fitness of breeders
        Find the best breeder
            Combine the pool of Workers & Breeders
        Find the current best solution
Update final best
End
```

3. Parametric adaptations

This section deals with the basic preliminaries of the parametric adaptations for the proposed algorithm. The section is divided into two subsections where the first subsection deals with the adaptations followed on the DE based worker phase, and the second subsection is meant for NMRA based parametric adaptation. The more in-depth analysis of the proposed algorithm is presented in the next section. The main aim of this section is to highlight various parameters which are playing a significant role in enhancing the exploration and exploitation properties of the proposed algorithm.

3.1. ***DE variant analysis:***

Here the DE variants are added in SaDN worker phase to enhance the exploitation capability of basic NMRA. It is analyzed here in terms of five different mutations such as Diversity Mutated self-adaptive hybridized DE and NMRA Algorithm ($SaDN_{Div}$), Cauchy mutated SaDN ($SaDN_C$), Trigonometric SaDN ($SaDN_T$), Neighbourhood based SaDN ($SaDN_{NB}$) and Lévy mutated SaDN ($SaDN_L$). Some common examples include mutation operators for FPA [28], GA [29] and others.

3.1.1. Diversity Mutated SaDN Algorithm: ($SaDN_{Div}$)

The concept of diversity mutation was suggested by [30] for modifying genetic algorithm (GA). It is a unique kind of mutation strategy and is generally followed where there is a requirement of only a single mutation per individual. The diversity mutation step size is generated using a certain type of adaptive exponential probability distribution ($p(i) = te^{-ti}$) for $i \in [0, n-1]$. The generalized distribution for a fixed value of n is given by

$$te^{-nt} - e^{-t} - t + 1 = 0 \tag{4}$$

The above equation is generalized for a random $u \in [0, 1]$ and is given by

$$\lambda(t) = \frac{1}{t}\log(1 - u(1 - e^{-nt})) \tag{5}$$

This strategy aims to increase the diversity among the search agents and hence improve the exploration properties of NMR algorithm. The diversity mutated strategy based algorithm is named as ($SaDN_{Div}$), and the general equation is inspired by [30].

3.1.2. Cauchy Mutated SaDN Algorithm: ($SaDN_C$)

Most of the domain research algorithms are prone to premature convergence, making the algorithm fall in some local minima. To overcome the effect of small mutations of the conventional evolutionary programming (CEP) techniques, a Cauchy based distribution was proposed [29] for CEP and recently extended to FPA [28] and CS [31]. The Cauchy based randomization is based on Cauchy distribution and is given by

$$y = \frac{1}{2} + \frac{1}{\pi} arctan(\frac{\delta}{g}) \tag{6}$$

The Cauchy density function is given by

$$f_{C(0,g)}(\delta) = \frac{1}{\pi} \frac{g}{g^2 + \delta^2} \tag{7}$$

where $g = 1$ is scale parameter, $y \in [0, 1]$ is a random number following a uniform distribution. Now the Cauchy distributed random number is given by equation 8 and is represented as

$$\delta = tan(\pi(y - \frac{1}{2})) \tag{8}$$

For a more in-depth understanding, the Cauchy based mutation operator is meant for providing larger step sized mutations and ultimately aiming for a better exploration of the search space. The major reason of larger step size of Cauchy distribution is because of its fatter tailed structure, which helps to generate larger steps over subsequent iterations and hence overcomes the problem of premature convergence and enhances the exploration operation. Due to the Cauchy mutation operation, the algorithm becomes capable of searching more solutions within the whole search space and makes it possible for the exploration phase to enhance the search process in an efficient way.

3.1.3. Trigonometric SaDN Algorithm: ($SaDN_T$)

This kind of mutation strategy has already been explored on the traditional DE and was found to provide reliable results [32]. The trigonometric mutation is followed by using three different individuals from the current population. From the three members, one is a perturbed scaled differential vector with respect to the other two and is perturbed in order to produce mutation vectors. Three individuals x^t_{r1}, x^t_{r2} and x^t_{r3} are initialized, where X^t_i is the i^{th} individual in the t^{th} generation, $r_1, r_2, r_3 = 1, ...n$ for $r_1 \neq r_2 \neq r_3 \neq i$ and n being the population size. The three members are taken from the centre of a hypergeometric triangle, and the perturbation is performed by using the weighted sum of three differential vectors given by

$$x^{t+1}_i = \frac{x^t_{r1} + x^t_{r2} + x^t_{r3}}{3} + (p_2 - p_1)(x^t_{r1} - x^t_{r2}) + (p_3 - p_2)(x^t_{r2} - x^t_{r3}) + (p_1 - p_3)(x^t_{r3} - x^t_{r1}) \tag{9}$$

In this case p_1, p_2, p_3 are the random perturbation weights. These individuals are required as $p_1 = 1$ and $p_2 + p_3 = 0$ to become better output. Here, the equation (9) is used and adapted to formulate the new equation for the proposed trigonometric mutated SaDN ($SaDN_T$). The general equation after parameter adaptation is given by

$$x_{t+1} = \frac{x^t_{r1} + x^t_{r2} + x^t_{r3}}{3} + x^t_{r2} + x^t_{r3} - 2x^t_{r1} \tag{10}$$

This kind of strategy is also meant for increasing the diversity among the search agents and the exploration process on the whole. The generalized discussion on how this modification has been introduced to the proposed SaDN algorithm is presented in the subsequent section.

3.1.4. Neighbourhood based SaDN Algorithm: ($SaDN_{NB}$)

Neighbourhood based mutation has been inspired by [33] and has been used in the basic DE algorithm. The new algorithm is named as ($SaDN_{NB}$) and uses multiple random solutions which are immediate neighbours of each other. The most important point which is to be kept in mind is that the vector generated must preserve the diversity of every individual within the search space.

$$x^t_i = X^t_i + m \times (X_{(n_{best})} - X_i) + n \times (X_p - X_q) \tag{11}$$

where $X_(n_{best})$ is the best vector in the X_i and $p, q \in [i - r, i + r](p \neq q \neq i)$ neighborhood, and m, n are scaling factors of randomized numbers ∈ [0,1]. Assume that in the current NMR population $X = (X_1, X_2, X_3, ..., X_n)$, $X_i(i \in [1, n])$ is a vector preserve the neighborhood's diversity and its dimension is D. Also, the new solution should be generated from the same generalized neighbourhood, and the size of the neighbourhood population must be less than the total population size. The new best solution is updated according to equation (11) in the improved version of SaDN, and the updated solution performs the worker phase. The new modified equation based on neighbourhood search is given by

$$x^{t+1}_i = x^t_i + U[0, 1](x^t_k - x^t_m) \tag{12}$$

where k and m are the radius of the total population vector should vary from $[0, (n-1)/2]$ and must be non-zero vectors. The solutions x^t_k and x^t_m are two random solutions with respect to k^{th} and m^{th} NMRs of the worker phase, which are generated around the new solution x^t_i, where $k \neq m$ and U[0,1] is a randomized scaling factor. Here, the general equation (2) of the worker phase is modified as to equation (12) in according to neighbourhood search. The primary aim is to increase diversity among the search agents to improve the overall global search or exploration of SaDN.

3.1.5. Lévy Flight based SaDN Algorithm: ($SaDN_L$)

Lévy flight mechanism is followed to generate new steps, and because of its fatter tailed, it is capable of generating larger step sizes and thus helping the algorithm in providing better exploration properties [28]. Further in [34], it has been formulated that Lévy flights based global search helps in exploring the search space in a much better way and helps in maintaining diversity among the search agents. The generalized equation of a Lévy distribution is given by

$$L(D) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\beta}} \quad r_1, r_2 \in [0,1]; \beta = 1.5 \tag{13}$$

where

$$\sigma = (\frac{\Gamma(1+\beta) \times sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2(\frac{\beta-1}{2})})^{1/\beta} \quad \Gamma(x) = (x-1)! \tag{14}$$

where $L(D)$ is an Lévy distributed random number for a D dimensional vector, α, β and Γ are other important parameters that are kept as random numbers. The Lévy distributed randomisation is found to have very positive results, and applying such operators helps the algorithm explore the search space in a much better way.

3.2. ***NMRA variant analysis***

Here the NMRA variants are analysed in SaDN breeder phase to enhance the exploration ability of basic NMRA. It is analysed here in terms of seven different inertia weights applied on NMRA mating factor (λ), which includes simulated annealing (SA), exponential (Exp), logarithmic decreasing (LD), sigmoidal decreasing (SD), oscillating (Osci), random (Random) and logarithmic decreasing (Log) inertia weights (IW). These inertia weights are inspired from particle swarm optimization (PSO) [35],[36] and have been applied extensively in the breeder phase. Here it should be noted that the parameter λ of the breeder phase is changed with respect to the inertia weights. A detailed overview of the basics of each of these inertia weights is presented as

3.2.1. Simulated Annealing inertia weight: (λ_{SA})

The simulated annealing based inertia weights has been taken from [37] and is meant for improving the convergence of an algorithm. The inertia weight is found to provide better exploitation properties and was first exploited from urban planning of block fitting [37]. The generalised mathematical formulation for this inertia weight is given by

$$\lambda_k = \alpha_{min} + (\alpha_{max} - \alpha_{min}) \times p^{k-1} \tag{15}$$

Here, for present case $\alpha_{min} = 0.5$; $\alpha_{max} = 0.9$; $p = 0.95$; k is a random number and is generally expressed from a uniform distribution of $[0,1]$. This inertia weights help provide a balanced breeder phase and more precisely, make the algorithm more efficient in exploitation operation. The SA is a new technique that uses certain parameters same as the NMRA seems to pose the best results. Moreover, the experimental results show that the SA based inertia weight provides the best solution compared to other variants.

3.2.2. Exponential randomized inertia weight: (λ_{Exp})

This type of inertia weight is found to have better convergence speed and helps the algorithm in performing exploration during the initial stages and exploitation towards the end [38]. Various researchers [39] has exploited the exponential inertia weight, and it has been found to provide better exploitation properties. The mathematical formulation for this inertia weight is given by

$$\lambda = \lambda_{min} + (\lambda_{max} - \lambda_{min}) \times e^{(\frac{-t}{t_{max}/10})} \tag{16}$$

where $\lambda_{min} = 0.95$ $\lambda_{max} = 0.4$, t is the iteration counter and t_{max} indicates maximum number of iterations. The major benefit of using exponential randomization based inertia weight is to improve the local search capabilities by avoiding a problem of local optima stagnation.

3.2.3. Logarithmic Decreasing inertia weight: (λ_{Log})

It has been already the fact that a larger value of λ means more global search and smaller value means local search. A decreasing inertia weight helps in reducing the value of λ consecutively over time and improve the exploration as well as exploitation capabilities of the algorithm. Logarithmic inertia weight works in a similar fashion and provides more viable solutions in balancing the algorithm for a proper global and local search operation [40]. The generalized equation for logarithmic inertia weight is given by

$$\lambda = \alpha_{max} + (\alpha_{min} - \alpha_{max}) \times \log_{10}(k + \frac{10t}{t_{max}}) \tag{17}$$

Here, $\alpha_{max} = 0.9, \alpha_{min} = 0.25$, k is a constant kept as 1 and is used to adjust the evolutionary speed.

3.2.4. Sigmoidal Decreasing inertia weight: (λ_{SD})

The sigmoidal inertia weight strategy is a typical randomization strategy where the inertia weight does not always decrease at each iteration [41], [42]. At the beginning of the search process, the value of inertia weight is largely due to the global search capabilities, whereas towards the end, the values start reducing abruptly. There is a minimal change between the large and small inertia weights, and this method helps to add a proper balance between the exploration and exploitation operation. The generalized equation for this phase is given by

$$\lambda = \frac{\alpha_{min} - \alpha_{max}}{1 + e^{-u\times(iter - h\times gen)}} + \alpha_{max} \tag{18}$$

$$u = 10^{\log(gen)-2} \tag{19}$$

where, $\alpha_{max} = 0.9, \alpha_{min} = 0.5$, $gen = 51$, h and k are the randomized numbers varies in the range of [0,1].

3.2.5. Oscillating inertia weight: (λ_{Osci})

The oscillating inertia weight generated periodic waves between exploration and exploitation operation [43, 44]. This inertia weight is found to be very competitive and also improves the convergence speed without getting trapped in some local minima. The temporal behavior of this inertia weight is modelled as

$$\lambda = \frac{\alpha_{min} + \alpha_{max}}{2} + \frac{\alpha_{min} - \alpha_{max}}{2} + \cos\frac{2\pi t}{T} \tag{20}$$

$$T = \frac{2S_1}{3 + 2k} \tag{21}$$

Here, $\alpha_{max} = 0.9, \alpha_{min} = 0.5$, S_1 and k are the randomized numbers varies in the range of [0,1].

3.2.6. Random inertia weight: (λ_{Random})

The random inertia weight is a simple yet efficient strategy and is inspired by [45]. It was formulated that this inertia weight also improves the exploitation properties and helps in searing potential solutions in a concise search space. The generalized equation for random inertia weight is given by

$$\lambda = 0.5 + \frac{rand()}{2} \tag{22}$$

3.2.7. Linearly Decreasing inertia weight: (λ_{LD})

The linearly decreasing inertia weight is inspired by [46, 31] and is meant for improving the efficiency and performance of an algorithm. The algorithm can converge to a global optimum solution instead of falling in some local optima. Overall this inertia weight plays a significant role in improving the exploitation properties of an algorithm. The generalized equation for this inertia weight is given by

$$\lambda = \alpha_{max} - \frac{\alpha_{max} - \alpha_{min}}{t_{max}} \times k \tag{23}$$

where $\alpha_{max} = 0.9$, $\alpha_{min} = 0.4$, and t_{max} is the maximum number of iterations. Here throughout the present iterations, the weight of randomization parameters decreases linearly, shifting the algorithm from exploration to exploitation operation. A further concentrated search is followed using the new improved λ inertia weight, and hence better-searching capabilities are desired in the breeder phase.

In the next section, a detailed analysis of the proposed algorithm and how these parameters have been incorporated into the SaDN algorithm is presented.

4. The proposed: SaDN Algorithm

NMRA is a recently introduced algorithm and has the potential to become a global problem solver. The algorithm is very simple and linear in nature and is found to provide reliable results in comparison to other recently introduced algorithms such as GWO, WOA, FA, CS and others in its basic form. The algorithm though is highly efficient but has a poor exploration phase. It can be estimated from the fact that the random solutions generated in the worker phase are far away from each other, and hence chances of local optima stagnation are very high. Thus there is a requirement of providing new prospective equations so that the worker phase can be enhanced significantly, and hence optimal global solutions are obtained. Apart from that, the parameters of NMRA also plays a very significant role in the performance of the algorithm. The major parameter is the scaling factor, and the mating factor is very important in analysing the performance of the NMRA algorithm. Adaptive properties have been added in present work, to make the algorithm self-sufficient and hence no user based initialisation of variables is required. The major highlights of the proposed work are

- The DE algorithm has been hybridized with the recently proposed NMRA and a new algorithm, namely SaDN, has been proposed. The DE algorithm has been incorporated in the worker phase of NMRA, and no modification has been introduced to the breeder phase of the algorithm.
- The self-adaptive property has been added to SaDN by making all the parameters adaptive in nature. Thus no user based initialization is required and hence ensuring that the algorithm becomes self-resilient.
- The combined parameters of DE and NMRA are added to the basic structure of the proposed SaDN algorithm. Here CR is adapted by using a general parametric adaptation of marine predator algorithm (MPA).

- Five different mutation operators have been added to the scaling factor parameter in the worker or local search phase and are meant for improving the exploration operation of the proposed SaDN algorithms.
- The mating factor is also subjected to seven different inertia weight strategies. These mating factors help in improving the exploitation properties of the proposed algorithms.
- The best among the mutation weights (Lévy mutation) and inertia weights (sigmoidal inertia weight) are combined along with the added hybridization, to formulate the new SaDN algorithm.

A detailed study of the requirement of the proposal and how the proposed work adds significance to the ever-expanding literature of nature inspired algorithms is presented in this section.

4.1. Why we need this new proposal

It is already a well known fact from the no free lunch theorem (NFL) that a single optimization without any adaptations cannot be used for solving all the optimization research problems [47]. It is because the dimension size of the problem may pose a severe challenge for the algorithm under consideration. The variation in the number of local optima and the presence of high peaks may push the algorithm further apart from the optimal global solution. Thus it becomes essential to adapt and formulate new algorithms that can solve the problem under consideration with minimal constraints. The basic structure of both the algorithms is kept intact with the added modifications. DE algorithm is highly efficient in exploration operation, and this is because of the presence of more than one random solution with in the whole of the search space. Thus, making the algorithm search for potential solutions in the search space, but the exploitation is of less significance and needs to be added. The major reason for such a response is that the DE algorithm consists of multiple different equations, and there is no particular time when one equation will add advantage. So maybe the equation where the solution is varied with respect to the best solution is added in the initial phases whereas, for other cases, the algorithms can change to different equations. Overall, these added equations lead to a better exploration operation, and hence there is a requirement of a better exploitation operation to be introduced to make the algorithm worthwhile. NMRA, on the other hand, has two phases, namely worker and breeder phase representing exploration and exploitation operation, respectively. The exploration operation here is controlled by two random solutions in the close vicinity of each other. It makes the algorithm search only particular sections of the search space, whereas other sections of the search, may contain the potential solution, remain completely out of search. Thus the algorithm gets trapped in some local minimal, and the final solution may never be found. Overall we can say that NMRA has a poor exploration operation and needs to be enhanced to make the algorithm more efficient. On the other hand, it has a very efficient exploitation operation. This can be estimated from the fact that the global solution of any problem lies somewhere in the close proximity of the current best solution during the final stages and every new solution generated is thus compared with the current best to check for the final potential solution. In the breeder phase of NMRA, the solution is searched with respect to the current best solution and hence provides better reliable solutions. Thus adding the exploration properties of DE and exploitation capabilities of NMRA can make the algorithm highly efficient. In the next subsection, the proposed algorithm has been highlighted. Here it should be noted that only DE based equation modifications have been presented in this section whereas original equations of NMRA are kept the same as discussed in the previous sections.

4.2. The proposed SaDN algorithm

It has been already discussed that the proposed SaDN algorithm is a self-adaptive hybrid version of DE and NMRA algorithms. This section provides a detailed study of the various parts of the proposed SaDN algorithm. It consists of four major parts including, initialization, worker phase, breeder phase and the selection operation. Each of these is elaborated in consecutive subsections.

4.2.1. Initialization

This is the first step and is common in almost every other algorithm. Here the algorithm starts by random initialization of search agents with in a particular range and the general equation is thus given by

$$S_{i,j} = S_{min,j} + U(0,1) \times (S_{max,j} - S_{min,j}) \tag{24}$$

where i lies in the range $[1, 2, 3....n]$, j as $[1, 2, 3....D]$, $S_{i,j}$ is i^{th} solution for the j^{th} dimension, $S_{min,j}$ and $S_{max,j}$ are lower and upper bounds of the problem under test and $U(0,1)$ is a uniform random number in the range of $[0, 1]$.

