Neurodynamic Differential Evolution Algorithm and Solving CEC2015 Competition Problems

Karam M. Sallam, Ruhul A. Sarker, Daryl L. Essam and Saber M. Elsayed
School of Engineering and Information technology
University of New South Wales at Canberra
Australia
Emails: karam.sallam@student.adfa.edu.au;{r.sarker, d.essam, s.elsayed}@adfa.edu.au

***Abstract*—Recently, the success history based parameter adaptation for differential evolution algorithm with linear population size reduction has been claimed to be a great algorithm for solving optimization problems. Neuro-dynamic is another recent approach that has shown remarkable convergence for certain problems, even for high dimensional cases. In this paper, we proposed a new algorithm by embedding the concept of neuro-dynamic into a modified success history based parameter adaptation for differential evolution with linear population size reduction. We have also proposed an adaptive mechanism for the appropriate use of the success history based parameter adaptation for differential evolution with linear population size reduction and neuro-dynamic during the search process. The new algorithm has been tested on the CEC'2015 single objective real-parameter competition problems. The experimental results show that the proposed algorithm is capable of producing good solutions that are clearly better than those obtained from the success history based parameter adaptation for differential evolution with linear population size reduction and a few of the other state-of-the-art algorithms considered in this paper.**

***Index Terms*—evolutionary algorithms, neuro-dynamic, adaptive mechanism.**

I. Introduction

Optimization plays an important role in many practical decision making processes. For several decades, optimization has attracted the attention of many researchers and practitioners for solving planning, scientific and engineering design problems that arise in industry, public and private sectors. Optimization problems can be classified as unconstrained or constrained optimization problems. The optimization problems may contain different types of variables, such as: integer, real, discrete, continuous or mixed [1]. In constrained problems, they may have equality or inequality constraints, or both. The objective and constraint functions can be either linear or nonlinear, convex or non-convex, continuous or discontinuous, and uni-modal or multimedia. The feasible region of such problems could be tiny compared to the search space, and it could be either one single bounded region or a collection of multiple disjointed regions. The optimal solution may or may not exist on the boundary of the feasible region of a constrained problem [2].

In practice, both conventional optimization techniques [3] and evolutionary algorithms (EAs) [4] have been applied for solving optimization problems. There are some common difficulties with most traditional optimization methods. Firstly, the convergence to an optimal or near optimal solution depends on the initial solution. Secondly, these methods require the satisfaction of specific mathematical properties (such as convexity, continuity and differentiability). Thirdly, some algorithms tend to get stuck with a suboptimal solution. Fourthly, they may require simplification of a problem by making various assumptions for the convenience of mathematical modeling [5]. Additionally, they are not good choices for applications when real-time solutions are needed. In contrast, EAs are simple in concept and implementation. They start with a population of initial solutions that is evolved until an optimal or near optimal solution can be found, and here, the final solution does not really depend on the initial solution. Also, they do not require the satisfaction of specific mathematical properties, are flexible to dynamic changes, can handle the evaluation of each solution in parallel, have the capability for self-organization and have broader applicability in practice [6].

In the recent past, a new approach, known as the neuro-dynamic (ND) optimization method, has been introduced for solving optimization problems, such as linear, nonlinear and quadratic [7]. ND is a kind of recurrent neural network that has no weights, and it consists of an input and an output layer. The number of neurons in the input layer equals the number of variables for the optimization problem and there is a feedback mechanism from the output layer to the input layer. It is a parallel computation method that converges very fast to the optimal solution for certain problems, such as unimodal, convex, and separable. The convergence rate is even very high for high dimensional problems. The algorithm L-SHADE [8], that is SHADE [9] (Success History Based Parameter Adaptation for Differential Evolution) with Linear Population Size Reduction, is an improved version of SHADE that incorporates a linear reduction resizing method to reduce the population size. SHADE uses a historical memory to store a set of crossover rates and control parameters that performed well in the past generations, and generates new values by selecting random pairs from the history, and applies an adaptive mechanism. L-SHADE showed very good performance in solving the CEC'2014 competition problems.