4.2.2. Worker Phase

After initialisation, the second step is the exploration operation, which is governed by the worker phase of NMRA algorithm. This phase helps the algorithm in searching for potential solutions within the whole of the search space and is important to analyse the local search capabilities of the proposed algorithm [20]. In present work, the DE based equation modifications have been introduced in the proposed algorithm. At each generation, DE employs two basic operations, namely mutation and crossover. A scaling factor controls the mutation, whereas crossover is controlled by using the crossover rate. The target solution is achieved by five different equations and are given by equation (25)

$$v_i^t = \begin{cases} x_{r_1}^t + F.(x_{r_1}^t - x_{r_2}^t); & "DE/rand/1" \\ x_{best}^t + F.(x_{r_1}^t - x_{r_2}^t); & "DE/best/1" \\ x_i^t + F.(x_{best}^t - x_i^t) + F.(x_{r_1}^t - x_{r_2}^t); & "DE/currenttobest/1" \\ x_{best}^t + F.(x_{r_1}^t - x_{r_2}^t) + F.(x_{r_3}^t - x_{r_4}^t); & "DE/best/2" \\ x_{r_1}^t + F.(x_{r_2}^t - x_{r_3}^t) + F.(x_{r_4}^t - x_{r_5}^t); & "DE/rand/2" \end{cases} \tag{25}$$

The equation (25) thus formulated is the mutation operation of DE algorithm. The crossover operation is given by

$$x_i^t = \begin{cases} v_i^t, & if(rand_j[0,1] \leq CR), \;\; j = 1,2,...,n \\ x_i^t, & otherwise \end{cases} \tag{26}$$

where x_i^t is the current solution of the i^{th} member of the population having j dimension, v_i^t is the velocity of the target solution, r_1, r_2, r_3, r_4 and r_5 are mutually exclusive random solutions taken from the pool of whole NMRA population and are different from the current solution i having j dimension, CR is the crossover rate, F is the scaling factor, x_{best} is the best solution and t is the current generation. In the proposed SaDN algorithm, both mutation and crossover operations have been introduced in the worker phase of the algorithm. Note that only one single equation among the five equations of equation 26 is selected at a particular iteration based on random probability. Thus any equation can be selected at any particular generation, providing better searching capabilities with in the search space.

4.2.3. Breeder Phase

This is the third phase of SaDN algorithm and is governed by the basic equations of the NMRA algorithm. Since this phase is mainly meant for exploitation operation, the solutions are generated with respect to the current best solution and are controlled by the mating factor (λ). The major reason for no change in this phase is that the breeders are limited in number and are meant for searching within particular sections of the search space. They do not correspond to any exploration operation and are found to provide potential searching capabilities in the close proximity of the current or the previous best solution, thus providing better exploitation properties. The next section details about the selection operation performed for evaluation of the proposed algorithm.

4.2.4. Selection Operation

The fourth and the final phase is the selection operation. In present case, a greedy selection operation is followed to find the current best solution. Here the new best solution is identified based on the current best solution and previous solution. If the previous solution is better than the newly generated solution, the previous solution is kept where as if the new solution is better, the previous solution is replaced by the new solution. For a generalized minimization process having $f(x_i^t)$ fitness for the x_i^t solution, the selection procedure is given by equation (27)

$$x_{new}^{t+1} = \begin{cases} x_{new} & iff(x_{new}) < f(x_i^t) \\ x_i^t & otherwise \end{cases} \tag{27}$$

Apart from the basic structure discussed above, the proposed SaDN algorithm also consists of some major parametric adaptations of both DE and NMRA. The next section details about these parameters and the procedure followed to adapt these parameters.

4.2.5. Parametric Adaptations

There are two major parameters for both the algorithms, namely CR and F for DE whereas bp and λ for NRMA. In the basic algorithms, these parameters are simply random numbers and modification is required to add adaptivity in the algorithm. Here the parameter F of DE is subjected to five different mutation weight Five different mutation operators have been added to the scaling factor parameter namely Lévy mutation (L), Cauchy based mutation (C), neighbourhood based mutation (NB), trigonometric mutation (T), and diversity mutation (Div). The mating factor for NMRA is also subjected to seven different inertia weight strategies including uniform random distribution ($random$), linearly decreasing initialization weights (LD), exponential decreasing distributed operator (Exp), oscillating inertia weight ($Osci$) simulated annealing based mutation (SA), the sigmoidal decreasing operator (SD) and logarithmic mutation (log). Apart from these parameters, bp has been subjected to five different values such as 0.05, 0.25, 0.50, 0.75 and 0.95. After careful investigation, it has been found that bp = 0.05 provides the most reliable results. Based on the experimental analysis, it has been found that Lévy distributed scaling factor and sigmoidal mating factor presents the best combination for the proposed algorithm. The final parameter is the CR

which has also been adapted based on the current iteration (t) and maximum number of generation (t_{max}), is inspired from MPA [48] and is given by equation (28)

$$CR = (1 - \frac{t}{t_{max}})^{2\frac{t}{t_{max}}} \tag{28}$$

All of the above discussed modifications, adds up to formulate the new algorithm. Note that the proposed modifications are added independently without compromising the basic properties of both DE and NMRA algorithm. The pseudo code of SaDN is shown in Algorithm 2.

Algorithm 2 Pseudo-code of the proposed SaDN algorithm

```
Begin
Define: population size ($\boldsymbol{n}$); Initial parameters ($\boldsymbol{F}$, $\boldsymbol{CR}$ and $\lambda$);
        stopping criteria & problem dimension ($D$)
if $i = 1 : t_{max}$ then
    Worker Phase:
        With respect to best by using Eq. (25)
    Breeder Phase:
        With respect to best by using Eq. (3)
    Greedy Selection:
        Evaluate fitness using Eq. (27)
    Update $F$ using Lévy mutation operation
    Update $\lambda$ using $SD$ inertia weight
    Update $CR$
update final best
End
```

4.3. Computational complexity

Computational complexity of any algorithm plays a very significant role in the performance of an algorithm and is meant for calculating the worst case complexities and generalised operations of any algorithm. Before analysing the complexities of proposed SaDN algorithm, let us compute the computational cost of NMRA. Here the computational complexity for NMRA is a function of n members of the population which are working in a D dimensional search space. The complexity for the first time is given by O($n.D$) and for a total of t_{max} generations, the complexity becomes O($n.D.t_{max}$). Similarly, for the proposed SaDN algorithm, each member of the population for any particular dimension D, performs a total of $\mathrm{O}(D)$ operations to complete that particular task. Similarly, for n members of the population, the computational burden increases n times and is given by $\mathrm{O}(n.D)$. Also, no algorithm can provide exact output in a single iteration, so some t_{max} number of iterations are performed to achieve the final solution. Thus this factor is also multiplied with the total operational burden, and the total complexity for this is given by $\mathrm{O}(n.D.t_{max})$. Now, suppose we compare the computational complexities of both the algorithms. In that case, it can be seen that added hybridisation and new structure does not add any challenge to the computational complexity of the proposed algorithm and the overall complexity of SaDN remains same as that of original NMRA.

5. Result and Discussion

This section describes the performance of the proposed SaDN algorithm. In the first stage, SaDN has been evaluated on fifteen different single objective CEC 2005 benchmark functions and then for thirty single objective numerical benchmark functions taken from IEEE CEC 2014 during the second stage. In the third or last stage, SaDN performance is evaluated on real-world applications pressure vessel, spring tension and welded beam mechanical design problems. The functions for CEC 2005 analysis for the first phase evaluation are listed in Table 1, different inertia weights for breeder phase are noted in Table 2 and the parameter settings of different algorithms under comparison in Table 3. These algorithms are JADE [21], SaDE [21], GWO-E [23], OEWOA [39], SCCSA [49], FO-FPA [50], CMA-ES [51], SHADE [6], LSHADE-SPACMA [6], EO [6] and basic NMRA [19]. The results on CEC 2005 benchmark functions including the effect of different mutations are presented in Table 4, inertia weights in Table 5, population sizes in Table 6, dimension sizes in Table 7 and among other state-of-art algorithms in Table 8. In case of CEC 2014 analysis, functions are noted in Table 9 whereas results among latest algorithms in Table 10 and Table 11. All the results are compared in terms of mean and standard deviation values for each algorithm and function. Moreover, the statistical tests are also conducted for Wilcoxon rank-sum test [52] and Friedman test [53] with the values of 51 independent runs for all the algorithms under test. The evaluation for each algorithm was done using the 64-bit operating system, Intel Core i7 processor, 32 GB RAM, window 10 and MATLAB version R2016a. In the end, this section provides a detailed summary of the results, drawbacks of the proposed approach, and some insightful implications. A detailed discussion about the experimental results is presented in the consecutive subsections.

Table 1: CEC 2005 Benchmark functions

Function	Dim	Range	Shift position	f_{min}
Unimodal functions				
$F_1(x)= \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	$[-30, -30, .., -30]$	0
$F_2(x)= \sum_{i=1}^{n} \lvert x_i\rvert + \Pi_{i=1}^{n}\lvert x_i\rvert$	100	$[-10, 10]$	$[-3, -3, .., -3]$	0
$F_3(x)= \sum_{i=1}^{n}(\sum_{j-1}^{i} x_j)^2$	30	$[-100, 100]$	$[-3, -3, .., -3]$	0
$F_4(x)= max_i\{\lvert x_i\rvert, 1 \leq i \leq n\}$	100	$[-100, 100]$	$[-3, -3, .., -3]$	0
$F_5(x)= \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_1 - 1)^2$	100	$[-30, 30]$	$[-3, -3, .., -3]$	0
$F_6(x)= \sum_{i=1}^{n}([x_i + 0.5])^2$	100	$[-100, 100]$	$[-3, -3, .., -3]$	0
$F_7(x)= \sum_{i=1}^{n} ix_i^4 = random[0, 1]$	100	$[-1.28, 1.28]$	$[-3, -3, .., -3]$	0
Multimodal functions				
$F_8(x)= \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	$[-30, -30, .., -30]$	0
$F_9(x)= -20exp(-0.2\sqrt{\frac{1}{n}} \sum_{i=1}^{n} x_i^2) - exp(\frac{1}{n} \sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	$[-30, -30, .., -30]$	0
$F_{10}(x)= \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \Pi_{i=1}^{N} cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	$[-30, -30, .., -30]$	0
$F_{11}(x)= \frac{\pi}{n} 10sin(\pi y_1) + \sum_{i=1}^{n} -1(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})]$ $(y_n - 1)^2 + \sum_{i=1}^{u}(x_i, 10, 100, 4)y_i = 1 = \frac{x_i+1}{4}$	100	$[-50, 50]$	$[-30, -30, .., -30]$	0
$F_12(x)= 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2(1 + \sin^2(3\pi x_i + 1)))$ $+0.1((x_n - 1)^2[1 + \sin^2(2\pi x_n]) + \sum_{i=1}^{n} u(x_1, 5, 100, 4)$	100	$[-50, 50]$	$[-30, -30, .., -30]$	0
Fixed dimension functions				
$F_{13}(x)= [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]*$ $[30 + (2x_1 - 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]$		3
$F_{14}(x)= -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[0, 1]$		-3.86
$F_{15}(x)= -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$		-3.86

5.1. Test Suite

The different test functions used to analyze the performance of the proposed SaDN algorithm have been illustrated in Table 1 consists of three major categories: unimodal, multimodal and fixed dimension functions. The unimodal functions $F_1 - F_7$ are used for evaluating SaDN exploitation property to calculate one optimal global solution, whereas the multimodal functions $F_8 - F_{12}$ have a large number of local minima, used to test exploitation as well as exploration properties of SaDN among proposed mutations. The function F_5 (Rosenbrock) is a challenging algorithm to achieve robust performance. The fixed dimension functions $F_{13} - F_{15}$ are involved in testing the consistency and effectiveness of the proposed algorithm to find the best global minimal solution. For functions $F_1, F_3, F_8 - F_{10}$, the dimension size is taken as $D = 30$ and fitness $f_{min} = 0$, whereas in case of $F_2, F_4 - F_7, F_{11}, F_{12}$, the dimension size is $D = 100$ and fitness $f_{min} = 0$. For functions $F_{13} - F_{15}$, the dimension size is $D = 2, 3, 6$ and fitness $f_{min} = 3, -3.86, -3.86$, respectively.

5.2. Parameter Settings

Firstly, the performance of the proposed SaDN has been evaluated with five different mutation strategies applied on scaling factor (F) in the worker phase operation. These results have been compared with the performance to basic NMRA on 15 standard numerical CEC 2005 benchmark functions, as shown in Table 4. The performance of the best mutation is again analyzed and compared with integration to seven different inertia weights applied on a mating factor (λ)

Table 2: Parameter settings and mathematical expressions for different Inertia Weights

Sr. No.	Name of Inertia Weight	Mathematical expression
1.	Exponential Inertia Weight (Exp)	$\lambda= \alpha_{min} + (\alpha_{max} - \alpha_{min}) \times e^{(\frac{-t}{t_{max}/10})}$
		where $\alpha_{min} = \alpha_{max} = rand()$
2.	Random Inertia Weight (Random)	$\lambda=0.5+\frac{rand()}{2}$
3.	Linearly Decreasing Inertia Weight (LD)	$\lambda=\alpha_{max} - \frac{\alpha_{max}-\alpha_{min}}{Maxiter} \times k$
		where $\alpha_{max} = \alpha_{min} = k = rand()$
4.	Simulated Annealing Inertia Weight (SA)	$\lambda_k= \alpha_{min} + (\alpha_{max} - \alpha_{min}) \times p^{k-1}$
		where $\alpha_{min} = 0.5; \alpha_{max} = 0.9; p = 0.95; k = rand()$
5.	Logarithm Decreasing Inertia Weight (Log)	$\lambda= \alpha_{max} + (\alpha_{min} - \alpha_{max}) \times \log_{10}(k + \frac{10t}{t_{max}})$
		where $\alpha_{min} = \alpha_{max} = k = rand()$
6.	Sigmoid Decreasing Inertia Weight (SD)	$\lambda= \frac{\alpha_{min}-\alpha_{max}}{1+e^{(-u(iter-h\times gen))}} + \alpha_{max}$
		where $\alpha_{min} = 0.5; \alpha_{max} = 0.9; k = h = rand(); gen = 51$
		$u = 10^{(\log(gen)-2)}$
7.	Oscillating Inertia Weight (Osci)	$\lambda= \frac{\alpha_{min}+\alpha_{max}}{2} + \frac{\alpha_{max}-\alpha_{min}}{2} \times cos(\frac{2\times pi\times iter}{T})$
		where $\alpha_{min} = 0.5; \alpha_{max} = 0.9; k = h = rand()$
		$T = \frac{2h}{3 + 2k}$

in the breeder phase, as mentioned in Table 5. Therefore, the algorithm based on the best mutation and inertia weight has been selected for further analysis in terms to compare with eleven state-of-the-art algorithms. Such as joint approximate diagonalization of eigen matrices algorithm (JADE), self-adaptive DE (SaDE), extended version of gray wolf optimization (GWO-E), Opposition and exponential whale optimization algorithm (OEWOA), sine cosine crow search algorithm (SCCSA), fractional-order flower pollination algorithm (FO-FPA), success history-based adaptation differential evolution (SHADE), hybridized linear population size reduction SHADE (LSHADE) and semi-parameter adaptation based CMA-ES (LSHADE-SPACMA), covariance matrix adaptation evolution strategy (CMA-ES), equilibrium optimizer (EO) and basic naked-mole-rat algorithm (NMRA). The performance of each of the algorithms is evaluated on the above said 15 benchmark functions. The results are compared in terms of mean and standard deviation, as shown in Table 3. The best results appeared with **bold** font.

Table 3: Parameter Settings

Algorithm	Parameters
JADE	$F = 0.5$; $CR = 0.9$; $1/c = [5, 20]$; $p = [0.05, 0.20]$
SaDE	F, CR = self adaptive
GWO-E	$\vec{\alpha}$ = Linearly decreasing from 2 to 0
OEWOA	$\vec{\alpha}$ = Exponentially decreasing function; $b = 1$
SCCSA	$r_1, r_2, r_3 = [0, 1]$; r_4 determines the sine or cosine movement
FO-FPA	Derivative order$\alpha = [0.1, 1]$; $S = adaptive$; number of historical terms $r = 2$ or 4 or 8
SHADE	$Pbest = 0.1$, $Arcrate = 2$
LSHADE-SPACMA	Learning rate $(c) = 0.8$; threshold=max-nfes/2; $NP = 18D$; $H = 5$; $Pbest = 0.11$
	Arc rate $(c) = 1.4$; Probability variable $(Fcp) = 0.5$; where D=dimension size
CMA-ES	$n = \mu = 10$; number of off-springs $\lambda = 40$
EO	$\alpha_1 = 2$; $\alpha_2 = 1$; Generation Probability $(GP) = 0.5$
NMRA	$bp = 0.5$; $\lambda = random[0, 1]$; number of breeders $B = population(n)/5$
GWO	$\vec{\alpha}$ = Linearly decreasing from 2 to 0
MSSA	c_1, c_2, c_3 = adaptive
CS	pa=0.25; $Tol = 1.0e - 05$; $\beta = 1.5$
FDO	weight factor $(W_f) = 0$; r is random [-1 1]
jDE100	maxFEs = 1e12; agelimit=1e9; sNP=25; bNP=1000; CR=[0.5, 1.1]; F=[0.5, 1.1]
EHOI	Numclan (nClan)=5; alpha=0.5; beta=0.1; number of elite=2; Jump Rate=0.3; a=2;
	r1 is linearly decreased from 2 to 0; r2=(2 $\times$ π) $\times$ rand(); r3=2 $\times$ rand(); r4=rand()
SaDN	$bp = 0.05$; λ = sigmoid decreasing inertia weight; CR=self adaptive; $\vec{r} = [0, 1]$

In the JADE algorithm, two new parameters: c and p have been added in addition to F and CR to improve its performance compared to the basic DE algorithm. The c parameter controls the rate of parameter adaptation, which is taken as in the range of $1/c = [5, 20]$. At the same time, p indicates the greediness of the mutation strategy and is taken $p = [0.05 - 0.2]$ [21]. Similarly, the SaDE performance also depends upon the crossover rate (CR) and scaling factor (F). Here, these parameters are taken as adaptation parameters [21]. In case of GWO-ES, it is found that the parameter $\vec{\alpha}$ acts as a deciding factor to control the exploration and exploitation ability. The values are linearly decreasing numbers in the range of 2 to 0, while the rest of the numbers are simply random numbers [23]. But in OEWOA, the parameter $\vec{\alpha}$ is taken as an exponentially decreasing number instead of a linearly decreasing number, and the rest of the parameters are the same as the basic version of WOA [39]. For SCCSA, r_1, r_2, r_3 and r_4 are the algorithm controlled parameters in which r_1 determines update direction, r_2 indicates updating distance and r_3 chooses emphasis or de-emphasis in the desalination and all these parameters are taken as random numbers $[0, 1]$ whereas r_4 chooses a sine or cosine movement [49]. The FO-FPA performance is in the hands of parameter: derivative order $\alpha = 0.4$ is the best rank [50]. Additionally, the parameters of SHADE, LSHADE-SPACMA are almost similar. The value of P_{best} is taken as lower value $[0.05 - 0.5]$ and $Arcrate$ as higher value $[1, 5]$ to achieve better exploration as well as exploitation ability of the algorithm [6]. Controlled agents of the CMA-ES are taken from [51]. For EO, α_1, α_2 and generation probability (GP) are the performance controlled parameters. α_1 is used to control exploration quantity as taking a higher value of α_1; the exploration ability increases. α_2 is a similar parameter to α_1, but it is used to control EO's exploitation property and its value is taken as lowest as possible. Here, its value is $\alpha_2 = 1$. The value of $GP = 1$ means that no generation rate term participating in the optimization process, whereas $GP = 0$ means it is always participating. In both conditions, EO's performance has degraded due to local optima stagnation, so that $GP = 0.5$ is observed as the best value [6]. For basic NMRA, performance can be adjusted by taking different values of breeding probability (bp). It is used to increase or decrease the number of breeder solutions, which will improve the NMRA performance in terms of enhancing the convergence rate and the exploitation ability of the breeder phase. After careful investigation, its value is selected as $bp = 0.05$. Although, the number of breeders is taken as $1/5^{th}$ of population size [19]. Finally, parameter selection aims to improve the algorithm's ability to obtain the best optimal solution with fast convergence compared to other algorithms. Apart from these algorithms, the common parameter values are taken as dimension size (D) 30, a population size of 60 and 51 NMR runs with 500 maximum iterations.

5.3. Parametric study

This subsection provides a detailed analysis of critical parameters such as the mutation scaling factor (F) of the DE algorithm and the mating factor (λ) of NMRA. These parameters effect has been examined on SaDN under consideration of other settings as $bp = 0.05$ and $CR = adaptive$ and explanation is briefly described in the following subsections:

5.3.1. Analysis of parameter dependence on DE

It is the first stage of analysis, the performance of SaDN among five different mutations such as Lévy mutation (L), Cauchy based mutation (C), neighborhood-based mutation (NB), trigonometric mutation (T) and diversity mutation (Div) to examine the effect of DE algorithm scaling factor on the worker phase of proposed SaDN. For instance, its main focus is on increasing the exploration ability of the algorithm. The results have been evaluated by taking dimension size of 10, population size 30, maximum no. of iterations is 500 with 51 independent NMR trails for each of 15 standard CEC 2005 benchmark functions. The performance is compared with basic NMRA in terms of mean and standard deviation, clearly illustrated in Table 4.

For functions F_1-F_3, F_5-F_7, F_{11}, F_{14}-F_{15}, $SaDN_L$ performs better as compared to all other variants in which it reaches zero standard deviation for F_1 and F_3 functions. In case of a function F_4, $SaDN_C$ provides good results among other algorithms and for functions F_{12} and F_{13}, results for all the algorithms were competitive and $SaDN_{NB}$ was found to be the best mutation. In the same way, it has been observed that all the mutations have the same performance for F_8 and F_{10} functions and all were able to reach the final global optimal solution. Finally, $SaDN_L$ is found to be better for nine functions, $SaDN_C$ for one function, $SaDN_{NB}$ for two functions, and all have the same performance for three functions. So overall, $SaDN_L$ is found to best among all other proposed mutations. Moreover, it can also be verified from the Friedman test, which is a non-parametric in nature. This test is used to evaluate the performance of the algorithm statistically by comparing the different algorithms. The rank is assigned concerning their performance in descending order, i.e. any variant with lower f-value is considered the best variant. Here, Lévy mutation-based SaDN ($SaDN_L$) algorithm is found to the best with a minimum f-value of 24 as the first rank.

5.3.2. Analysis of parameter dependence on NMRA

After findings of Lévy mutation as the best mutation, now in the second stage, the performance of SaDN has been evaluated among seven different inertia weights (IW) strategy. Such as uniform random distribution (*Random*), linearly decreasing initialization weights (*LD*), exponential decreasing distributed operator (*Exp*), oscillating inertia weight (*Osci*), simulated annealing based mutation (*SA*), a sigmoidal decreasing operator (*SD*) and logarithmic mutation (*log*) to examine the effect of NMRA mating factor (λ) in the breeder phase of the proposed SaDN, are used to explore both the exploration as well as the exploitation ability of SaDN. The results have been evaluated by taking a dimension size of 10, population size 30 and 51 NMR runs with 500 maximum iterations on 15 numerical benchmark functions listed in Table 1. The performance is compared with basic NMRA in terms of mean and standard deviation as noted in Table 5.
For functions F_1-F_5, F_7, F_{12} and F_{15}, SD inertia weight has better performance among all the other variants, it was able to reach zero fitness (zero mean and std) for F_1-F_4 functions. All the variants have competitive results for F_6 function, and there is not an easy to say which one is the best variant, so overall, LD IW is found to be best based on std. Similarly, for functions F_{11} and F_{14}, oscillating IW was able to reach an optimal global solution and have good results along with mean value. In case of functions F_8-F_{10}, all the proposed variants performed equally and were able to attain zero fitness. For function F_{13}, Logarithmic decreasing (Log) IW has better mean and std and is considered as a good strategy. Finally, it is concluded that the SD IW is the best variant for eight functions, LD for one, Osci for two, Logarithmic for one, and the same performance for three functions. Moreover, SD IW also got first f-rank with a minimum weight of f-value as 28. Overall, Sigmoidal decreasing (SD) IW strategy is best among all other proposed variants.

Table 4: Parameter analysis of SaDN based on DE scaling factor subjected to different mutation operators

Simulation results for different mutation operators in Worker phase						
Function		$SaDN_L$	$SaDN_{Div}$	$SaDN_C$	$SaDN_T$	$SaDN_{NB}$
F_1	Mean	**5.77E-264**	9.50E-259	4.14E-257	3.10E-253	5.23E-263
	Std	**0**	0	0	0	0
	rank	1	3	4	5	2
F_2	Mean	**6.23E-131**	4.26E-130	2.32E-130	5.07E-130	3.19E-130
	Std	**3.60E-130**	3.01E-129	1.15E-129	2.88E-129	2.10E-129
	rank	1	2	5	3	4
F_3	Mean	**2.51E-265**	1.64E-260	5.58E-252	9.16E-262	1.11E-259
	Std	**0**	0	0	0	0
	rank	1	3	5	2	4
F_4	Mean	8.51E-128	1.34E-130	**2.36E-131**	6.44E-131	2.54E-130
	Std	6.04E-127	7.13E-130	**7.68E-131**	3.36E-130	1.20E-129
	rank	5	2	1	3	4
F_5	Mean	**9.89E+01**	9.89E+01	9.89E+01	9.89E+01	9.89E+01
	Std	**1.82E-03**	1.89E-02	1.96E-02	1.87E-02	2.89E-02
	rank	1	4	3	5	2
F_6	Mean	**2.42E+01**	2.42E+01	2.43E+01	2.43E+01	2.41E+01
	Std	**9.64E-01**	7.25E-01	5.02E-01	5.33E-01	6.23E-01
	rank	1	2	5	4	3
F_7	Mean	**2.81E-04**	3.05E-04	3.04E-04	3.20E-04	1.33E-04
	Std	**3.34E-04**	2.72E-04	2.50E-04	2.46E-04	1.60E-04
	rank	1	2	3	4	5
F_8	Mean	**0**	0	0	0	0
	Std	**0**	0	0	0	0
	rank	1	1	1	1	1
F_9	Mean	**8.88E-16**	8.88E-16	8.88E-16	8.88E-16	8.88E-16
	Std	**0**	0	0	0	0
	rank	1	1	1	1	1
F_{10}	Mean	**0**	0	0	0	0
	Std	**0**	0	0	0	0
	rank	1	1	1	1	1
F_{11}	Mean	**1.20E+00**	1.22E+00	1.21E+00	1.22E+00	1.20E+00
	Std	**8.94E-02**	7.97E-02	8.85E-02	7.44E-02	8.19E-02
	rank	1	4	2	5	3
F_{12}	Mean	9.99E+00	9.99E+00	9.99E+00	9.99E+00	**9.99E+00**
	Std	2.20E-03	2.00E-03	2.40E-03	2.60E-03	**3.00E-03**
	rank	4	5	3	2	1
F_{13}	Mean	1.69E+01	1.64E+01	1.47E+01	1.65E+01	**1.15E+01**
	Std	1.20E+01	1.30E+01	1.07E+01	1.20E+01	**8.86E+00**
	rank	3	4	2	3	1
F_{14}	Mean	**-3.44E+00**	-3.42E+00	-3.47E+00	-3.50E+00	-3.57E+00
	Std	**5.95E-01**	3.03E-01	3.02E-01	2.43E-01	2.36E-01
	rank	1	2	3	4	5
F_{15}	Mean	**-1.97E+00**	-1.92E+00	-1.90E+00	-1.84E+00	-2.14E+00
	Std	**4.66E-01**	4.03E-01	3.99E-01	4.12E-01	3.95E-01
	rank	1	3	4	2	5
Overall f-value		24	39	43	45	42
Overall f-rank		1	2	4	5	3

Table 5: Parameter analysis of SaDN based on NMRA mating factor subjected to different inertia weights

Simulation results for different cases of parameter (λ) in Breeder phase								
Function		λ_{SA}	λ_{Exp}	λ_{LD}	λ_{SD}	λ_{Osci}	λ_{Random}	λ_{Log}
F_1	Mean	6.57E-286	3.16E-178	4.07E-178	**0**	2.44E-247	5.16E-261	4.49E-182
	Std	0	0	0	**0**	0	0	0
	rank	2	6	7	1	4	3	5
F_2	Mean	2.67E-143	9.25E-90	5.52E-90	**0**	1.71E-126	5.68E-130	1.54E-90
	Std	1.89E-142	3.62E-89	2.13E-89	**0**	8.63E-126	2.66E-129	7.11E-90
	rank	2	6	7	1	4	3	5
F_3	Mean	2.89E-287	3.19E-177	9.76E-172	**0**	7.40E-248	3.32E-258	1.63E-183
	Std	0	0	0	**0**	0	0	0
	rank	2	6	7	1	4	3	5
F_4	Mean	7.50E+01	1.56E-90	1.45E-87	**0**	7.61E-125	2.97E-129	4.07E-90
	Std	3.97E+01	4.94E-90	1.03E-86	**0**	5.02E-124	2.19E-128	2.64E-89
	rank	7	4	6	1	3	2	5
F_5	Mean	8.91E+08	9.89E+01	9.89E+01	**9.89E+01**	9.89E+01	9.89E+01	9.89E+01
	Std	4.79E+08	2.05E-02	1.70E-02	**3.08E-02**	1.28E-02	2.11E-02	2.52E-02
	rank	7	5	6	1	3	4	2
F_6	Mean	1.46E+05	2.41E+01	**2.40E+01**	2.45E+01	2.43E+01	2.42E+01	2.43E+01
	Std	1.34E+05	5.99E-01	**7.76E-01**	3.49E-01	5.22E-01	5.68E-01	6.26E-01
	rank	7	3	1	6	5	4	2
F_7	Mean	1.55E+03	2.48E-04	3.37E-04	**2.86E-04**	3.93E-04	2.79E-04	2.87E-04
	Std	6.94E+02	2.58E-04	3.06E-04	**5.38E-04**	3.84E-04	2.24E-04	2.49E-04
	rank	7	4	3	1	2	6	5
F_8	Mean	0	0	0	**0**	0	0	0
	Std	0	0	0	**0**	0	0	0
	rank	1	1	1	1	1	1	1
F_9	Mean	8.88E-16	8.88E-16	8.88E-16	**8.88E-16**	8.88E-16	8.88E-16	8.88E-16
	Std	0	0	0	**0**	0	0	0
	rank	1	1	1	1	1	1	1
F_{10}	Mean	0	0	0	**0**	0	0	0
	Std	0	0	0	**0**	0	0	0
	rank	1	1	1	1	1	1	1
F_{11}	Mean	2.29E+09	1.19E+00	1.18E+00	1.23E+00	**1.18E+00**	1.22E+00	1.20E+00
	Std	1.02E+09	6.83E-02	7.33E-02	7.15E-02	**8.92E-02**	8.00E-02	6.48E-02
	rank	7	5	3	4	1	2	6
F_{12}	Mean	3.74E+09	9.99E+00	9.99E+00	**9.99E+00**	9.99E+00	9.99E+00	9.99E+00
	Std	2.35E+09	2.00E-03	2.90E-03	**3.40E-03**	2.80E-03	2.40E-03	2.70E-03
	rank	6	7	2	1	3	5	4
F_{13}	Mean	2.44E+01	1.47E+01	1.03+01	1.87E+01	1.82E+01	1.81E+01	**1.07E+01**
	Std	1.97E+01	1.16E+01	1.17+01	1.43E+01	1.46E+01	1.62E+01	**9.12E+00**
	rank	7	2	3	4	5	6	1
F_{14}	Mean	-3.48E+00	-3.50E+00	-3.50E+00	-3.45E+00	**-3.46E+00**	-3.43E+00	-3.43E+00
	Std	2.18E-01	2.19E-01	2.63E-01	2.79E-01	**3.12E-01**	3.09E-01	2.17E-01
	rank	6	5	4	3	1	2	7
F_{15}	Mean	-1.73E+00	-2.09E+00	-2.00E+00	**-1.95E+00**	-1.89E+00	-1.90E+00	-2.14E+00
	Std	4.16E-01	4.89E-01	4.54E-01	**5.10E-01**	4.89E-01	4.29E-01	4.90E-01
	rank	6	3	4	1	3	5	2
Overall f-value		69	59	56	28	41	48	52
Overall f-rank		7	6	5	1	2	3	4

5.4. Effect of population Size

After careful investigation of parameters effect, Lévy mutated scaling and sigmoidal decreasing mating factors are observed as the best variants. In the third stage, the impact of population size for the proposed SaDN is compared to DE and basic NMRA has been investigated. For this, four different sets of population sizes ($25, 50, 75$ and 100) have been used. The total number of iterations has taken as 500, whereas all other parameters are same as discussed in the above subsections. The results for all the population sizes are listed in Table 6 and are presented in terms of mean and standard deviation values over 51 NMR runs for all the algorithms under consideration. The results discussion is as follows:

Population size 25: For this case, it has been found that for functions F_1-F_7, F_9-F_{12} and F_{15}, SaDN was able to reach a near-global optimal solution out of which for F_1-F_4 and F_9, it reaches to a final solution of zero value with zero mean and std. For functions F_{13} and F_{14}, all the algorithms have comparative results, but overall, DE is found to be the best. For the function, F_8, both of the SaDN and NMRA algorithms were able to obtain a near-optimum solution, which means both algorithms perform equally well for this function. Finally, SaDN has been observed as a better variant for 12 functions, DE for 2 and the same performance for one function. Hence, SaDN is found to be the best among compared algorithms for a population size of 20.

Population size 50: The simulation results for a population size of 50 are given in Table 6 and it can be observed that for functions F_1-F_7, F_9-F_{12} and F_{15}, SaDN provided best results both in terms of mean and standard deviation and is the best algorithm. For functions F_{13} and F_{14}, DE performs better among other variants and has better results, whereas in case of a function F_8, SaDN and NMRA both of the algorithms have the same performance in which mean and standard deviation reaches to zero value. The overall conclusion is that SaDN is better for 13 functions and DE for 2 functions, among other algorithms for a population size of 50.

Population size 75: It can be seen that for functions F_1-F_{12}, the results of SaDN are better as compared to other variants, whereas for functions F_{13} and F_{14}, SaDN failed to reach the optimal global solution and overall DE is found to be the best. In case of function F_{15}, the results of all the algorithms are comparable in terms of standard deviation where basic NMRA provides the best results. Overall, it is concluded that for a population size of 60, SaDN is best for most of the functions.

Population size 100: From simulated results with a population size of 100, it can be seen that for functions F_1-F_{11}, SaDN has performed better than other variants.For functions F_{13} and F_{14}, SaDN failed to reach an optimal global solution and overall, DE is found to be the best algorithm. In case of functions F_{12} and F_{15}, the results of all the algorithms are comparable in terms of standard deviation, from which basic NMRA is observed as the best algorithm. Overall, it is concluded that SaDN has better results for 11 functions, DE for 2 and NMRA for 2 functions. Hence, SaDN is found to be best for a population size of 100.

Inferences from effect of population size: From the results, it has been observed that for variable population size, with an increase in the population size, the results are getting better. For the population sizes of 25 and 50, there is less variation in the results. But as the population size increases to 75 or 100, the overall performance degrades. Beyond the population size of 50, there is not much improvement in the simulation results. Here it should be noted that with an increase in population sizes, the total number of function evaluations increases many folds. So overall, we can say that SaDN performs best for a population size of 50.

5.5. Effect of dimension Size

This section describes the effect of dimension sizes for the proposed SaDN among DE and basic NMRA algorithms. In this case, five different set of dimension sizes ($30, 50, 100, 500$ and 1000) have been evaluated over 12 different CEC 2005 functions. The parameters for simulation are taken as the population size of 50, maximum no. of iterations as 500 with 51 NMR runs. Results for all the dimension sizes based on mean and standard deviation values are described in Table 7 and their description is given as below:

Dimension size 30: The simulation results for a dimension size of 30 are illustrated in Table 7. It has been observed that for functions F_1-F_4, F_7-F_{10} and F_{12}, SaDN was capable of attaining near-optimal global solution in which SaDN approaches to zero fitness for functions F_5, F_6 and F_{11}, basic NMRA is found to be best among all other variants. Overall, SaDN performs better for 9 functions and NMRA for 3. Hence, SaDN is found to be the best among all diverse variants for a dimension size of 30.

Dimension size 50: The simulation results with 50 dimension size are described in this subsection. For functions F_1-F_{11}, all the algorithms have comparative results. But overall, SaDN is found to be best among other algorithms. In contrast, for function F_{12}, NMRA was able to reach the global optimum solution and gives the best performance in terms of standard deviation only. Finally, SaDN has good results for 11 functions and NMRA for 1 function. Hence, all the algorithms failed to compete with SaDN, which is found to be best compared to other algorithms.