In this research, we use modified L-SHADE combined with

ND. Historically, EAs are capable of solving a wide range of optimization problems, but their convergence becomes slow after some generations, and the quality of solution deteriorates with the increase of the size of problems. On the other hand, ND converges very fast, but it is only suitable for certain types of problems. So, if we can appropriately embed ND's fast convergence feature with L-SHADE, it is possible to design a new powerful algorithm, that could be effective and efficient for a wide range of practical problems. However, there is a challenge to run ND for non-differential functions, so we use a technique to calculate the approximate gradient numerically. In the new methodology, we have also developed an adaptive mechanism for the appropriate use of the two chosen algorithms, based on the rate of improvement made in each generation. The performance of the proposed algorithm has been tested on a set of 15 unconstrained test problems [10], those are CEC'2015 single objective real-parameter competition problems, which have different mathematical properties, with 10, 30, 50 and 100 dimensions. This competition has two stages, the first stage is the training and the second is the testing stage, and this is similar to what proposed in [11]. From the results, the proposed algorithm shows consistent performance in obtaining good quality solutions, and the results are better than the same of L-SHADE and SHADE, as well as the two other state-of-the-art algorithms considered in this paper.

The rest of this paper is organized as follows. In Section II, basic algorithms and operators are reviewed. Section III presents a new algorithm, where the concept of ND is embedded with L-SHADE. The simulation results on benchmark problems are provided in Section IV. Finally, Section V provides the conclusions of this paper.

II. Basic Algorithms

In this section, we will discuss a few basic EAs and their search operators. In addition, the ND approach will be discussed.

A. Basic Evolutionary Algorithms

EAs are population based algorithms where a set of individuals evolve in finding the optimal solution. These algorithms use different operators, such as crossover, and mutation and different parameters, such as crossover rate, mutation rate, and population size [12] [13]. EAs have the ability to solve different kinds of problems regardless of their mathematical properties. However, they have some drawbacks, such as there is no guarantee that an evolutionary algorithm will find the optimal solution to a given problem. In the literature, there are a good number of population based algorithms for solving practical optimization problems. Some of the widely used algorithms, are: genetic algorithm (GA), differential evolution (DE). These algorithms have their own advantages and disadvantages.

DE is a popular EA because it usually converges fast, simple in implementation and the same settings can be used for many different optimization problems. As of the literature, DE showed good performance in comparison to several other EAs on a wide variety of problems [14]. DE algorithm contains the following operations: mutation, crossover and selection.

A. Success-History based Adaptive Differential Evolution with Linear Population Size Reduction (L-SHADE)

This section describes L-SHADE [8], which is an improved version of SHADE [9]. SHADE is an enhancement of JADE [15], which is a well-known EA that computes its control parameters by using an adaptation mechanism.

To improve the convergence performance of DE, Zhang and Sanderson [15] proposed a new mutation strategy 'DE/current-to-pbest with/without external archive', they also developed an adaptive technique to control parameters. Their proposed method is called JADE. JADE uses binomial crossover and the one-to-one selection scheme ,as do many other classical DE variants. The mutation strategy used by JADE is current-to-pbest/1, which is a variant of the current-to-best/1, strategy and it is computed using equation 1.

$$\overrightarrow{u}_{z,i} = \overrightarrow{x}_{z,i} + F_z(\overrightarrow{x}_{pbest,i} - \overrightarrow{x}_{z,i}) + F_z(\overrightarrow{x}_{r_1,i} - \overrightarrow{x}_{r_2,i}) \quad (1)$$

In equation 1, $\overrightarrow{x}_{pbest,i}$ is randomly selected from the top $PopSize \times P$ individuals in the i^{th} generation, where P is a random number between $[0, 1]$. $\overrightarrow{x}_{r_2,i}$ is selected from the union of the population and the archive. The binomial crossover is one that is used in this method and is calculated as follows:

$$\overrightarrow{v}_{z,j} = \begin{cases} \overrightarrow{u}_{z,j} & \text{if } (rand \leq Cr \text{ or } j = j_{rand}) \\ \overrightarrow{x}_{z,j} & otherwise \end{cases} \quad (2)$$

where $j_{rand} \in 1, 2, ..., n$ is a randomly selected index, which ensures $\overrightarrow{v_z}$ gets at least one component from $\overrightarrow{u_z}$.