Table 6: Experimental results for population size of $25, 50, 75, 100$

Function	Algorithm	Pop Size 25		Pop Size 50		Pop Size 75		Pop Size 100	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
	DE	6.69E+04	1.15E+04	6.25E+04	1.02E+04	6.17E+04	7.40E+03	6.00E+04	6.85E+03
F_1	NMRA	6.89E-31	4.91E-30	7.72E-29	5.51E-28	2.17E-21	1.54E-20	3.09E-05	2.21E-04
	SaDN	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	DE	1.97E+51	1.12E+52	1.97E+48	1.06E+49	3.63E+47	2.18E+48	3.30E+47	1.55E+48
F_2	NMRA	1.76E-17	1.20E-16	5.17E-16	2.76E-15	2.20E-03	1.10E-02	1.10E-03	6.30E-03
	SaDN	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	DE	7.54E+04	3.92E+04	9.19E+04	3.35E+04	1.01E+05	2.84E+04	8.93E+04	2.11E+04
F_3	NMRA	2.32E-25	1.62E-24	5.40E-26	3.85E-25	7.71E-05	5.50E-04	1.19E-04	8.53E-04
	SaDN	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	DE	9.60E+01	1.19E+00	9.57E+01	1.18E+00	9.49E+01	1.32E+00	9.51E+01	1.08E+00
F_4	NMRA	1.69E-13	1.20E-12	1.44E-16	7.24E-16	4.10E-17	2.35E-16	2.90E-04	2.10E-03
	SaDN	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	DE	1.18E+09	9.83E+07	1.16E+09	7.97E+07	1.12E+09	7.55E+07	1.09E+09	8.49E+07
F_5	NMRA	9.89E+01	1.27E-02	9.89E+01	1.38E-02	9.89E+01	1.64E-02	9.89E+01	1.85E-02
	SaDN	9.89E+01	1.49E-02	9.89E+01	**3.01E-02**	9.89E+01	1.85E-02	9.89E+01	2.59E-02
	DE	2.79E+05	1.46E+04	2.66E+05	1.26E+04	2.65E+05	1.19E+04	2.63E+05	1.17E+04
F_6	NMRA	2.46E+01	5.10E-01	2.46E+01	5.43E-01	2.40E+01	5.12E-01	2.38E+01	5.31E-01
	SaDN	2.45E+01	5.15E-01	2.45E+01	**6.18E-01**	2.44E+01	4.31E-01	2.42E+01	5.35E-01
	DE	1.89E+03	1.87E+02	1.84E+03	1.74E+02	1.77E+03	1.62E+02	1.75E+03	1.36E+02
F_7	NMRA	5.40E-03	6.00E-03	3.40E-03	3.50E-03	2.80E-03	2.50E-03	2.90E-03	2.70E-03
	SaDN	8.61E-04	**6.98E-04**	2.62E-04	2.30E-04	1.66E-04	1.69E-04	1.14E-04	1.58E-04
	DE	4.18E+02	6.46E+01	4.23E+02	3.86E+01	4.22E+02	2.85E+01	4.14E+02	2.46E+01
F_8	NMRA	**0**	**0**	**0**	**0**	1.37E-04	9.83E-04	3.23E-05	2.31E-04
	SaDN	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	DE	2.03E+01	5.84E-01	2.04E+01	2.78E-01	2.04E+01	3.28E-01	2.04E+01	2.41E-01
F_9	NMRA	1.16E-15	1.56E-15	8.88E-16	**0**	2.53E-14	1.40E-13	8.01E-05	5.72E-04
	SaDN	8.88E-16	**0**	8.88E-16	**0**	8.88E-16	**0**	8.88E-16	**0**
	DE	1.15E-01	1.53E-01	1.86E-02	4.78E-02	9.74E-10	6.95E-09	**0**	**0**
F_{10}	NMRA	4.77E-05	3.40E-05	**0**	**0**	4.61E-06	1.95E-05	9.60E-06	3.69E-05
	SaDN	**0**	**0**	**0**	**0**	**0**	**0**	3.00E-03	2.13E-02
	DE	2.86E+09	3.32E+08	2.75E+09	2.76E+08	2.64E+09	2.46E+08	2.52E+09	2.80E+08
F_{11}	NMRA	1.21E+00	1.32E-01	1.26E+00	7.22E-02	9.10E-01	7.42E-02	8.88E-01	7.87E-02
	SaDN	1.25E+00	6.09E-02	1.22E+00	**9.25E-02**	1.22E+00	7.73E-02	1.19E+00	8.77E-02
	DE	5.31E+09	4.47E+08	5.05E+09	4.42E+08	4.97E+09	4.24E+08	4.80E+09	3.64E+08
F_{12}	NMRA	9.99E+00	1.60E-03	9.99E+00	1.20E-03	9.99E+00	1.62E-02	9.99E+00	2.10E-03
	SaDN	9.99E+00	1.70E-03	9.99E+00	**2.30E-03**	9.99E+00	**2.30E-03**	1.01E+01	8.17E-01
	DE	3.00E+00	5.68E-05	3.00E+00	2.65E-15	3.00E+00	**3.26E-15**	3.00E+00	3.22E-15
F_{13}	NMRA	3.37E+00	2.34E+00	3.00E+00	2.19E-02	3.00E+00	3.96E-04	3.00E+00	8.78E-05
	SaDN	1.86E+01	2.35E+01	2.37E+01	1.48E+01	1.60E+01	1.53E+01	1.56E+01	1.63E+01
	DE	-3.86E+00	1.27E-04	-3.86E+00	3.03E-15	-3.86E+00	3.06E-15	-3.86E+00	**3.10E-15**
F_{14}	NMRA	-3.85E+00	2.04E-02	-3.86E+00	6.80E-03	-3.86E+00	1.40E-03	-3.86E+00	1.10E-03
	SaDN	-3.28E+00	4.79E-01	-3.46E+00	2.38E-01	-3.54E+00	1.96E-01	-3.60E+00	1.69E-01
	DE	-3.19E+00	1.64E-01	-3.21E+00	3.88E-02	-3.21E+00	3.83E-02	-3.21E+00	3.22E-02
F_{15}	NMRA	-3.21E+00	1.20E-01	-3.25E+00	4.82E-02	-3.28E+00	4.14E-02	-3.29E+00	4.15E-02
	SaDN	-1.46E+00	4.79E-01	-1.93E+00	**6.69E-02**	-2.02E+00	3.75E-01	-2.24E+00	3.78E-01

Dimension size 100: For functions, F_1-F_4, F_6-F_{10} and F_{12}, all the algorithms have comparative results, but overall, SaDN is found to be best among other algorithms. For functions F_5, F_{11}, NMRA was able to reach global optimum solution and gives the best performance. In conclusion, SaDN works efficiently for 10 functions and NMRA for 2 functions. Hence SaDN is found to be best as along with other variants.

Dimension size 500: In functions F_1, F_5, F_6 and F_{10}, all the algorithms have competitive results. It is tough to say which one is the best variant. However, based on standard deviation, SaDN is found to be the best. In case of functions F_1-F_{12}, the results of all the algorithms were poor, not up to a satisfactory level. Only SaDN performs better as compared to other variants. Finally, SaDN has proved its capability and is identified as the best algorithm.

Dimension size 1000: This subsection discusses the results of dimension size of 1000. For functions F_1-F_4, SaDN was able to obtain the optimal global solution and provides better results among other variants. Similarly, for functions F_8-F_{12}, the performance of basic NMRA and SaDN is better than the DE algorithm. SaDN performed better than basic NMRA except for the function F_{10} because, in F_{10}, SaDN approaches to final zero fitness value. For the rest of the functions F_5-F_7, SaDN was able to reach an optimal global solution. Finally, it is observed that SaDN performs better for all the 12 functions; no other algorithm is capable enough for a population size of 1000. Hence SaDN has again proved itself as the best variant.

Inferences from the effect of dimension size: Any algorithm's performance is said to be better when it provides excellent results for higher dimension sizes. Here, SaDN provided better results for dimension size larger than 200 as compared to DE and basic NMRA. But, as we move to higher dimension sizes,the performance of all algorithms degrade due to increase in computational complexity.Therefore, SaDN performance also degrades for dimension size of 500 and 1000 as compared to 30, 50 and 100 as can be seen from Table 7. Overall, SaDN is best for a maximum number of functions in all sets of dimension sizes.

5.6. Comparison with respect to other algorithms

5.6.1. Experimental Testing

Here the results are compared in terms of mean and standard deviation values for 7 uni-modal functions, 5 multi-modal functions and 3 fixed dimension functions. The parameters of SaDN are taken as $bp = 0.05$, $CR = adaptive$, population size 50, dimension size of 30, 500 iterations with 51 independent NMR trails. For functions F_1-F_5, F_8-F_{10}, SaDN results reach zero fitness in terms of

Table 7: Experimental results for dimension size of $30, 50, 100, 500, 1000$

Function	Algorithm	Dim Size 30		Dim Size 50		Dim Size 100		Dim Size 500		Dim Size 1000	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F_1	DE	6.54E+04	7.43E+03	1.21E+05	8.69E+03	2.67E+05	1.44E+04	1.51E+06	3.46E+04	3.11E+06	4.22E+04
	NMRA	1.01E-33	7.19E-33	3.66E-34	1.86E-33	3.02E-31	2.16E-30	1.86E+03	4.49E+03	1.40E+03	2.22E+03
	SaDN	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.89E+01	2.77E+02	4.30E-17	3.07E-16
F_2	DE	3.37E+07	2.40E+08	2.56E+18	1.80E+19	6.65E+48	3.24E+49	1.35E+266	Inf	Inf	NaN
	NMRA	2.41E-18	1.49E-17	2.45E-18	1.19E-17	2.40E-18	1.03E-17	5.30E+01	4.48E+01	3.98E+01	3.83E+01
	SaDN	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.53E-24	1.07E-23	3.59E-16	2.56E-15
F_3	DE	8.55E+04	3.05E+04	2.55E+05	7.18E+04	1.05E+06	3.35E+05	2.64E+07	9.56E+06	1.02E+08	3.03E+07
	NMRA	8.51E-32	6.04E-31	1.41E-30	9.02E-30	7.99E-30	5.70E-29	3.08E+05	1.00E+06	1.41E+06	2.90E+06
	SaDN	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.80E-56	2.71E-55	1.15E+03	8.27E+03
F_4	DE	8.61E+01	3.24E+00	9.18E+01	1.97E+00	9.57E+01	1.27E+00	9.90E+01	2.79E-01	9.95E+01	1.38E-01
	NMRA	4.96E-17	2.45E-16	1.70E-16	1.15E-15	2.02E-17	1.25E-16	2.02E+00	1.87E+00	2.16E+00	2.71E+00
	SaDN	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.37E-37	2.40E-36	1.81E-17	1.29E-16
F_5	DE	2.35E+08	3.82E+07	4.74E+08	6.24E+07	1.14E+09	8.83E+07	7.00E+09	2.37E+08	1.46E+10	2.79E+08
	NMRA	2.89E+01	2.07E-02	4.89E+01	1.73E-02	**9.89E+01**	**7.20E-03**	1.93E+04	6.09E+04	5.01E+04	1.83E+05
	SaDN	2.89E+01	1.94E-02	4.89E+01	1.82E-02	9.89E+01	1.43E-02	2.39E+03	1.35E+04	9.98E+02	1.63E+00
F_6	DE	6.46E+04	8.43E+03	1.21E+05	9.00E+03	2.65E+05	1.14E+04	1.51E+06	2.74E+04	3.12E+06	5.46E+04
	NMRA	7.14E+00	4.27E-01	1.21E+01	3.62E-01	2.46E+01	4.75E-01	1.24E+03	2.00E+03	1.49E+03	2.42E+03
	SaDN	6.89E+00	4.02E-01	1.20E+01	3.99E-01	**2.43E+01**	**5.81E-01**	1.11E+02	3.93E+00	2.34E+02	6.18E+00
F_7	DE	1.11E+02	2.25E+01	3.83E+02	5.36E+01	1.83E+03	1.54E+02	5.69E+04	1.73E+03	2.40E+05	4.80E+03
	NMRA	3.50E-03	3.00E-03	3.70E-03	3.40E-03	3.60E-03	3.60E-03	5.56E-01	1.17E+00	1.96E+00	6.07E+00
	SaDN	2.82E-04	2.08E-04	**2.95E-04**	**4.08E-04**	2.76E-04	2.63E-04	4.30E-04	4.54E-04	7.20E-03	4.77E-02
F_8	DE	4.24E+02	4.05E+01	7.70E+02	3.47E+01	1.63E+03	4.77E+01	8.76E+03	1.00E+02	1.78E+04	1.95E+02
	NMRA	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.32E+02	3.88E+02	7.90E+02	1.44E+03
	SaDN	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.08E-08	7.74E-08	5.51E+00	3.90E+01
F_9	DE	2.03E+01	4.02E-01	2.07E+01	1.36E-01	2.09E+01	5.35E-02	2.11E+01	2.24E-02	2.11E+01	1.84E-02
	NMRA	8.88E-16	**0.00E+00**	1.35E-14	9.05E-14	8.88E-16	**0.00E+00**	2.18E+00	1.90E+00	2.68E+00	1.63E+00
	SaDN	8.88E-16	**0.00E+00**	8.88E-16	**0.00E+00**	8.88E-16	**0.00E+00**	1.50E-02	9.78E-02	8.88E-16	**0.00E+00**
F_{10}	DE	1.20E-02	4.10E-02	2.04E-02	5.34E-02	8.90E-03	3.58E-02	1.12E-02	4.11E-02	1.30E-02	4.37E-02
	NMRA	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.99E-02	5.59E-02	4.56E-02	5.63E-02
	SaDN	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	7.60E-03	5.42E-02	3.80E-03	2.74E-02
F_{11}	DE	5.07E+08	1.06E+08	1.12E+09	1.75E+08	2.73E+09	2.77E+08	1.72E+10	5.83E+08	3.63E+10	7.75E+08
	NMRA	1.24E+00	2.84E-01	1.33E+00	1.58E-01	1.25E+00	8.15E-02	1.71E+00	9.48E-01	1.71E+00	8.04E-01
	SaDN	1.17E+00	2.45E-01	1.22E+00	1.65E-01	1.22E+00	6.89E-02	**9.19E-01**	**7.99E-02**	1.01E+00	5.09E-02
F_{12}	DE	1.05E+08	1.78E+08	2.08E+09	2.26E+08	5.10E+09	4.22E+08	3.14E+10	1.06E+09	6.65E+10	1.57E+09
	NMRA	2.99E+00	1.70E-03	4.99E+00	1.40E-03	9.99E+00	2.20E-03	1.19E+02	1.05E+02	2.51E+02	1.97E+02
	SaDN	2.99E+00	2.00E-03	4.99E+00	1.30E-03	**9.99E+00**	**2.60E-03**	4.99E+01	3.56E-02	1.02E+02	1.78E+01

mean and standard deviation. For functions F_6 and F_{11}, all the algorithms have competitive results, but CMA-ES is found to be best in terms of standard deviation. In the case of function F_7, the GWO-E algorithm results were better as compared to other variants. For functions F_{12} and F_{14}, JADE is the only algorithm that attains a value of global optima whereas, for a function F_{13}, SaDE is found to be best based on standard deviation. Similarly, for function $F15$, all the algorithms have comparable results,hence, it is difficult to say which one is the best algorithm. Overall, the proposed SaDN is best for 8 functions, CMA-ES for 2, JADE for 2, SaDE for 1 and OEWOA for 1 function. Hence it can be seen that the proposed SaDN is best for the maximum number of functions.

5.6.2. Statistical Testing

This section examines the statistical performance of the proposed SaDN algorithm using two statistical tests such as Wilcoxon Rank-sum test and Friedman $F - rank$ test. For the Rank-Sum test, two different algorithms are compared in terms of the best samples and give a P-value correspondingly, which determines the statistical significance of that algorithm at 5% level of significance. For the present case, the proposed SaDN algorithm has been compared with some recent algorithms such as SaDN with JADE, SaDN/SaDE, SaDN/GWO-E, SaDN/OEWOA, SaDN/SCCSA, SaDN/FO-FPA, SaDN/SHADE, SaDN/LSHADE-SPACMA, SaDN/CMA-ES, SaDN/EO and SaDN/NMRA to prove the efficiency of SaDN. It is impossible to compare the SaDN with itself so that NA (Not Applicable) or blank has been used in its place. When the algorithm under comparison to proposed SaDN is identical or there is no statistical relevance between them, then '=' sign is inserted in its place, which shows both of the algorithms have the same performance and can not be compared. Similarly, the '+' sign is used if the algorithm compared to SaDN performs better and assigns a '$-$' sign if SaDN is found to be the best among the algorithms under test. The w(win)/l(loss)/t(tie) analysis represents how many times the algorithm under comparison win, loss and have an equal performance as compared to the proposed SaDN, as shown in Table 8. From the table, it has been observed that JADE, SaDE, SCCSA, LSHADE-SPACMA and CMA-ES losses 11 times, OEWOA and SHADE for 10 times, GWO-E, FO-FPA, EO and NMRA for 8 times loses or have the worse performance as compared to SaDN. Hence it is proved that the SaDN performance is significantly better for the maximum number of cases and found to be the best algorithm statistically.

In the second stage, the Friedman $F - rank$ analysis (non-parametric in nature) has been used to test the performance of SaDN. In order to have a reliable comparison, at least 10 benchmark functions with more than five different algorithms are considered [53]. Here this work has considered 15 benchmark functions with 12 different algorithms. As a resultant, Table 8 illustrates that F-rank has been assigned to all the algorithms according to their performance in descending order. Then, add up all the individual ranks of particular algorithms to find the f-value of that algorithm. At last, all the algorithms are sorted in f-rank wise based on their f-values performance. The algorithm with the lowest f-value is considered the best algorithm, whereas the algorithm with higher f-value indicates poor performance. Finally, from these analyses, SaDN, JADE and CMA-ES have significant results among other variants under consideration. But overall, the proposed SaDN is found to be the best algorithm with first ranked, minimum f-value of 58.

5.7. Convergence profiles

This section presents an impact of merging DE exploration and NMRA exploitation within the frame structure of basic NMRA algorithm to enhance the acceleration convergence trends of SaDN, for twelve different CEC 2005 benchmark functions in Figure 2. Generally, the convergence curve exists at a slow rate at the initial iterations of an algorithm due to the requirement of the exploration phase in the starting stage while the exploitation phase is required towards the algorithm end iterations. In the exploitation phase, the algorithm moves faster than the exploration end of the phase, and this enables the algorithm to converge faster towards an optimal solution at the later stages. Here SaDN algorithm proved that it has excellent performance to enhance exploration propensity and maintain a balance between exploration and exploitation. From the inspection of Figure 2, it can be observed that for functions F_1-F_4, F_8 and F_{10}, SaDN fitness reached to a final solution of zero value after the few initial stages while at the same time DE and NMRA algorithms have been failed to obtain zero fitness. Moreover, SaDN exhibits the faster decaying rate for the optimal values for all the given functions except F_7. So overall, the proposed SaDN is found the best convergence trends as compared to DE and NMRA.

5.8. Comparison on CEC2014 benchmark problems

In this part, performance evaluation of the proposed SaDN algorithm has been tested on thirty CEC 2014 benchmark functions [25] and compared the results with respect to some standard Laplacian Biogeography-Based Optimization (Lx-BBO) [54], Blended Biogeography-Based Optimization (B-BBO) [54], Random Walk-Grey Wolf Optimization (RW-GWO) [55], Isomorphic Algorithms (ISOS) [56], Marine Predators Algorithm (MPA) [57], Improved Elephant Herding optimization (IMEHO) [58], Variable Neighborhood Bat Algorithm (VNBA) [58], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [59], Improved Differential Evolution (IDE) [60], Sinusoidal Differential Evolution (SinDE) [61], Beta Differential Evolution (BDE) [62], Memory Guided Sine Cosine

Table 8: Statistical results of proposed algorithm in comparison to other algorithms for CEC 2005

Function		JADE	SaDE	GWO-E	OEWOA	SCCSA	FO-FPA	SHADE	LSHADE	CMA-ES	EO	NMRA	**Proposed**
		[21]	[21]	[23]	[39]	[49]	[50]	[6]	-SPACMA [6]	[6]	[6]		**SaDN**
F_1	mean	1.80E-60	4.50E-20	3.92E-67	7.75E-176	9.22E-69	1.51E-184	1.42E-09	2.23E-01	1.42E-18	3.32E-40	7.72E-29	**0.00E+00**
	std	8.40E-60	6.90E-20	1.11E-66	**0.00E+00**	3.81E-68	**0.00E+00**	3.09E-09	1.48E-01	3.13E-18	6.78E-40	5.51E-28	**0.00E+00**
	P-rank	+	+	+	+	+	+	+	+	+	+	+	
	F-rank	5	8	4	2	3	2	10	11	9	6	7	1
F_2	mean	1.80E-25	1.90E-14	4.31E-36	1.86E-115	8.25E-41	5.04E-93	8.70E-03	2.11E+01	2.98E-07	7.12E-23	5.17E-16	**0.00E+00**
	std	8..8E-25	1.05E-14	6.57E-36	1.32E-114	4.19E-40	3.47E-93	2.13E-02	9.57E+00	1.78E+00	6.36E-23	2.76E-15	**0.00E+00**
	P-rank	+	+	+	+	+	+	+	+	+	+	+	
	F-rank	6	9	5	2	4	3	10	12	11	7	8	1
F_3	mean	5.70E-61	9.00E-37	3.75E-37	2.87E+04	4.31E-13	1.23E-183	1.54E+01	8.87E+01	1.59E-05	8.06E-09	5.40E-26	**0.00E+00**
	std	2.70E-60	5.43E-36	1.36E-36	1.02E+04	2.83E-30	**0.00E+00**	9.94E+00	4.72E+01	2.21E-05	1.60E-08	3.85E-25	**0.00E+00**
	P-rank	+	+	+	+	+	+	+	+	+	+	+	
	F-rank	3	4	5	12	6	2	10	11	9	8	7	1
F_4	mean	8.20E-24	7.40E-11	2.39E-25	1.06E+01	2.15E-17	9.97E-93	9.79E-01	2.11E+00	2.01E-06	5.39E-10	1.44E-16	**0.00E+00**
	std	4.00E-23	1.82E-10	6.80E-25	2.22E+01	1.06E-16	7.31E-93	7.99E-01	4.92E-01	1.25E-06	1.38E-09	7.24E-16	**0.00E+00**
	P-rank	+	+	+	+	+	+	+	+	+	+	+	
	F-rank	4	7	3	12	6	2	10	11	9	8	5	1
F_5	mean	8.00E-02	2.10E+01	2.65E+01	2.85E+01	5.90E+00	2.89E+01	2.44E+01	2.88E+01	3.67E+01	2.53E+01	9.89E+01	9.89E+01
	std	5.60E-01	7.80E+00	5.19E-01	2.22E-01	9.13E-01	1.72E-02	1.12E+01	8.24E-01	3.34E+01	1.69E-01	1.38E-02	**3.01E-02**
	P-rank	+	+	+	+	+	+	+	+	+	+	+	
	F-rank	6	10	7	8	4	2	11	5	12	9	3	1
F_6	mean	2.90E+00	9.30E+02	2.65E+01	1.62E+00	4.14E-08	5.88E+00	5.31E-10	2.48E-01	**6.83E-19**	8.29E-06	2.46E+01	2.45E+01
	std	1.20E+00	1.80E+02	5.19E-01	6.93E-01	5.22E-08	5.86E-01	6.35E-10	1.13E-01	**6.71E-19**	5.02E-06	5.43E-01	4.18E-01
	P-rank	+	+	−	−	−	−	−	+	−	−	−	
	F-rank	11	12	8	5	3	6	2	10	1	4	7	9
F_7	mean	6.40E-04	4.80E-03	**9.90E-05**	1.37E-03	1.33E-03	1.13E-04	2.35E-02	4.70E-03	2.75E-02	1.17E-03	5.40E-03	8.61E-04
	std	2.50E-04	1.20E-03	**8.37E-05**	2.85E-03	1.72E-03	8.94E-04	8.80E-03	1.90E-03	7.90E-03	6.54E-04	6.00E-03	6.98E-04
	P-rank	+	+	−	+	+	−	+	+	+	+	+	
	F-rank	6	12	1	9	11	2	7	10	8	4	5	3
F_8	mean	1.00E-04	1.20E-03	**0.00E+00**	**0.00E+00**	5.46E+00	**0.00E+00**	8.53E+00	6.75E+01	2.53E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	6.00E-05	6.50E-04	**0.00E+00**	**0.00E+00**	5.62E+00	**0.00E+00**	2.19E+00	1.00E+01	8.55E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
	P-rank	+	+	=	=	+	=	+	+	+	=	=	
	F-rank	2	3	1	1	5	1	4	7	6	1	1	1
F_9	mean	8.20E-10	2.70E-03	5.58E-15	3.02E-15	8.88E-16	8.88E-16	3.95E-01	3.93E-02	1.55E+01	8.34E-14	8.88E-16	**8.88E-16**
	std	6.90E-10	5.10E-04	1.67E-15	2.27E-15	9.36E-32	**0.00E+00**	5.86E-01	1.51E-02	7.92E+00	2.53E-14	**0.00E+00**	**0.00E+00**
	P-rank	+	+	+	+	+	=	+	+	+	+	=	
	F-rank	6	7	4	3	2	1	9	8	10	5	1	1
F_{10}	mean	9.90E-08	7.80E-04	**0.00E+00**	1.42E-02	3.33E-02	**0.00E+00**	4.80E-03	8.94E-01	5.76E-15	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	6.00E-07	1.20E-03	**0.00E+00**	1.00E-01	4.56E-02	**0.00E+00**	7.70E-03	1.07E-01	6.18E-15	**0.00E+00**	**0.00E+00**	**0.00E+00**
	P-rank	+	+	=	+	+	=	+	+	+	=	=	
	F-rank	3	5	1	8	6	1	4	7	2	1	1	1
F_{11}	mean	4.60E-17	1.90E-05	1.98E-02	1.06E-01	1.34E-02	8.32E-01	3.46E-02	8.18E-04	**2.87E-16**	7.97E-07	1.26E+00	1.22E+00
	std	1.90E-16	9.20E-06	1.01E-02	4.97E-02	1.60E-02	1.78E-01	8.75E-02	1.00E-03	**5.64E-16**	7.69E-07	7.22E-02	9.25E-02
	P-rank	−	−	+	+	+	+	+	−	−	−	+	
	F-rank	2	4	11	9	10	12	7	5	1	3	8	6
F_{12}	mean	**2.00E-16**	6.10E-05	2.50E-01	1.03E+00	2.01E-02	2.94E+00	7.32E-04	1.02E-02	3.66E-04	2.92E-02	9.99E+00	9.99E+00
	std	**6.50E-16**	2.00E-05	1.63E-01	3.61E-01	7.23E-02	1.59E-01	2.80E-03	1.03E-02	2.00E-03	3.52E-02	1.20E-03	2.30E-03
	P-rank	−	−	+	+	+	+	−	+	+	+	+	
	F-rank	1	2	11	10	7	12	3	9	5	8	6	4
F_{13}	mean	3.00E+00	**3.00E+00**	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	8.40E+00	3.00E+00	3.00E+00	2.37E+01
	std	1.10E-15	**3.00E-15**	1.25E-05	4.96E-04	8.93E-05	3.13E-09	1.87E-15	1.25E-15	2.05E+01	1.56E-15	2.19E-02	1.48E+01
	P-rank	−	−	−	−	−	−	−	−	+	−	−	
	F-rank	5	1	8	9	7	6	2	4	12	3	10	11
F_{14}	mean	**-3.86E+00**	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.01E-01	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.46E+00
	std	**0**	3.10E-15	4.16E-06	2.92E-04	9.29E-06	2.25E-16	2.70E-15	2.70E-15	2.70E-15	6.80E-03	9.45E-08	2.38E-01
	P-rank	−	−	−	−	−	−	−	−	−	−	−	
	F-rank	1	3	7	8	6	2	4	4	4	9	5	10
F_{15}	mean	-3.31E+00	-3.31E+00	-3.26E+00	**-3.24E+00**	-3.27E+00	-3.29E+00	-3.27E+00	-3.28E+00	-3.29E+00	-3.26E+00	-3.25E+00	-1.93E+00
	std	3.60E-02	2.80E-02	7.50E-02	**8.18E-02**	6.00E-02	1.97E-02	6.00E-02	5.70E-02	5.35E-02	5.70E-02	6.69E-02	4.28E-02
	P-rank	+	+	−	−	−	+	−	−	−	−	−	
	F-rank	8	9	2	1	4	10	4	5	6	5	3	7
w/l/t		11/4/0	11/4/0	8/5/2	10/4/1	11/4/0	8/4/3	10/5/0	11/4/0	11/4/0	8/5/2	8/4/3	NA
Overall F-value		69	96	78	99	84	64	97	119	105	81	77	58
Overall F-rank		3	8	5	10	7	2	9	12	11	6	4	1

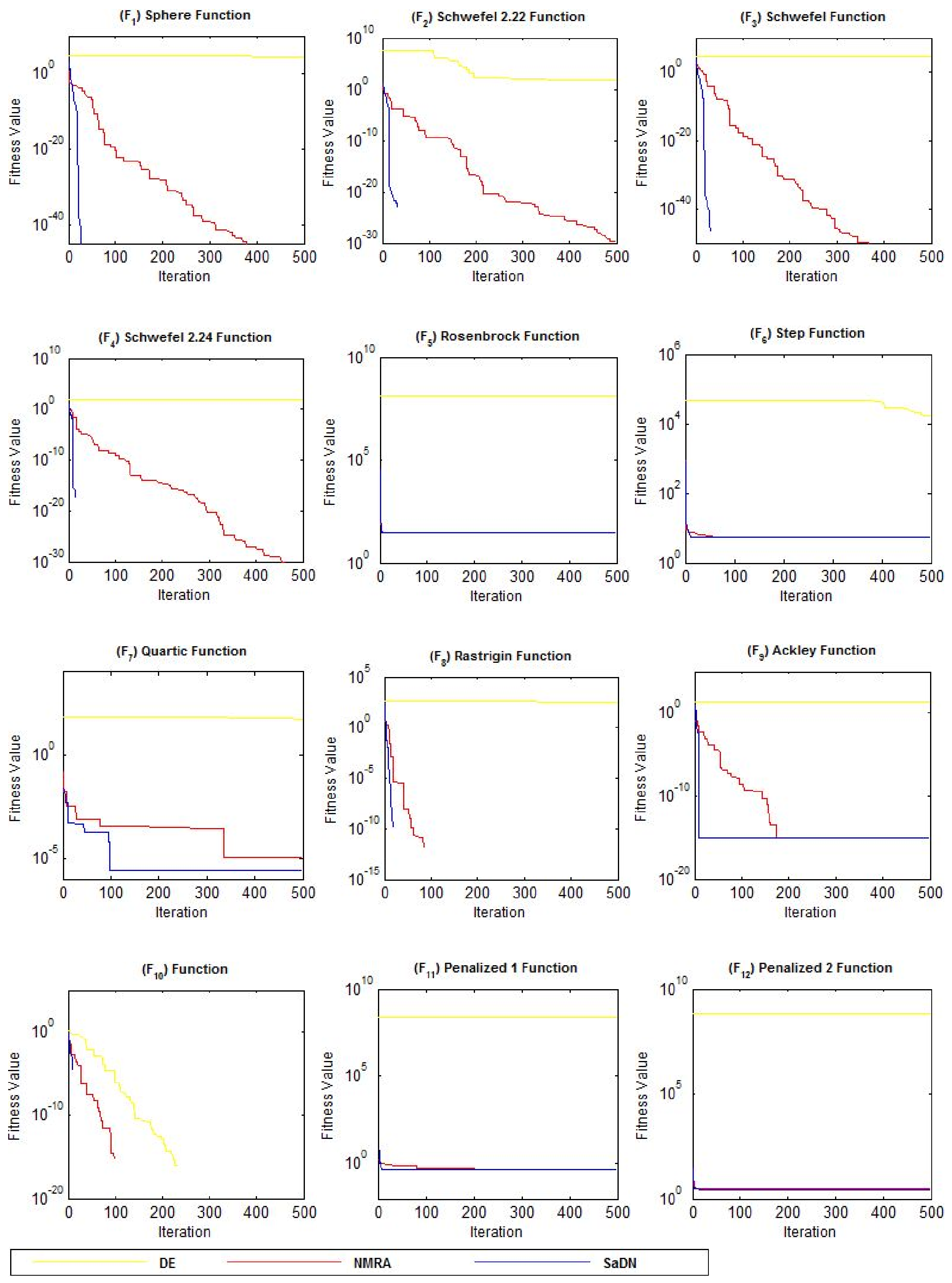

Figure 2: SaDN convergence profiles of different functions

Algorithm (MG-SCA) [63] and basic Naked Mole-rat Algorithm (NMRA). These functions are divided into four categories as: 3 unimodal functions ($F_1 - F_3$), 13 multimodal functions ($F_4 - F_{16}$), 6 hybrid functions ($F_{17} - F_{22}$) and 8 composition functions ($F_{23} - F_{30}$), which are well explained in Table 9. During the simulation, the dimension size of SaDN is taken as 30, the population kept as 50 with 30000 maximum number of iterations. The simulation performance is noted for each of the 51 NMR runs. Apart from this, the simulated results for all the algorithms are compared in terms of mean and standard deviation as shown in Table 10 for functions $F_1 - F_{15}$ and in Table 11 for $F_{16} - F_{30}$.
From these analyses, it is observed that for unimodal functions F_1, F_2 and F_3, CMA-ES, SinDE and IDE algorithms have been performed well and found to be best among other variants. In case of multimodal F_4-F_{16} functions, SaDN is found to be the best for all multimodal functions except the function F_4, F_7 and F_8. Here, SaDN comes out as a very effective algorithm because of the inserted DE structure to NMRA frame enhances the exploration propensity in addition to an already existing strong exploitation of NMRA. For function F_4, CMA-ES was able to obtain the global optimal solution. Whereas, for functions F_7 and F_8, IDE and B-BBO are found as best algorithms among other variants, respectively. For hybrid (F_{17}-F_{22}) functions, MPA is identified as a best algorithm for functions F_{17}, F_{20} and F_{21} while for rest of the functions F_{18}, F_{19} and F_{22}, SaDN is found to be the best. In case of composition functions (F_{23}-F_{30}), SaDN again provided high-grade results in terms of mean and std., except the function F_{25} and F_{27}. For functions F_{25} and F_{27}, MPA performed very well among other algorithms. Moreover, based on the wilcoxon Rank-sum test (p-rank), only MPA win (w) the proposed algorithm for 6 functions, CMA-ES for 5 functions, SinDE for 4, IDE for 3, ISOS for 1, and B-BBO for 1 function. All other algorithms are failed to beat the SaDN. After that, the statistical Friedman rank test has been performed for all the benchmark functions to validate the capability and efficiency of the proposed SaDN algorithm, as listed in Table 11. Finally, it is analysed that SaDN gave the highest ranking for 19 functions, MPA for 5, CMA-ES for 2, IDE for 2, SinDE for 1 and B-BBO for 1 function. Overall, SaDN attained the first rank with a minimum f-value of 49. Hence, SaDN is again recognized as the best algorithm as compared to all other variants.

5.9. Comparison on CEC 2019 benchmark problems

In this section, the proposed SaDN algorithm performance has been evaluated on ten CEC 2019 benchmark functions (100-Digit Challenge) [64] and compared the results with respect to some standard Grey Wolf Optimization (GWO) [55], Mutated Salp Swarm Algorithm (MSSA) [65],

Table 9: CEC 2014 Real parameter benchmark optimization functions

	No.	Functions	$F_i^* = F_i(x^*)$
Unimodal Functions	F_1	Rotated High Conditioned Elliptic Function	100
	F_2	Rotated Cigar Function	200
	F_3	Rotated Discus Function	300
Simple Multimodal Functions	F_4	Shifted and Rotated Rosenbrock's Function	400
	F_5	Shifted and Rotated Ackley's Function	500
	F_6	Shifted and Rotated Weierstrass Function	600
	F_7	Shifted and Rotated Griewank's Function	700
	F_8	Shifted Rastrigin's Function	800
	F_9	Shifted and Rotated Rastrigin's Function	900
	F_{10}	Shifted Schwefel's Function	1000
	F_{11}	Shifted and Rotated Schwefel's Function	1100
	F_{12}	Shifted and Rotated katsuura Function	1200
	F_{13}	Shifted and Rotated HappyCat Function	1300
	F_{14}	Shifted and Rotated HGBat Function	1400
	F_{15}	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	1500
	F_{16}	Shifted and Rotated Expanded Scaffer's F6 Function	1600
Hybrid Functions	F_{17}	Hybrid Function 1($N = 3$)	1700
	F_{18}	Hybrid Function 2($N = 3$)	1800
	F_{19}	Hybrid Function 3($N = 4$)	1900
	F_{20}	Hybrid Function 4($N = 4$)	2000
	F_{21}	Hybrid Function 5($N = 5$)	2100
	F_{22}	Hybrid Function 6($N = 5$)	2200
Composition Functions	F_{23}	Composition Function 1($N = 5$)	2300
	F_{24}	Composition Function 2($N = 3$)	2400
	F_{25}	Composition Function 3($N = 3$)	2500
	F_{26}	Composition Function 4($N = 5$)	2600
	F_{27}	Composition Function 5($N = 5$)	2700
	F_{28}	Composition Function 6($N = 5$)	2800
	F_{29}	Composition Function 7($N = 3$)	2900
	F_{30}	Composition Function 8($N = 3$)	3000
		Search Range: $[-100, 100]^D$	

Table 10: Statistical results of proposed algorithm in comparison to other algorithms for CEC 2014

		LX-BBO	B-BBO	RW-GWO	ISOS	MPA	IMEHO	VNBA	CMA-ES	IDE	SinDE	BDE	MG-SCA	NMRA	SaDN
		[54]	[54]	[55]	[56]	[57]	[58]	[58]	[63]	[63]	[63]	[63]	[63]		
F_1	mean	1.01E+07	6.50E+06	8.02E+06	9.82E+05	3.98E+06	2.38E+06	2.43E+08	**2.70E-14**	3.81E+07	1.51E+06	1.25E+07	2.92E+07	4.48E+04	2.17E+04
	std	1.01E+07	1.30E+06	3.31E+06	7.05E+05	1.00E+06	4.32E+06	5.93E+07	**1.13E-14**	7.92E+06	1.19E+06	7.47E+07	2.07E+07	2.24E+05	2.14E+04
	p-rank	−	−	−	−	−	−	−	+	−	−	−	−	−	
	f-rank	10	8	9	4	7	6	14	1	13	5	11	12	3	2
F_2	mean	5.34E+04	2.35E+04	2.23E+05	5.27E+00	1.50E+04	5.69E+03	1.92E+10	5.29E-14	**0.00E+00**	**0.00E+00**	9.25E+03	2.26E+09	8.56E+03	1.06E+02
	std	2.14E+04	9.99E+03	5.51E+05	1.72E+01	1.32E+04	4.87E+03	4.23E+09	2.11E-14	**0.00E+00**	**0.00E+00**	6.19E+04	1.69E+09	4.76E+03	7.60E+02
	p-rank	−	−	−	+	−	−	−	+	+	+	−	−	−	
	f-rank	10	9	11	3	8	6	13	2	1	1	7	12	5	4
F_3	mean	1.63E+04	6.03E+03	3.16E+02	4.79E+02	1.00E-04	4.41E+02	2.96E+04	1.07E-13	**0.00E+00**	3.13E-11	5.27E+02	1.77E+04	2.04E+04	6.90E+01
	std	1.70E+04	3.15E+03	4.34E+02	6.24E+02	9.94E-05	1.58E+02	1.39E+04	4.31E-14	**1.80E-14**	1.54E-10	1.11E+03	6.63E+03	5.04E+03	4.92E+02
	p-rank	−	−	−	−	+	−	−	+	+	+	−	−	−	
	f-rank	10	9	6	8	4	7	13	2	1	3	8	11	12	5
F_4	mean	9.99E+01	1.02E+02	3.41E+01	5.98E+01	9.05E+00	5.24E+02	2.20E+03	**1.07E-13**	4.81E+01	4.24E+00	5.76E+02	2.76E+02	1.83E+01	2.57E-01
	std	2.84E+01	3.13E+01	1.80E+01	3.57E+01	2.15E+01	4.77E+01	3.63E+02	**5.11E-14**	4.31E+01	1.27E+01	4.29E+01	6.55E+01	6.10E+00	1.83E+00