JADE uses an external archive to maintain diversity. At first, the archive is empty, and after each generation, the parent solutions that are not selected are added to the archive. When the size of the archive exceeds a certain threshold, (i.e $PopSize$), then some individuals are deleted randomly from the archive [8].

Instead of generating new parameters, based on using a single pair of parameters μCR and μF, SHADE uses a history based parameter adaptation method. The authors [8] used historical memories MCR and MF which store sets of CR and F values that have performed well in the past, then generate new CR and F by selecting random pairs from the history. Initially, the values MCR and MF were set at a value of 0.5, and then at the beginning of each generation the values of CR and F were set by selecting an index r_z randomly from the history. Then their values were computed using equations 3 and 4:

$$CR_z = randn_z(MCR_{r_z}, 0.1) \quad (3)$$

$$F_z = randc_z(MF_{r_z}, 0.1) \quad (4)$$

where $randn_z(\mu, \sigma^2)$, and $randc_z(\mu, \sigma^2)$ are values that are selected randomly from a normal and Cauchy distribution, respectively, with mean μ and variance σ^2. CR_z and F values that were used by successful individuals in generating trial vectors, were stored in SCR and SF. At the end of the generation, the values of MCR and MF were both initialized at a value of 0.5 and were then calculated by the self adaptive formula in equations, 5 and 6:

$$MCR_{m,i+1} = \begin{cases} meanw_A(SCR) & \text{if } SCR \neq \phi \\ MCR_{m,i} & otherwise \end{cases} \tag{5}$$

$$MF_{m,i+1} = \begin{cases} meanw_L(SF) & \text{if } SF \neq \phi \\ MF_{m,i} & otherwise \end{cases} \tag{6}$$

In equations 5 and 6, m is the position in the history, and its value lies between 1 and the size of the history H ($1 \leq m \leq H$). $meanw_A$is the weighted mean and $meanw_L(SF)$ is the Lehmer mean and is computed using equation 7.

$$meanw_L(SF) = \frac{\sum_{m=1}^{|SF|} w_z.SF_m^2}{\sum_{m=1}^{|SF|} w_z.SF_m} \tag{7}$$

The L-SHADE algorithm used a linear population size reduction (LPSR) [8] to dynamically resize the population during a SHADE run. LPSR reduces the population linearly as a function of fitness evaluations. $N_{i+1}$is computed by using equation 8.

$$N_{i+1} = round[(\frac{N^{min} - N^{init}}{FES_{max}}) \text{X} FES + N^{init}] \tag{8}$$

where N^{min} is the smallest number of individuals that the L-SHADE operator can be applied to (i.e $N^{min} = 4$). FES is the current number of fitness evaluations, FES_{max} is the maximum of fitness evaluations.

Algorithm 1 describes the main steps of L-SHADE.

Algorithm 1 L-SHADE algorithm

```
Initialization
  • Generate a population of size $PopSize$ randomly, and each variable is generated inside its boundary $[l, h]$;
  • Set $PopSize = N_i = N^{init}$; $A = \{\}$;
  • Set the values of $MCR$, $MF$ to 0.5;
  • Set $i = 1$
Main loop
  • Set $SCR = \phi$ and $SF = \phi$.
  • Compute $MCR$, and $MF$ using equations 5, 6, and 7
  • Mutation: Generate a mutant vector $\overrightarrow{v}_{z,i}$ using equation 1.
  • Crossover: Use binomial crossover to generate the trial vector $\overrightarrow{u}_{z,i}$.
  • Selection: if $(f(\overrightarrow{u}_{z,i}) \leq f(\overrightarrow{x}_{z,i}))$ then $\overrightarrow{x}_{z,i+1} = \overrightarrow{u}_{z,i}$, else $\overrightarrow{x}_{z,i+1} = \overrightarrow{x}_{z,i}$,
  • Update Archive: if $(f(\overrightarrow{u}_{z,i}) \leq f(\overrightarrow{x}_{z,i}))$ then $A \leftarrow \overrightarrow{x}_{z,i}$; $SCR \leftarrow CR_{z,i}$; and $SF \leftarrow F_{z,i}$;
  • Linear population size reduction: Calculate $N_{i+1}$using equation 8.
    – if $N_i < N_{i+1}$then
      ∗ sort all individuals based on their fitness values and delete the lowest $N_i - N_{i+1}$individuals;
      ∗ Resize archive size .
  • Set $i = i + 1$;
```