	p-rank	−	−	−	−	−	−	−	+	−	−	−	−	−	
	f-rank	9	10	6	8	4	12	14	1	7	3	13	11	5	2
F_5	mean	3.06E+00	3.74E+00	2.05E+01	2.03E+01	2.00E+01	5.21E+02	5.21E+02	2.00E+01	2.07E+01	2.05E+01	2.02E+01	2.04E+01	2.08E+01	**3.92E-01**
	std	7.86E-01	4.91E-01	7.46E-02	6.67E-02	2.86E-02	5.99E-02	5.43E-02	1.57E-02	6.70E-02	4.61E-02	1.33E-01	1.44E-01	6.15E-02	**2.80E+00**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	2	3	10	7	5	14	13	4	11	9	6	8	12	1
F_6	mean	1.70E+01	1.99E+01	9.84E+00	1.05E+01	6.88E+00	6.12E+02	6.33E+02	4.20E+01	2.38E+01	3.48E+00	2.38E+01	1.94E+01	2.30E+01	**3.65E-01**
	std	3.12E+00	2.70E+00	3.49E+00	2.39E+00	1.93E+00	2.72E+00	2.58E+00	1.06E+01	1.66E+00	2.01E+00	4.39E+00	2.89E+00	2.04E+00	**1.60E+00**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	6	8	4	5	3	13	14	12	10	2	11	7	9	1
F_7	mean	1.75E-01	7.81E-02	2.53E-01	1.56E-02	3.49E-02	7.00E+02	8.11E+02	2.66E-03	**1.07E-09**	1.93E-04	1.50E+00	1.99E+01	7.00E-03	1.51E-07
	std	8.56E-02	4.44E-02	1.43E-01	1.83E-02	2.92E-02	1.19E-01	1.81E+01	4.65E-03	**7.61E-09**	1.38E-03	6.08E+00	1.18E+01	3.05E-03	1.07E-06
	p-rank	−	−	−	−	−	−	−	−	+	−	−	−	−	
	f-rank	9	8	10	6	7	13	14	4	1	3	11	12	5	2
F_8	mean	5.53E+01	**4.71E-01**	4.38E+01	1.47E+01	1.33E+01	8.33E+02	9.74E+02	4.29E+02	1.05E+01	9.99E-01	6.12E+01	1.07E+02	9.60E+01	1.15E+00
	std	3.78E+02	**6.79E-01**	8.48E+00	3.34E+00	5.10E+01	9.19E+00	1.61E+01	9.03E+01	1.06E+01	1.48E+00	2.60E+01	2.14E+01	1.10E+01	8.23E+00
	p-rank	−	+	−	−	−	−	−	−	−	+	−	−	−	
	f-rank	8	1	7	6	5	13	14	12	4	2	9	11	10	3
F_9	mean	7.66E+01	9.11E+01	6.33E+01	2.56E+02	7.06E+01	9.32E+02	1.15E+03	6.37E+02	1.41E+02	3.29E+01	1.10E+02	1.39E+02	2.79E+02	**5.14E+00**
	std	1.61E+01	1.54E+01	1.30E+01	1.34E+01	1.43E+01	1.15E+01	2.03E+01	1.59E+02	1.17E+01	7.50E+00	4.69E+01	2.56E+01	2.84E+01	**1.07E+01**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	5	6	3	10	4	13	14	12	9	2	7	8	11	1
F_{10}	mean	1.25E+04	6.68E+03	9.61E+02	1.78E+03	4.50E+02	3.26E+03	4.50E+03	5.15E+03	1.03E+02	5.93E+01	1.75E+03	2.82E+03	1.35E+03	**1.42E+01**
	std	1.16E+02	4.58E+02	2.72E+02	4.09E+01	2.14E+02	5.72E+02	3.47E+02	8.13E+02	7.48E+01	7.09E+01	7.28E+02	6.83E+02	2.28E+02	**1.01E+01**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	14	13	5	8	4	10	11	12	3	2	7	9	6	1
F_{11}	mean	1.23E+04	6.71E+03	2.68E+03	1.48E+03	1.96E+03	3.96E+03	7.90E+03	5.14E+03	5.49E+03	2.43E+03	3.90E+03	3.30E+03	3.16E+03	**5.34E+01**
	std	3.41E+02	5.17E+02	3.68E+02	4.54E+02	4.57E+02	5.38E+02	3.79E+02	7.61E+02	2.91E+02	5.14E+02	7.38E+02	6.26E+02	3.82E+02	**3.81E+02**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	14	12	5	2	3	9	13	10	11	4	8	7	6	1
F_{12}	mean	1.11E-02	1.11E-02	5.45E-01	3.55E-01	1.00E-01	1.20E+03	1.20E+03	2.82E-01	1.24E+00	4.10E-01	3.77E-01	6.33E-01	8.00E-01	**3.90E-03**
	std	1.75E-18	1.75E-18	1.66E-01	5.73E-02	2.36E-02	5.26E-01	3.51E-01	2.61E-01	1.83E-01	1.58E-01	1.71E-01	3.36E-01	1.46E-01	**2.82E-02**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	2	2	8	5	3	13	12	4	11	7	6	9	10	1
F_{13}	mean	6.55E-01	6.78E-01	2.80E-01	3.77E-01	3.00E-01	1.30E+03	1.30E+03	2.35E-01	3.54E-01	1.36E-01	5.13E-01	5.51E-01	3.00E-01	**4.80E-03**
	std	1.56E-01	7.98E-02	6.30E-02	7.10E-02	5.31E-02	6.25E-02	3.64E-01	5.87E-02	3.60E-02	3.42E-02	1.41E-01	8.94E-02	3.24E-02	**3.44E-02**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	11	12	4	8	6	14	13	3	7	2	9	10	5	1
F_{14}	mean	6.20E-01	3.93E-01	4.23E-01	2.71E-01	2.00E-01	1.40E+03	1.46E+03	3.60E-01	3.05E-01	2.45E-01	4.41E-01	2.34E+00	2.00E-01	**4.10E-03**
	std	2.96E-01	1.55E-01	2.15E-01	5.12E-02	3.70E-02	9.85E-02	1.22E+01	7.16E-02	7.85E-02	1.03E-01	2.46E-01	3.31E+00	2.40E-02	**2.93E-02**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	11	8	9	5	3	13	14	7	6	4	10	12	2	1
F_{15}	mean	1.55E+01	1.88E+01	8.81E+00	1.06E+01	4.70E+00	1.50E+03	3.89E+03	3.52E+00	1.44E+01	3.89E+00	3.78E+01	8.72E+01	2.98E+01	**3.52E-01**
	std	5.49E+00	5.64E+00	1.51E+00	3.71E+00	1.37E+00	1.35E+00	1.22E+03	1.06E+00	9.90E-01	9.19E-01	9.26E+01	1.01E+02	3.90E+00	**2.51E+00**
	p-rank	−	−	−	−	−	−	−	−	−	−	−	−	−	
	f-rank	8	9	5	6	4	13	14	2	7	3	11	12	10	1

Table 11: Statistical results of proposed algorithm in comparison to other algorithms for CEC 2014

		LX-BBO	B-BBO	RW-GWO	ISOS	MPA	IMEHO	VNBA	CMA-ES	IDE	SinDE	BDE	MG-SCA	NMRA	SaDN
		[54]	[54]	[55]	[56]	[57]	[58]	[58]	[63]	[63]	[63]	[63]	[63]		
F_{16}	mean	1.08E+01	1.06E+01	1.03E+01	9.21E+01	1.04E+01	1.61E+03	1.61E+03	1.43E+01	1.20E+01	1.01E+01	1.26E+01	1.16E+01	1.28E+01	**2.30E-01**
	std	5.84E-01	6.25E-01	6.11E-01	7.31E-01	6.12E-01	7.64E-01	3.66E-01	3.67E-01	2.27E-01	5.32E-01	5.92E-01	6.91E-01	2.58E-01	**1.64E+00**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	6	5	3	12	4	14	13	11	8	2	9	7	10	1
F_{17}	mean	1.49E+06	1.27E+06	5.71E+05	1.75E+05	**8.90E+00**	7.86E+04	7.53E+06	1.65E+03	1.54E+06	1.32E+05	2.80E+04	9.56E+05	1.81E+05	2.91E+03
	std	9.34E+05	5.46E+05	4.10E+05	1.64E+05	**1.30E+02**	8.38E+04	3.34E+06	3.68E+02	6.94E+05	1.24E+05	2.24E+04	7.62E+05	8.26E+04	2.08E+04
	p-rank	–	–	–	–	+	–	–	+	–	–	–	–	–	
	f-rank	12	11	9	7	1	5	14	2	13	6	4	10	8	3
F_{18}	mean	2.89E+03	8.22E+02	6.52E+03	3.89E+03	1.40E+01	5.10E+03	1.66E+08	1.35E+02	2.65E+03	9.53E+02	3.54E+04	1.48E+05	8.90E+02	**4.64E+00**
	std	4.27E+03	1.00E+03	4.63E+02	5.15E+03	5.44E+00	3.52E+03	1.03E+08	4.10E+01	2.05E+03	1.64E+03	1.63E+05	9.00E+05	3.02E+02	**3.31E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	8	4	11	9	2	10	14	3	7	6	12	13	5	1
F_{19}	mean	5.19E+03	7.81E+03	1.14E+01	7.79E+00	4.50E+00	1.91E+03	2.02E+03	9.88E+00	8.33E+00	4.76E+00	1.11E+01	2.28E+01	1.04E+01	**1.78E-01**
	std	5.67E+03	4.67E+03	2.03E+00	1.78E+00	7.52E-01	1.74E+00	3.82E+01	1.95E+00	7.27E-01	8.44E-01	1.08E+01	1.43E+01	1.52E+00	**1.27E+00**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	13	14	9	4	2	11	12	6	5	3	8	10	7	1
F_{20}	mean	2.61E+04	1.62E+04	6.27E+02	4.98E+03	**1.11E+01**	2.21E+03	1.89E+04	2.63E+02	2.85E+02	2.17E+01	3.02E+03	4.24E+03	8.93E+03	7.27E+01
	std	1.57E+04	4.11E+03	1.12E+03	3.40E+03	**3.81E+00**	8.17E+01	6.57E+03	1.11E+02	1.12E+02	3.05E+01	7.27E+03	3.82E+03	4.28E+03	5.19E+02
	p-rank	–	–	–	–	+	–	–	–	–	+	–	–	–	
	f-rank	14	12	6	10	1	7	13	4	5	2	8	9	11	3
F_{21}	mean	1.11E+06	1.22E+06	2.58E+05	8.90E+04	**7.78E+01**	2.93E+04	2.31E+06	1.10E+03	2.11E+05	1.81E+04	5.08E+04	2.35E+05	5.10E+04	2.98E+02
	std	7.95E+05	7.96E+05	1.76E+05	1.07E+05	**6.46E+01**	1.83E+04	1.35E+06	3.28E+02	1.01E+05	2.57E+04	1.24E+05	2.39E+05	2.25E+04	2.13E+03
	p-rank	–	–	–	–	+	–	–	–	–	–	–	–	–	
	f-rank	12	13	11	8	1	5	14	3	9	4	6	10	7	2
F_{22}	mean	1.88E+03	1.68E+02	2.08E+02	2.75E+02	1.29E+02	2.41E+03	3.04E+03	3.10E+02	1.12E+02	5.91E+01	6.81E+02	3.39E+02	3.93E+02	**1.29E+01**
	std	2.03E+02	2.47E+02	2.08E+02	1.45E+02	6.10E+01	1.01E+02	1.28E+02	1.79E+02	6.33E+01	4.84E+01	2.11E+02	1.78E+02	1.20E+02	**9.21E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	6	5	7	8	4	13	14	9	3	2	12	10	11	1
F_{23}	mean	4.11E+02	3.43E+02	3.15E+02	3.15E+02	3.15E+02	2.62E+03	2.69E+03	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.29E+02	2.02E+02	**3.92E+00**
	std	6.43E+01	2.84E+01	2.77E-01	0.00E+00	6.60E-08	4.78E-01	2.47E+01	1.71E-13	1.36E-12	1.35E-12	1.20E-01	4.03E+00	1.60E+01	**2.80E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	12	11	9	3	7	13	14	4	6	5	8	10	2	1
F_{24}	mean	1.47E+04	3.41E+04	2.00E+02	2.00E+02	2.00E+02	2.64E+03	2.63E+03	2.98E+02	2.25E+02	2.26E+02	2.46E+02	2.00E+02	2.00E+02	**3.92E+00**
	std	8.37E+03	2.35E+04	3.04E-03	1.50E-03	3.16E-04	6.46E+00	2.53E+01	3.80E+02	2.75E+00	5.40E+00	5.69E+00	1.56E-03	0.00E+00	**2.80E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	13	14	6	4	3	11	10	12	7	8	9	5	2	1
F_{25}	mean	5.29E+02	6.53E+02	2.04E+02	2.00E+02	**1.00E-01**	2.71E+03	2.71E+03	2.04E+02	2.12E+02	2.04E+02	2.08E+02	2.11E+02	2.00E+02	3.92E+00
	std	4.36E+01	6.01E+01	1.18E+00	8.07E-01	**3.61E-01**	2.08E+00	1.07E+01	2.99E+00	1.71E+00	9.08E-01	4.51E+00	2.82E+00	0.00E+00	2.80E+01
	p-rank	–	–	–	–	+	–	–	–	–	–	–	–	–	
	f-rank	11	12	6	4	1	14	13	7	10	5	8	9	3	2
F_{26}	mean	2.12E+00	3.64E+01	1.00E+02	1.00E+02	1.00E+01	2.70E+03	2.70E+03	1.02E+02	1.00E+02	1.00E+02	1.01E+02	1.01E+02	1.00E+02	**1.96E+00**
	std	3.46E+00	5.62E+01	7.36E-02	9.55E-02	2.99E+01	6.00E-02	4.30E-01	1.38E+01	4.35E-02	4.28E-02	1.17E+00	1.53E-01	7.16E-02	**1.40E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	2	4	8	9	3	14	13	12	6	5	10	11	7	1
F_{27}	mean	1.95E+02	3.04E+02	4.09E+02	5.43E+02	**8.00E-01**	3.28E+03	3.99E+03	4.17E+02	5.06E+02	3.51E+02	8.82E+02	8.19E+02	3.33E+02	3.92E+00
	std	1.04E+02	1.60E+02	6.09E+00	1.36E+02	**4.25E-01**	1.41E+02	3.30E+01	1.75E+02	1.08E+02	4.30E+01	2.03E+02	9.17E+01	9.95E+01	2.80E+01
	p-rank	–	–	–	–	+	–	–	–	–	–	–	–	–	
	f-rank	3	4	7	10	1	13	14	8	9	6	12	11	5	2
F_{28}	mean	1.94E+03	2.12E+03	4.34E+02	9.68E+02	8.38E+02	3.77E+03	4.48E+03	3.98E+03	8.39E+02	8.35E+02	1.29E+03	9.68E+02	1.12E+03	**3.92E+00**
	std	1.04E+02	4.44E+02	8.45E+00	4.12E+01	5.42E+01	2.44E+02	2.33E+02	3.09E+03	2.71E+01	2.48E+01	2.88E+02	1.06E+02	1.03E+02	**2.80E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	10	11	2	6	4	12	14	13	5	3	9	7	8	1
F_{29}	mean	1.98E+07	3.09E+07	2.14E+02	5.70E+05	4.40E+02	4.11E+03	7.48E+06	8.01E+02	1.98E+05	9.27E+05	4.80E+06	1.19E+06	2.08E+02	**3.92E+00**
	std	3.95E+06	6.91E+06	2.37E+00	2.14E+06	2.60E+02	2.16E+02	1.20E+06	9.49E+01	1.18E+06	2.61E+06	4.83E+06	3.25E+06	2.86E+00	**2.80E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	13	14	3	8	4	6	12	5	7	9	11	10	2	1
F_{30}	mean	6.95E+06	1.38E+07	6.69E+02	2.38E+05	7.28E+02	7.08E+03	1.92E+05	2.55E+03	2.89E+03	3.11E+03	1.76E+04	1.92E+04	4.39E+02	**3.92E+00**
	std	1.03E+07	1.08E+07	2.14E+02	1.10E+03	3.48E+02	1.44E+03	1.03E+05	6.60E+02	1.08E+03	1.24E+03	3.18E+04	8.25E+03	1.77E+02	**2.80E+01**
	p-rank	–	–	–	–	–	–	–	–	–	–	–	–	–	
	f-rank	13	14	3	12	4	8	11	5	6	7	9	10	2	1
w/l/t		30/0/0	29/1/0	30/0/0	29/1/0	24/6/0	30/0/0	30/0/0	25/5/0	27/3/0	26/4/0	30/0/0	30/0/0	30/0/0	NA
Overall F-value		277	266	202	205	112	325	395	192	208	125	269	293	201	49
Overall F-rank		11	9	6	7	2	13	14	4	8	3	10	12	5	1

Levy based Naked-Mole Rat Algorithm (LNMRA) [66], Cuckoo Search (CS), Fitness Dependent Optimizer (FDO) [67], DE, Adaptive Flower Pollination Algorithm (AFPA) [64], Naked Mole-Rat Algorithm version 3.0 (NMRV 3.0) [68], self-adaptive Differential Evolution (jDE100) [69], and improved elephant herding optimization (EHOI) [70]. All the functions are scalable functions. Few of them have shifted and rotated properties (CEC4-CEC10), whereas CEC1-3 have default properties are noted in Table 12. During the simulation, the population size kept as 50 with 500 maximum number of iterations. The simulation performance is noted for each of the 51 NMR runs. Apart from this, the simulated results for all the algorithms are compared in terms of best, worst, median, mean and standard deviation as shown in Table 13. From these analyses, it is observed that the proposed SaDN algorithm has been performed well and found to be best among other algorithms. SaDN attained zero value for the best and median. Hence, SaDN is again recognized as the best algorithm as compared to all other variants.

Table 12: CEC 2019 benchmark functions (100-digit challenge)

Function No.	Functions	Dim	Search range	$F_i^* = F_i(x^*)$
CEC01	Storn's Chebyshev Polynomial Fitting Problem	9	[-8192, 8192]	1
CEC02	Inverse Hilbert Matrix Problem	16	[-16,384, 16384]	1
CEC03	Lennard-Jones Minimum Energy Cluster	18	[-4, 4]	1
CEC04	Rastrigin's Function	10	[-100, 100]	1
CEC05	Griewangk's Function	10	[-100, 100]	1
CEC06	Weierstrass Function	10	[-100, 100]	1
CEC07	Modified Schwefel's Function	10	[-100, 100]	1
CEC08	Expanded Schaffer's F6 Function	10	[-100, 100]	1
CEC09	Happy Cat Function	10	[-100, 100]	1
CEC10	Ackley Function	10	[-100, 100]	1

5.10. Summary of results

Though NMRA is a good algorithm but it suffers from the problem of poor exploration and may get stuck in some local optimal solution. So it becomes very necessary to enhance the NMRA algorithm to improve its working capabilities. The salient features and the major enhancements proposed in present work are highlighted as:

- The later versions of DE algorithm are found to have better exploration operation and are governed by very simple random parameters. The exploitation part is a bit challenging and a lot of studies have been proposed to overcome the same.

- In present work, NMRA is hybridized with DE algorithm to improve the exploration properties of NMRA algorithm and exploitation properties of DE. The new modification has been added

Table 13: Statistical results of proposed algorithm in comparison to other algorithms for CEC 2019

		GWO	MSSA	LNMRA	CS	FDO	DE	NMRA	AFPA [64]	NMRV 3.0 [68]	jDE100 [69]	EHOI [70]	SaDN
CEC01	best	4.29E+04	7.10E+04	1.22E+05	1.00E+10	4.34E+05	9.01E+05	5.87E+04	1.05E+06	3.68E+04	3.37E+04	-	**0.00E+00**
	worst	1.11E+09	1.17E+06	1.29E+06	1.00E+10	2.85E+09	5.75E+08	1.53E+06	1.75E+08	4.82E+04	6.23E+05	-	**2.76E+05**
	mean	1.32E+08	2.05E+05	5.79E+05	1.00E+10	7.37E+08	6.70E+07	8.60E+05	2.52E+07	4.01E+04	1.59E+05	4.69E+04	**1.08E+04**
	std	2.64E+08	2.04E+05	2.94E+05	0.00E+00	7.36E+08	9.11E+07	4.69E+05	3.59E+07	2.45E+03	1.59E+05	2.87E+03	5.41E+04
CEC02	best	1.73E+01	1.73E+01	1.75E+01	1.73E+01	2.01E+01	1.73E+01	1.83E+01	1.73E+01	1.73E+01	2.33E+06	-	**0.00E+00**
	worst	1.73E+01	1.85E+01	1.98E+01	1.73E+01	1.73E+01	1.74E+01	1.98E+01	1.73E+01	1.74E+01	2.43E+06	-	**1.85E+01**
	mean	1.73E+01	1.76E+01	1.84E+01	1.73E+01	1.73E+01	1.73E+01	1.95E+01	1.73E+01	1.73E+01	2.38E+06	1.73E+01	**7.28E-01**
	std	2.14E-04	2.36E-01	6.09E-01	3.94E-05	3.94E-06	1.22E-02	4.51E-01	2.80E-05	6.40E-03	2.71E+04	1.18E-15	3.64E+00
CEC03	best	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.37E+05	-	**0.00E+00**
	worst	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	2.81E+06	-	**1.27E+01**
	mean	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.27E+01	1.31E+06	1.27E+01	**4.98E-01**
	std	1.69E-06	1.23E-05	4.12E-08	8.37E-11	1.19E-11	3.63E-15	8.68E-04	2.78E-14	9.00E-15	8.51E+05	1.87E-15	2.49E+00
CEC04	best	2.23E+01	2.03E+02	1.00E+02	1.87E+01	1.53E+01	6.96E+00	1.94E+03	2.06E+01	4.23E+01	1.25E+05	-	**0.00E+00**
	worst	4.18E+03	1.97E+03	8.65E+02	3.53E+01	1.10E+02	1.10E+03	2.16E+04	4.09E+01	1.96E+03	5.09E+05	-	**3.22E+04**
	mean	1.49E+02	6.02E+02	3.70E+02	2.65E+01	4.18E+01	1.51E+02	9.25E+03	2.92E+01	7.05E+02	3.47E+05	1.27E+01	**1.26E+02**
	std	5.88E+02	3.53E+02	1.84E+02	3.72E+00	2.22E+01	2.34E+02	4.40E+03	5.41E+00	5.32E+02	1.14E+05	3.95E+00	6.31E+03
CEC05	best	1.07E+00	1.57E+00	1.29E+00	1.02E+00	1.00E+00	1.00E+00	2.33E+00	1.03E+00	1.17E+00	5.52E+04	-	**0.00E+00**
	worst	1.85E+00	2.10E+00	2.55E+00	1.11E+00	1.40E+00	1.29E+00	5.88E+00	1.08E+00	2.69E+00	3.77E+05	-	**3.60E+00**
	mean	1.34E+00	1.83E+00	1.84E+00	1.05E+00	1.17E+00	1.13E+00	3.73E+00	1.06E+00	1.59E+00	1.67E+05	1.04E+00	**1.41E-01**
	std	2.39E-01	1.17E-01	2.87E-01	1.74E-02	1.00E-01	7.20E-02	8.73E-01	1.25E-02	3.43E-01	8.42E+04	2.12E-02	7.06E-01
CEC06	best	8.50E+00	8.34E+00	8.50E+00	7.01E+00	8.29E+00	4.69E+00	8.40E+00	8.25E+00	9.65E+00	3.50E+04	-	**0.00E+00**
	worst	1.17E+01	1.21E+01	1.19E+01	1.04E+01	1.22E+01	1.037E+01	1.26E+01	1.16E+01	1.25E+01	4.15E+04	-	**1.38E+01**
	mean	1.06E+01	1.06E+01	1.03E+01	9.17E+00	1.08E+01	7.65E+00	1.11E+01	1.03E+01	1.13E+01	3.84E+04	8.29E+00	**5.43E-01**
	std	7.16E-01	8.21E-01	7.26E-01	6.72E-01	8.66E-01	1.52E+00	7.51E-01	7.95E-01	7.07E-01	2.06E+03	8.19E-01	2.71E+00
CEC07	best	1.96E+00	1.32E+02	-1.88E+02	-1.02E+02	-3.91E+01	-1.05E+02	4.73E+01	-4.56E+01	2.67E+02	2.48E+06	-	**0.00E+00**
	worst	1.01E+03	1.28E+03	1.93E+02	2.01E+02	7.81E+02	9.80E+02	6.93E+02	1.81E+02	1.34E+03	1.58E+07	-	**1.44E+03**
	mean	3.63E+02	6.46E+02	4.31E+01	6.07E+01	2.93E+02	3.17E+02	3.31E+02	7.83E+01	7.74E+02	9.10E+06	1.40E+02	**5.65E+01**
	std	2.65E+02	2.48E+02	8.16E+01	7.29E+01	1.79E+02	2.46E+02	1.38E+02	6.45E+01	2.34E+02	4.52E+06	1.04E+02	2.82E+02
CEC08	best	2.53E+00	4.63E+00	4.74E+00	4.47E+00	2.72E+00	2.54E+00	5.44E+00	2.94E+00	2.86E+00	1.59E+08	-	**0.00E+00**
	worst	6.37E+00	6.89E+00	6.18E+00	5.71E+00	6.45E+00	7.05E+00	6.94E+00	5.63E+00	6.79E+00	1.77E+09	-	**7.49E+00**
	mean	4.60E+00	6.14E+00	5.59E+00	5.26E+00	5.12E+00	5.26E+00	6.17E+00	4.59E+00	5.64E+00	1.21E+09	2.72E+00	**2.93E-01**
	std	1.01E+00	4.96E-01	2.71E-01	2.33E-01	8.15E-01	1.03E+00	2.99E-01	8.36E-01	8.41E-01	4.38E+08	8.77E-01	1.46E+00
CEC09	best	2.74E+00	3.19E+00	3.01E+00	2.47E+00	2.37E+00	2.35E+00	1.53E+02	2.42E+00	3.27E+00	6.08E+08	-	**0.00E+00**
	worst	5.94E+00	7.30E+01	1.03E+02	2.96E+00	4.68E+00	4.83E+00	3.46E+03	4.72E+00	1.88E+02	1.05E+09	-	**2.85E+03**
	mean	4.27E+00	1.20E+01	1.02E+01	2.73E+00	2.60E+00	2.57E+00	1.34E+03	3.02E+00	4.48E+01	9.20E+08	2.35E+00	**1.11E+02**
	std	8.96E-01	1.35E+01	1.73E+01	1.16E-01	3.62E-01	3.75E-01	7.79E+02	5.23E-01	5.65E+01	1.31E+08	6.23E-03	5.59E+02
CEC10	best	3.60E+00	2.01E+01	2.01E+01	2.01E+01	4.35E-14	2.00E+01	2.00E+01	1.01E+01	1.44E+01	1.79E+05	-	**0.00E+00**
	worst	2.05E+01	2.06E+01	2.05E+01	2.03E+01	2.00E+01	2.05E+01	2.05E+01	2.06E+01	2.07E+01	2.53E+06	-	**2.05E+01**
	mean	1.98E+01	2.04E+01	2.04E+01	2.02E+01	1.92E+01	2.02E+01	2.02E+01	1.98E+01	2.02E+01	1.54E+06	1.98E+01	**8.07E-01**
	std	2.99E+00	1.16E-01	8.37E-02	5.92E-02	3.92E+00	1.25E-01	1.25E-01	2.35E+00	1.24E+00	7.46E+05	1.50E+00	4.03E+00

in the worker phase of the basic algorithm and general equations of breeder phase are kept same as that of original NMRA.

- The scaling factor parameter of DE has been enhanced by using five different mutation weights namely L, C, NB, T and D.
- Seven different mating factor are used including *random*, *LD*, *Exp*, *Osci*, *SA*, *SD* and *log*.
- Initial experiments show that F is best for Lévy mutation and λ is best for SD inertia weight. Based on all of the added modifications, the SaDN algorithm is proposed.
- The performance evaluation of the proposed SaDN algorithm is done with respect to variable population and dimension size on CEC 2005 and CEC 2014 benchmark functions. Here the best population size is identified and based on that a comparison with respect to other recent state-of-the-art algorithms is performed.
- Though the algorithm provides potentially viable solutions but experimental results demonstrate that the algorithm is less efficient for fixed dimension problems and more work is required to be done to improve its properties for these set of optimization problems.

Overall, it can be summarized that the proposed hybrid variant of DE and NMRA is a self-adaptive algorithm and no user based initialization of parameter is required to improve its working properties. The newly proposed SaDN algorithm has performed better than most of the recently proposed algorithms and hence is expected to provide viable results for solving various domain research problems. In the next section, the proposed SaDN algorithm is used for optimum design of frame structures.

6. Real Time Mechanical Engineering Design Problems

The proposed SaDN algorithm has been tested on the mechanical engineering design problems such as the optimization of a pressure vessel, a welded beam design problem and the design of a tension compression spring [71]. For completeness, we report the results in terms of best and worst fitness value achieved, together with the median and standard deviation value for each problem attained for 30 runs. It is noted that all the engineering design problems are conducted using the 500 number of iterations and population size of 30 for SaDN.

6.1. Pressure Vessel Design Problem

This engineering problem is the problem considered for multidisciplinary design optimization for a mixed (continuous/discrete) type of variables. A cylindrical pressure vessel with hemispheric heads at both ends will operate at a pressure of 3,000 psi ($2.1 \times 10^7 Pa$) and a minimum volume of 750 ft^3 ($21.24m^3$) in compliance with the requirement of the ASME boiler code. The fitness function of this problem is the cost minimization of a cylindrical vessel's material, formation and welding shown in Figure 3. Four specific variables are used in this engineering application and these are: Cylindrical shell Thickness (T_s), Cylindrical head Thickness (T_h), Cylindrical Inside radius (R) and Length of the cylindrical segment (L). As per the availability of the thickness of rolled steel plates, both thickness variables (T_s, T_h) must be 0.0625-inch integer multiple values.

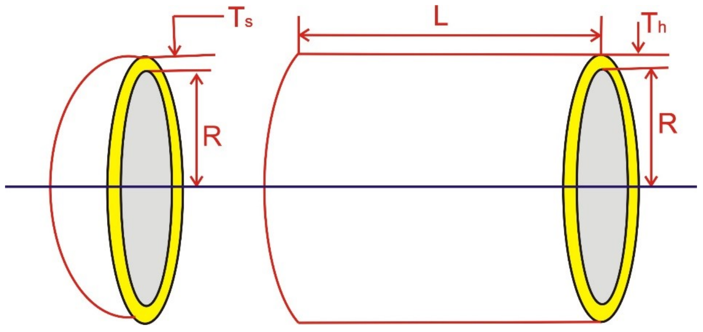

Figure 3: Pressure Vessel Design Problem

Four constraints are set on the Pressure Vessel Design problem. The problem formulation along with its constraints is given in equations (29) to (33) below.

Consider, $\vec{a} = [a_1\ a_2\ a_3\ a_4]$=[$T_s$ T_h R L]

$$Minimize, \quad f(\vec{a}) = 0.6224a_1a_3a_4 + 1.7781a_2a_3^2 + 3.1661a_1^2a_4 + 19.84a_1^2a_3 \tag{29}$$

$$subjected\,to, \quad g_1(\vec{a}) = -a_1 + 0.0193a_3 \leq 0 \tag{30}$$

$$g_2(\vec{a}) = a_2 + 0.00954a_3 \leq 0 \tag{31}$$

$$g_3(\vec{a}) = \pi a_3^2a_4 - \frac{4}{3}\pi a_3^2 + 1296000 \leq 0 \tag{32}$$

$$g_4(\vec{a}) = a_4 - 240 \leq 0 \tag{33}$$

where the range of a_1, a_2, a_3 and a_4 Varies between 0.0625$\leq$ a_1 $\leq$ 99×0.0625, $0.0625 \leq a_2 \leq 99 \times 0.0625$, $10 \leq a_3 \leq 200$, $10 \leq a_4 \leq 200$

Table 14 summarizes the optimal findings in the literature obtained by other approaches along with the results obtained for SaDN algorithm. The outcome of Pressure Vessel Design problem is compared with meta-heuristic algorithms [6],[72]-[57]. The results for SaDN are presented in mixed variable form and compared it with mixed variable solutions only. SaDN has outperformed other processes, as shown in Tables 14 and 15. The statistical findings are summarized in Table 15.

Table 14: Optimum results comparison for the pressure vessel design problem

Algorithm	T_s	T_h	R	L	$f(\vec{a})$
HGA(2) [72]	1.1250	0.5625	58.1267	44.59410000	6832.5830
T-Cell [73]	0.8125	0.4375	42.0984	190.78769500	6390.5540
HGA(1) [72]	0.8125	0.4375	42.0492	177.25220000	6065.8210
CPSO [74]	0.8125	0.4375	42.0913	176.74650000	6061.0777
BFOA [75]	0.8125	0.4375	42.0964	176.68323100	6060.4610
HAIS-GA [76]	0.8125	0.4375	42.0931	176.70310000	6060.3670
DTS-GA [77]	0.8125	0.4375	42.0974	176.65404700	6059.9463
ES [78]	0.8125	0.4375	42.0981	176.64051800	6059.7450
CDE [79]	0.8125	0.4375	42.0984	176.63769000	6059.7340
SaDN	0.8125	0.4375	42.0984	176.63716500	6059.7199

Table 15: Statistical result comparison for the pressure vessel design problem

Algorithm	Best	Mean	Worst	Std	No. of Evaluations
HGA(2) [72]	6832.584	7187.314	8012.615	276	80,000
T-Cell [73]	6390.554	6737.065	7694.066	357	80,000
HGA(1) [72]	6065.821	6632.376	8248.003	515	80,000
CPSO [74]	6061.0777	6147.1332	6363.8041	86.4545	2,00,000
BFOA [75]	6060.46	6074.625	N.A	156	48,000
HAIS-GA [76]	6061.1229	6743.0848	7368.0602	457.99	1,50,000
DTS-GA [77]	6059.9463	6177.2532	6469.322	130.92	80,000
ES [78]	6059.746	6850	7332.87	426	25,000
CDE [79]	6059.734	6085.2303	6371.0455	43.013	2,40,000
EO [6]	6059.7143	6668.114	7544.4925	566.24	15,000
SaDN	6059.7004	6128.73183	6416.3528	46.4824	15,000

6.2. The Tension/Compression Spring Design Problem

Another well-known benchmark case is the design of a tension/compression spring with an objective of spring weight minimization subject to minimal deflection, shear stress, surge frequency and some box constraints given in equations (34) to (38). This is a multidisciplinary process optimization designed to improve the architecture as shown in Figure 4. Three specification variables are used to solve the problem: Number of active coils (N), Wire diameter (d), and Mean coil diameter (D).

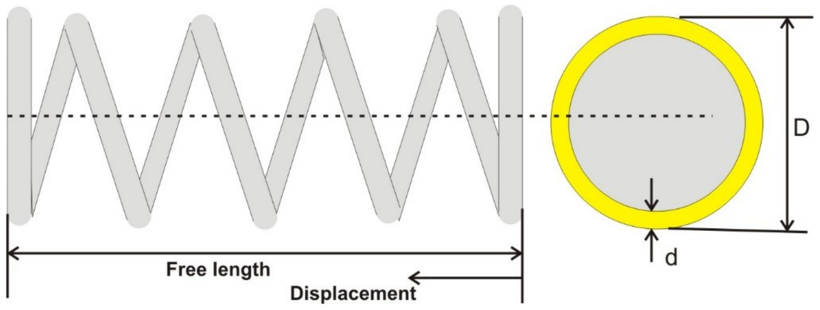

Figure 4: The Spring Design Problem

The statistical description of the problem stated above is expressed as:

Consider $\vec{a}$= $[a_1\ a_2\ a_3]$=[d D N]

$$Minimize, \quad f(\vec{a}) = (a_3 + 2)a_2a_1^2, \tag{34}$$

$$subjected\,to, \quad g_1(\vec{a}) = 1 - \frac{a_2^3a_3}{71875a_1^4} \leq 0 \tag{35}$$

$$g_2(\vec{a}) = \frac{4a_2^2 - a_1a_2}{12566(a_2a_1^3 - a_1^4)} + \frac{1}{5108a_1^2} \leq 0 \tag{36}$$

$$g_3(\vec{a}) = 1 - \frac{140.45a_1}{a_3a_2^2} \leq 0 \tag{37}$$

$$g_4(\vec{a}) = \frac{a_1 + a_2}{1.5} - 1 \leq 0 \tag{38}$$

where the range of a_1, a_2 and a_3 varies between 0.005$\leq a_1 \leq$ 2.0, 0.25 $\leq a_2 \leq$1.30, 2.0 $\leq a_3 \leq$ 15.0 For comparison, the meta-heuristic algorithms [73],[74],[75],[79]-[63] were used. The optimum and statistical summary results are given in Tables 16 and 17. SaDN achieved much better results than the other algorithms, with comparatively less number of function evaluations.

Table 16: Optimum results comparison for the spring design problem

Algorithm	d	D	N	f($\vec{a}$)
SI [80]	0.050417	0.321532	13.979910	0.013060
GA(1) [81]	0.05148	0.351661	11.632201	0.012704
CA [82]	0.05000	0.317395	14.031795	0.012721
GA(2) [83]	0.051989	0.363965	10.890522	0.012681
CPSO [74]	0.051728	0.357644	11.244543	0.012674
BFOA [75]	0.051825	0.359935	11.107103	0.012671
CDE [79]	0.051609	0.354714	11.410831	0.012670
SCA [84]	0.05216	0.368159	10.648442	0.012669
HGA [85]	0.05130	0.347475	11.852177	0.012668
T-Cell [73]	0.051622	0.355105	11.384534	0.012665
EO [6]	0.05162	0.355054	11.387968	0.012666
SaDN	0.051622	0.355105	11.384415	0.012665

Table 17: Statistical result comparison for the spring design problem

Algorithm	Best	Mean	Worst	Std	No. of Evaluations
SI [80]	0.013060	0.015526	0.018992	N.A	20,000
GA(1) [81]	0.012704	0.012769	0.012822	3.93E-05	N.A.
CA [82]	0.012721	0.013568	0.015116	8.40E-04	50,000
GA(2) [83]	0.012681	0.012742	0.012973	9.50E-05	80,000
CPSO [74]	0.012674	0.012730	0.012924	5.19E-05	2,00,000
BFOA [75]	0.012671	0.012759	N.A	1.36E-04	48,000
CDE [79]	0.012670	0.012703	0.012790	2.07E-05	2,40,000
SCA [84]	0.012669	0.012922	0.016717	5.92E-04	25,167
HGA [85]	0.012668	0.013481	0.016155	N.A.	36,000
T-Cell [73]	0.012665	0.013309	0.012732	9.40E-05	36,000
EO [6]	0.012666	0.013017	0.013997	3.91E-04	15,000
SaDN	0.012665	0.012678	0.012872	6.73E-08	15,000

6.3. Welded Beam Design Problem

Another aspect in engineering optimization is the welded beam design which has also served as a benchmark problem to check the effectiveness of optimization algorithms. This is a cantilever beam that is welded at one end and at the other end is subjected to point load (P). The goal is to reduce a welded beam's manufacturing costs with bending stress, shear stress, deflection, buckling load, and other constraints as shown in Figure 5. The problem consists of four variables, which are Bar length (l), thickness (b), height (t), and Weld thickness (h) of the attached part. The problem has four restrictions including lateral limitation, bar buckling load (P_c), end beam deflection (d), beam bending stress (h) and shear stress (s). The expression of the above problem is detailed in (39) to (52), as follows:

Consider, $\vec{a} = [a_1\ a_2\ a_3\ a_4]$=[h l t b]

$$Minimize, \quad f(\vec{a}) = 1.10471 a_1^2 a_2 + 0.04811 a_3 a_4 (14.0 + a_2) \tag{39}$$

$$subjected\, to, \quad g_1(\vec{a}) = \tau(\vec{a}) - \tau_{max} \leq 0 \tag{40}$$

$$g_2(\vec{a}) = \sigma(\vec{a}) - \sigma_{max} \leq 0 \tag{41}$$

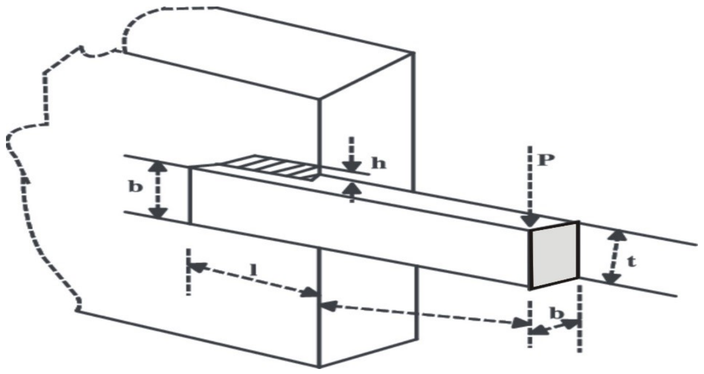

Figure 5: Welded Beam Design Problem

$$g_3(\vec{a}) = \delta(\vec{a}) - \delta_{max} \leq 0 \tag{42}$$

$$g_4(\vec{a}) = a_1 - a_4 \leq 0 \tag{43}$$

$$g_5(\vec{a}) = P - P_c(\vec{a}) \leq 0 \tag{44}$$

$$g_6(\vec{a}) = 0.125 - a_1 \leq 0 \tag{45}$$

$$g_7(\vec{a}) = 1.10471a_1^2 - 0.04811a_3a_4(14.0 + a_2) - 5.0 \leq 0 \tag{46}$$

where the range of a_1, a_2, a_3 and a_4 varies between 0.1$\leq$ a_1 $\leq$ 2, 0.1 $\leq$ a_2 $\leq$ 10, 0.1 $\leq$ a_3 $\leq$ 10, 0.1 $\leq$ a_4 $\leq$ 2

$$\tau(\vec{a}) = \sqrt{(\tau^{/})^2 + 2\tau^{/}\tau^{//}\frac{a_2}{2R} + (\tau^{//})^2} \tag{47}$$

$$where, \quad \tau^{/} = \frac{P}{\sqrt{2}a_1a_2}, \quad \tau^{//} = \frac{MR}{J}, \quad M = P(L + \frac{a_2}{2}), \tag{48}$$

$$R = \sqrt{\frac{a_2^2}{4} + (\frac{a_1 + a_3}{2})^2}, \tag{49}$$

$$J = 2[\sqrt{2}a_1a_2[\frac{a_2^2}{4} + (\frac{a_1 + a_3}{2})^2]] \tag{50}$$

$$\sigma(\vec{a}) = \frac{6PL}{a_4 a_3^2}, \quad \delta(\vec{a}) = \frac{6PL^3}{E a_2^2 a_4}, \tag{51}$$

$$P_c(\vec{a}) = \frac{4.013E\frac{\sqrt{a_3^2 a_4^6}}{36}}{L^2}(1 - \frac{a_3}{2L}\sqrt{\frac{E}{4G}}), \tag{52}$$

Table 18 summarizes the optimal results for different algorithms while the statistical description of results is given in Table 19. For comparison, the meta-heuristic algorithms [6],[73],[74],[75],[86] were used. The tables' findings reveal that SaDN has outperformed other algorithms with lower costs compared to others. SaDN also obtained the results in a small number of function evaluations and with lower values for standard deviation, mean and worst solutions compared to most other algorithms. However, the results of EO are nearly equal to SaDN for same number of evaluations.