B. Neuro-Dynamic Algorithm

Dynamical systems and neural network approaches have been used for solving optimization problems for more than three decades. In these algorithms, a common feature is a continuous path that starts from an initially generated point and that converges to an optimal or near optimal solution. In the dynamical system approach, it is required to convert an optimization problem into a set of ordinary differential equations (ODE), so that the solution of the optimization problem corresponds to an equilibrium point of the dynamical system [16]. The main steps of this method are: a) establish an ODE system, b) study the convergence of the ode solution, and c) solve the ODE system numerically. The principal merit of the dynamical system approach is that, based on optimization methods, many dynamical systems can be constructed. In considering the neural network approach, Hopfield introduced an artificial neural network (HNN) for solving the traveling salesman problem [17]. The mathematical representation of HNN approach was an ordinary differential equation; also, there was an energy function which was a Lyapunov function [17]. The most attractive feature of this approach was that the solution of an optimization problem could be obtained online in real-time by constructing an electrical circuit which represented an artificial neural network. The main advantage of the neural network approach was the formulation of an energy function which decreased monotonically in time [16]. The HNN model consists of the following components [16]:

1) A Lyapnouv function as an energy function;
2) A neural network which is in the form of an ode; and
3) A hardware implementation, or an architectural neural network diagram with computer simulation.

After the work done by Hopfield and Tank[17] on using ND for solving the TSP problem, ND has received increased attention for solving optimization problems. The HNN model was extended by Kennedy and Chua [18] for solving nonlinear programming models, and it was capable of solving a large class of optimization problems. However, the method contained a finite penalty parameter and approximate solutions were generated. To overcome this shortcoming, Rodriguez-Vazquezet *et al.*[19] proposed a switched capacitor neural network for solving optimization problems, but the model was only appropriate, where the optimal solution located inside the feasible region. Maa and Shanblatt [20] proposed a two-phase method that was converged to the exact solution; however it was relatively complex and some parameters tuning were requited. To overcome the weakness of two-phase optimization methods, Xia [21] introduced a new type of neural network for solving linear programming problems that deal with primal and dual problems without needing parameter tuning. Lillo *et al.* [22] proposed a neural network architecture that used the penalty function approach to form a new network that was capable of solving constrained optimization problems. Liang and Wang [23] proposed a continuous-time RNN for optimizing any nonlinear programming problem, in which the objective function should be continuous and differentiable, and the constraints should be bounded. Their method was used to solve quadratic optimization problems with bound constraints as a particular type of optimization problem. Leung *et al.* [24] proposed a neural network for solving convex nonlinear programming problems. The proposed neural network was different from the existing methods, such that no dual variables being required, and penalty parameters or Lagrange multipliers were involved in the proposed network. Therefore, it had only a few state variables, simple structure, and much better convergence rate. Wang *et al.* [25] introduced a one layer recurrent neural network for solving nonlinear convex programming problems in which the constraints were linear. In the literature, there was some limited research studies that combine EAs and the ND model. Wang *et al.* [26] proposed a collective Neuro-dynamic approach within the framework of PSO for solving non-convex optimization problems with variable bounds. The proposed approach was implemented in a manner that mimics the brainstorming process of PSO. However, they did not compare their computational results with other approaches. The main disadvantage of the neural network based approach for solving optimization problems is that it can be trapped easily in local minima. So Wang *et al.* [27] proposed a discrete Hopfield neural network (DHNN), combined with an estimation of distribution algorithm (EDA), for solving combinatorial optimization problems. Once its network was trapped in a local minima, a perturbation based on the EDA could generate a new starting point for the DHNN for further search. However, they also ignored the computational effort of their approach.

The neuro-dynamic approach combines the merits from both dynamical systems and the neural network approach. It suggests a systematic methodology on how to construct a neuro-dynamical system for both unconstrained and constrained optimization problems. Algorithm 2 shows how the neuro-dynamic approaches work.