Table 18: Optimum results comparison for the welded beam design problem

Algorithm	h	l	t	b	f(a)
SBM [86]	0.2407	6.4851	8.2399	0.2497	2.4426
BFOA [75]	0.2057	3.4711	9.0367	0.2057	2.3868
SCA [84]	0.2444	6.238	8.2886	0.2446	2.3854
EA [87]	0.2443	6.2201	8.2940	0.2444	2.3816
T-Cell [73]	0.2444	6.1286	8.2915	0.2444	2.3811
FSA [88]	0.2444	6.1258	8.2939	0.2444	2.3811
IPSO [89]	0.2444	6.2175	8.2915	0.2444	2.3810
HSA-GA [90]	0.2231	1.5815	12.8468	0.2245	2.2500
SaDN	0.2444	6.21787	8.2915	0.2444	1.9773

Table 19: Statistical result comparison for the welded beam design problem

Algorithm	Best	Mean	Worst	Std	No. of Evaluations
SBM [86]	2.4426	2.5215	2.6315	N.A	19,259
BFOA [75]	2.3868	2.4040	N.A	0.016	48,000
SCA [84]	2.3854	3.2551	6.3996	0.959	33,095
EA [87]	2.3816	N.A	2.3830	0.0003	28,897
T-Cell [73]	2.3811	2.4398	2.7104	0.0931	3,20,000
FSA [88]	2.3811	2.4041	2.4889	N.A	56,243
IPSO [89]	2.3810	2.3819	N.A	0.0052	30,000
HSA-GA [90]	2.25	2.26	2.28	0.0078	26,466
SaDN	1.9773	1.9828	1.9947	5.50E-06	15,000

7. Conclusion

This paper proposes a SaDN algorithm, based on the hybrid concepts of DE and NMRA. The concept of hybridization has been utilized to mitigate the poor exploration properties of NMRA and poor exploitation in case of DE algorithm. Thus it is expected that the hybrid algorithm will have the combined advantage of both the algorithms and provides better exploration and exploitation operations. In order to have a balance between the exploration and exploitation operation, the breeding probability has been identified and validated. All the basic parameters of proposed SaDN have been made self-adaptive with scaling factor changing with respect to mutation weights, mating factor adapting as per inertia weights and crossover rate changing with respect to the total number of generations. The added hybridization along with Lévy mutation and sigmoidal inertia weights provides better searching capabilities. The proposed algorithm is them subjected to variable population and dimension sizes to prove its significance for higher dimension and find the best population size. Here a comparative analysis with respect to other recent state-of-the-art algorithms is presented for CEC 2005, CEC 2014 and CEC 2019 benchmark problems. Statistical tests in terms of Wilcoxon's ranksum test and Freidman's test have also been done to prove the significance of the presented results. Moreover, the proposed algorithm has also been tested on three mechanical engineering design problems to further validate the results on real world problems. It has been found that the proposed SaDN algorithm provides highly competitive results and can be considered as a potential candidate for future application to real world scenarios. As a future direction, the algorithm can be further enhanced by apply population size reduction, division of iteration, more balanced operations using fuzzy based adaptations, and other hybridization as well as general modifications. The SaDN is highly effective and can be applied to various domain research problems such as clustering, image segmentation, structural engineering design, antenna design, medical imaging, time series analysis, forecast modelling, feature selection and others. Apart from that more research can be done on the basic parameters of SaDN algorithm for high dimensional computing.

Compliance with Ethical Standards

Conflicts of Interest The authors declare that they have no conflict of interest. Informed Consent All procedures followed were in accordance with the ethical standards of the responsible committee on human experimentation (institutional and national) and with the Helsinki Declaration of 1975, as revised in 2008 (5). Additional informed consent was obtained from all patients for which identifying information is included in this article.

Human and Animal Rights This article does not contain any studies with human or animal subjects performed by the any of the authors.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] X.-S. Yang, S. Deb, Cuckoo search via lévy flights, in: 2009 World congress on nature & biologically inspired computing (NaBIC), IEEE, 2009, pp. 210–214.

[2] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in engineering software 95 (2016) 51–67.

[3] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Advances in Engineering Software 114 (2017) 163–191.

[4] R. Salgotra, U. Singh, S. Singh, G. Singh, N. Mittal, Self-adaptive salp swarm algorithm for engineering optimization problems, Applied Mathematical Modelling 89 (2020) 188–207.

[5] Z. Lv, R. Peng, A novel meta-matching approach for ontology alignment using grasshopper optimization, Knowledge-Based Systems (2020) 106050.

[6] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowledge-Based Systems 191 (2020) 105190.

[7] X.-S. Yang, Flower pollination algorithm for global optimization, in: International conference on unconventional computing and natural computation, Springer, 2012, pp. 240–249.

[8] L. Xia, L. Zhang, Q. Xia, T. Shi, Stress-based topology optimization using bi-directional evolutionary structural optimization method, Computer Methods in Applied Mechanics and Engineering 333 (2018) 356–370.

[9] W. Zhao, C. Du, S. Jiang, An adaptive multiscale approach for identifying multiple flaws based on xfem and a discrete artificial fish swarm algorithm, Computer Methods in Applied Mechanics and Engineering 339 (2018) 341–357.

[10] V. V. de Melo, W. Banzhaf, Drone squadron optimization: a novel self-adaptive algorithm for global numerical optimization, Neural Computing and Applications 30 (10) (2018) 3117–3144.

[11] S. Gholizadeh, M. Danesh, C. Gheyratmand, A new newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames, Computers & Structures 234 (2020) 106250.

[12] S. Gholizadeh, E. Salajegheh, Optimal design of structures subjected to time history loading by swarm intelligence and an advanced metamodel, Computer Methods in Applied Mechanics and Engineering 198 (37-40) (2009) 2936–2949.

[13] N. A. Kallioras, N. D. Lagaros, D. N. Avtzis, Pity beetle algorithm–a new metaheuristic inspired by the behavior of bark beetles, Advances in Engineering Software 121 (2018) 147–166.

[14] K. Kaur, U. Singh, R. Salgotra, An enhanced moth flame optimization, Neural Computing and Applications (2018) 1–35.

[15] R. Salgotra, U. Singh, A novel bat flower pollination algorithm for synthesis of linear antenna arrays, Neural Computing and Applications 30 (7) (2018) 2269–2282.

[16] X. Pan, L. Xue, R. Li, A new and efficient firefly algorithm for numerical optimization problems, Neural Computing and Applications 31 (5) (2019) 1445–1453.

[17] A. Sadollah, H. Sayyaadi, H. M. Lee, J. H. Kim, et al., Mine blast harmony search: a new hybrid optimization method for improving exploration and exploitation capabilities, Applied Soft Computing 68 (2018) 548–564.

[18] C. Wang, J. M. Koh, T. Yu, N. G. Xie, K. H. Cheong, Material and shape optimization of bi-directional functionally graded plates by giga and an improved multi-objective particle swarm optimization algorithm, Computer Methods in Applied Mechanics and Engineering 366 (2020) 113017.

[19] R. Salgotra, U. Singh, The naked mole-rat algorithm, Neural Computing and Applications 31 (12) (2019) 8837–8857.

[20] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: 2005 IEEE congress on evolutionary computation, Vol. 2, IEEE, 2005, pp. 1785–1791.

[21] J. Zhang, A. C. Sanderson, Jade: adaptive differential evolution with optional external archive, IEEE Transactions on evolutionary computation 13 (5) (2009) 945–958.

[22] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, K. M. Jambi, Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems, in: 2017 IEEE Congress on evolutionary computation (CEC), IEEE, 2017, pp. 145–152.

[23] R. Salgotra, U. Singh, S. Sharma, On the improvement in grey wolf optimization, Neural Computing and Applications (2019) 1–40.

[24] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, KanGAL report 2005005 (2005) 2005.

[25] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 635.

[26] K. Price, N. Awad, M. Ali, P. Suganthan, 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, 2018.

[27] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[28] R. Salgotra, U. Singh, Application of mutation operators to flower pollination algorithm, Expert Systems with Applications 79 (2017) 112–129.

[29] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary computation 3 (2) (1999) 82–102.

[30] K. Deb, D. Deb, Analysing mutation schemes for real-parameter genetic algorithms, International Journal of Artificial Intelligence and Soft Computing 4 (1) (2014) 1–28.

[31] R. Salgotra, U. Singh, S. Saha, New cuckoo search algorithms with enhanced exploration and exploitation properties, Expert Systems with Applications 95 (2018) 384–420.

[32] H.-Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, Journal of global optimization 27 (1) (2003) 105–129.

[33] S. Das, A. Abraham, U. K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 526–553.

[34] X.-S. Yang, Engineering optimization: an introduction with metaheuristic applications, John Wiley & Sons, 2010.

[35] J. Kennedy, Particle swarm optimization, Encyclopedia of machine learning (2010) 760–766.

[36] Y. Shi, Eberhart., A modified particle swarm optimizer, IEEE world congress on computational intelligence (2002) 69–73.

[37] M. F. W.Al-Hassan, S. Shaheen, Psosa: An optimized particle swarm technique for solving the urban planning problem, Computer Engineering and Systems (2007) 401–405.

[38] H. Li, Y. Gao, Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation, Second International Conference on Information and Computing Science (2009) 66–69.

[39] R. Salgotra, U. Singh, S. Saha, On some improved versions of whale optimization algorithm, Arabian Journal for Science and Engineering 44 (11) (2019) 9653–9691.

[40] M. F. W.Al-Hassan, S. Shaheen, A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation, Computer Engineering and Systems 1 (2008) 61–65.

[41] J. Pitono, A. Soeprijanto, T. Hiyama, Hybrid optimization of emission and economic dispatch by the sigmoid decreasing inertia weight particle swarm optimization, World of Science, Engineering and Technology 60 (2009) 315–320.

[42] R. F. Malik, T. A. Rahman, S. Z. M. Hashim, R. Ngah, New particle swarm optimizer with sigmoid increasing inertia weight, International Journal of Computer Science and Security 1 (2) (2007) 35–44.

[43] K. Kentzoglanakis, M. Poole, Particle swarm optimization with an oscillating inertia weight, in: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, 2009, pp. 1749–1750.

[44] J. K. Kordestani, A. Rezvanian, M. R. Meybodi, An efficient oscillating inertia weight of particle swarm optimisation for tracking optima in dynamic environments, Journal of Experimental & Theoretical Artificial Intelligence 28 (1-2) (2016) 137–149.

[45] R. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, Evolutionary computational 1 (2002) 94–100.

[46] G. c. J.xin, Y. Hai, A particle swarm optimizer with multistage linearly-decreasing inertia weight, Computational Sciences and Optimization 1 (2009) 505–508.

[47] D. H. Wolpert, W. G. Macready, et al., No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (1) (1997) 67–82.

[48] A. Faramarzi, M. Heidarinejad, S. Mirjalili, A. H. Gandomi, Marine predators algorithm: A nature-inspired metaheuristic, Expert Systems with Applications (2020) 113377.

[49] S. Khalilpourazari, S. H. R. Pasandideh, Sine–cosine crow search algorithm: theory and applications, Neural Computing and Applications (2019) 1–18.

[50] D. Yousri, M. Abd Elaziz, S. Mirjalili, Fractional-order calculus-based flower pollination algorithm with local search for global optimization and image segmentation, Knowledge-Based Systems (2020) 105889.

[51] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evolutionary computation 11 (1) (2003) 1–18.

[52] F. Wilcoxon, S. Katti, R. A. Wilcox, Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test, Selected tables in mathematical statistics 1 (1970) 171–259.

[53] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.

[54] V. Garg, K. Deep, Performance of laplacian biogeography-based optimization algorithm on cec 2014 continuous optimization benchmarks and camera calibration problem, Swarm and Evolutionary Computation 27 (2016) 132–144.

[55] S. Gupta, K. Deep, A novel random walk grey wolf optimizer, Swarm and evolutionary computation 44 (2019) 101–112.

[56] G. G. Tejani, V. J. Savsani, V. K. Patel, S. Mirjalili, Truss optimization with natural frequency bounds using improved symbiotic organisms search, Knowledge-Based Systems 143 (2018) 162–178.

[57] N. E. Humphries, N. Queiroz, J. R. Dyer, N. G. Pade, M. K. Musyl, K. M. Schaefer, D. W. Fuller, J. M. Brunnschweiler, T. K. Doyle, J. D. Houghton, et al., Environmental context explains lévy and brownian movement patterns of marine predators, Nature 465 (7301) (2010) 1066–1069.

[58] W. Li, G.-G. Wang, A. H. Alavi, Learning-based elephant herding optimization algorithm for solving numerical optimization problems, Knowledge-Based Systems (2020) 105675.

[59] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary computation 9 (2) (2001) 159–195.

[60] Z. Hongfeng, Dynamic economic dispatch based on improved differential evolution algorithm, Cluster Computing 22 (4) (2019) 8241–8248.

[61] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, Applied Soft Computing 27 (2015) 99–126.

[62] A. K. Bhandari, A novel beta differential evolution algorithm-based fast multilevel thresholding for color image segmentation, Neural Computing and Applications 32 (9) (2020) 4583–4613.

[63] S. Gupta, K. Deep, A. P. Engelbrecht, A memory guided sine cosine algorithm for global optimization, Engineering Applications of Artificial Intelligence 93 (2020) 103718.

[64] P. Singh, N. Mittal, An efficient localization approach to locate sensor nodes in 3d wireless sensor networks using adaptive flower pollination algorithm, Wireless Networks (2021) 1–16.

[65] R. Salgotra, U. Singh, G. Singh, S. Singh, A. H. Gandomi, Application of mutation operators to salp swarm algorithm, Expert Systems with Applications 169 (2021) 114368.

[66] G. Singh, U. Singh, R. Salgotra, Effect of parametric enhancements on naked mole-rat algorithm for global optimization, Engineering with Computers (2021) 1–29.

[67] J. M. Abdullah, T. Ahmed, Fitness dependent optimizer: inspired by the bee swarming reproductive process, IEEE Access 7 (2019) 43473–43486.

[68] P. Singh, N. Mittal, U. Singh, R. Salgotra, Naked mole-rat algorithm with improved exploration and exploitation capabilities to determine 2d and 3d coordinates of sensor nodes in wsns, Arabian Journal for Science and Engineering 46 (2) (2021) 1155–1178.

[69] J. Brest, M. S. Maučec, B. Bošković, The 100-digit challenge: Algorithm jde100, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 19–26.

[70] H. Muthusamy, S. Ravindran, S. Yaacob, K. Polat, An improved elephant herding optimization using sine–cosine mechanism and opposition based learning for global optimization problems, Expert Systems with Applications 172 (2021) 114607.

[71] A. Maesani, G. Iacca, D. Floreano, Memetic viability evolution for constrained optimization, IEEE Transactions on Evolutionary Computation 20 (1) (2015) 125–144.

[72] H. S. Bernardino, H. J. Barbosa, A. C. Lemonge, L. Fonseca, A new hybrid ais-ga for constrained optimization problems in mechanical engineering, in: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), IEEE, 2008, pp. 1455–1462.

[73] V. S. Aragón, S. C. Esquivel, C. A. C. Coello, A modified version of a t-cell algorithm for constrained optimization problems, International Journal for Numerical Methods in Engineering 84 (3) (2010) 351–378.

[74] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Engineering applications of artificial intelligence 20 (1) (2007) 89–99.

[75] E. Mezura-Montes, B. Hernández-Ocana, Bacterial foraging for engineering design problems: preliminary results, Laboratorio Nacional de Informática Avanzada (LANIA AC)-Universidad Juárez Autónoma de Tabasco. México.

[76] C. A. C. Coello, N. C. Cortés, Hybridizing a genetic algorithm with an artificial immune system for global optimization, Engineering Optimization 36 (5) (2004) 607–634.

[77] C. C. Coello, E. M. Montes, Use of dominance-based tournament selection to handle constraints in genetic algorithms, Intelligent Engineering Systems through Artificial Neural Networks 11 (1) (2001) 177–182.

[78] E. Mezura-Montes, C. A. C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, International Journal of General Systems 37 (4) (2008) 443–473.

[79] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Applied Mathematics and computation 186 (1) (2007) 340–356.

[80] T. Ray, P. Saini, Engineering design optimization using a swarm with an intelligent information sharing among individuals, Engineering Optimization 33 (6) (2001) 735–748.

[81] C. A. C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Computers in Industry 41 (2) (2000) 113–127.

[82] C. A. Coello Coello, R. L. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, Engineering Optimization 36 (2) (2004) 219–236.

[83] C. A. C. Coello, E. M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advanced Engineering Informatics 16 (3) (2002) 193–203.

[84] T. Ray, K.-M. Liew, Society and civilization: An optimization algorithm based on the simulation of social behavior, IEEE Transactions on Evolutionary Computation 7 (4) (2003) 386–396.

[85] H. S. Bernardino, H. J. Barbosa, A. C. Lemonge, A hybrid genetic algorithm for constrained optimization problems in mechanical engineering, in: 2007 IEEE congress on evolutionary computation, IEEE, 2007, pp. 646–653.

[86] S. Akhtar, K. Tai, T. Ray, A socio-behavioural simulation model for engineering design optimization, Engineering Optimization 34 (4) (2002) 341–354.

[87] J. Zhang, C. Liang, Y. Huang, J. Wu, S. Yang, An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization, Applied Mathematics and Computation 211 (2) (2009) 392–416.

[88] A.-R. Hedar, M. Fukushima, Derivative-free filter simulated annealing method for constrained continuous global optimization, Journal of Global optimization 35 (4) (2006) 521–549.

[89] S. He, E. Prempain, Q. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Engineering optimization 36 (5) (2004) 585–605.

[90] S.-F. Hwang, R.-S. He, A hybrid real-parameter genetic algorithm for function optimization, Advanced Engineering Informatics 20 (1) (2006) 7–21.