The following formula is used to compute the projection value. This is used to handle boundaries for ND

$$P_\Omega(u_i) = \begin{cases} l_i & \text{if } u_i < l_i \\ u_i & \text{if } l_i \le u_i \le h_i \\ h_i & \text{if } u_i > h_i \end{cases} \tag{9}$$

A black-box optimization problem maybe noncontinuous or non-differentiable, so we use the numerical gradient instead of the exact gradient, which is calculated by equation 12:

Algorithm 2 The steps for the Neurodynamic approach

```
Step 1: generate an initial random solution $\overrightarrow{x}$
Step 2: calculate the objective value $f(\overrightarrow{x})$
Step 3: compute the gradient $\nabla f(\overrightarrow{x})$
Step 4: construct the ordinary differential equation (ode) system using $\frac{d\overrightarrow{x}}{dt} = \lambda\,[-\overrightarrow{x} + P_\Omega(\overrightarrow{x} - \alpha\nabla f(\overrightarrow{x}))]$ ,
where $\lambda > 0, \alpha > 0$ are the scaling and step size parameters, $P_\Omega$ is a projection operator calculated by equation9.
Step 5: solve the constructed ode system using ODE solvers, such as Euler method, Runge-kutta method or exact solution, and return $\overrightarrow{x}$
Step 6: if a pre-defined stopping criterion is met, then stop; else go to step 2.
```

$$\frac{df(\overrightarrow{x})}{dx_i} \cong \frac{f(x_i + \delta) - f(x_i)}{\delta} \tag{10}$$

III. The Proposed Algorithm

EAs and ND have their own strengths and weaknesses. As discussed earlier, ND converges very fast for certain types of problems, while EAs are able to solve a wide range of optimization problems, but with a low convergence rate in comparison with ND. So in order to utilize the strengths of both methods, a new algorithm is developed by combining the good features of ND and a well-know EA algorithm (LSHADE) after modifying it, so that the algorithm can solve a wide range of practical problems effectively and efficiently. The algorithm is named LSHADE-ND. In LSHADE-ND, a self-adaptive mechanism that helps to determine when to run either ND or LSHADE during the evolutionary process. This mechanism is based on the amount of improvement in the fitness value the algorithm achieves at each generation, such that:

$$diff = FV_c - FV_p \tag{11}$$

$$P_{ND} = max(0.01, \frac{diff}{FV_p}) \tag{12}$$

$$P_{LSHADE} = 1 - P_{ND} \tag{13}$$

where FV_c is the best fitness value at the current generation, FV_p the best fitness value at the previous generation, and P_{ND} and P_{LSHADE} are the probability values to apply ND or LSHADE, respectively.

Two modifications have been applied to LSHADE, to improve its performance

- The calculation of CR_z is computed as follows, in which a random number is generated and if it is less than a predefined threshold (here it is equal to 0.8), the value of the $CR_z$is computed using the Gaussian distribution, otherwise the value is computed using Cauchy distribution.

$$CR_z = \begin{cases} randn_z(MCR_{r_z}, 0.1) & \text{if } rand < 0.8 \\ randc_z(MCR_{r_z}, 0.1) & otherwise \end{cases}, \tag{14}$$

Algorithm 3 The framework of the proposed method

```
Initialization:
  • Generate a population of size $PopSize$ randomly, and each variable is generated inside its boundary $[l, h]$;
  • Set $PopSize = N_i = N^{init}$, $A = \{\}$;
  • Set the values of $MCR$, $MF$ to 0.5,
  • Set $i = 1, P_{ND} = 1,\ FES_{max} = 10,000 \times D$
Evaluations:
  • Compute the fitness values for each individuals in the population
  • Select the best individuals and pass it to ND
Adaptive Mechanism:
  • Generate random number ($rnd$) between [0,1],
  • if $rnd < P_{ND}$and $FES \leq FES_{ND}$ then
      – Apply ND on the best solution from the entire population.
      – Replace the best five solutions in entire population with the best five solution obtained from ND.
      – Compute $P_{ND}$ using equations 11,12, and 13
  • else
      – Apply the modified L-SHADE
      – Pass the best solution from L-SHADE to ND
  • set $i = i + 1$
```

- A linear reduction to compute the value of the memory size is conducted, in which the memory size is set at value of 100, at the beginning, and then it is linearly reduced to reach 5.

It is believed that at one generation ND may perform bad, and may perform good in another stage of the search process. Bearing this in mind, in equation 12, if $diff = 0$, a minimum probability value is assigned to P_{ND}, here it is 0.01, so that a possibility to apply ND still exists. It is worthy to mention here that ND considers only the best solution from the entire population to be evolved. Algorithm 3 shows the general framework of the proposed method.

In ND, the selection of ODE solver is crucial. Using a built-in Matlab function such as: ODE45, ODE15, ODE23 and ODE113 is very common. However, we noticed that these methods may consume many fitness evaluations in solving some types of optimization problems. Hence, in this research, the exact solution method [28] is used to update the values of the decision variables, such that:

$$\overrightarrow{x}_{t_{new}} = e^{-(\triangle t)}\overrightarrow{x}_{t_{old}} + (1 - e^{-(\triangle t)})P_{\Omega}[\overrightarrow{x}_{t_{old}} - \nabla f(\overrightarrow{x}_{t_{old}})] \tag{15}$$

where P_{Ω} is the projection operator used to handle the decision variables boundaries, and it is calculated using equation 9; and $\triangle t$ is the time step which is the difference between current t and previous t. The value of the time step is calculated using equation 16.

$$\triangle t = t_{new} - t_{old} \tag{16}$$

Based on our empirical analysis, we set $\triangle t$ at a value of 1.

Note that in this paper the maximum value of t is 20. As a consequence, 20 solutions are generated (each one represents a solution at a time period t). To get the benefit of those 20 solutions, the best five solutions are selected and compared with those best five solutions in the entire population. A tournament selection takes place, and the winner is survived to the next generation.

IV. Experimental Results

In this section, the performance of the proposed algorithm is tested by solving a set of problems taken from the CEC2015 competition on learning-based real-parameter single objective optimization [10]. The CEC2015 benchmark test set contains 15 test problems. The search space for all the problems is $[-100, 100]$. Functions$F01 \sim F02$ are unimodal, $F03 \sim F5$ are simple multimedia functions, $F06 \sim F08$ are hybrid functions, and $F09 \sim F15$ are composite functions which combine multiple test problems into a complex landscape. The proposed algorithm was run following the guidelines of the CEC2015 competition [10], that required 51 independent runs for each test problem with up to $10,000D$ fitness evaluations. In the experimentation, if the deviation of fitness value from the optimal is less than or equal to $1.0e-8$, it was considered zero. The algorithm was coded using Matlab R2014a, and was run on a PC with a 3.4 GHz Core I7 processor with a 16 GB RAM, and windows 7.

A. Algorithm Parameters

In reference to [8], the algorithm parameters are set as follows: N^{init} is set at a value of $18n$, where n is the number of decision variables, A 1.4, for H, we use a linear reduction method to compute it; for the current-to-pbest/1 and P is set at 0.11. For ND, only one parameter for calculating the numerical gradient is used, which is delta δ, and its range was$\in [1.0e-08 - 1.0e-01]$. Based on empirical study, δ is set at a value of $1.0e-02$ for all problems. ND is possible to run up to a predefined level (FES_{ND}) which is set at a value of $(\frac{2}{3} \times FES_{max})$, this value is based on our empirical study.

B. Results for 10D

The computational results $(f(x_{best}) - f(x^{*}))$ of LSHADE-ND for the $10D$ problems are shown in Table I. From the results obtained, it is clear that LSHADE-ND performed very well for the unimodal problems ($F01$ and $F02$), the algorithm obtained the optimal solution in regard to the best and mean solutions. For the multimedia problems ($F03 - F05$), it was able to obtain the optimal solution for $F03$ and close values to the optimal solutions for $F04$ and $F05$ in regard to the best solutions. However, for the mean results, it provided near optimal values to the optimal solutions. In the case of the hybrid functions ($F06 - F08$), the algorithm obtained the optimal solution for $F06$ and $F07$ in regard to the best result; however in regard to the average results, it provided a very close value to the optimal solution. For $F08$, the algorithm provided very close values to the optimal solutions.

Table I
RESULTS FOR 10D

	Best	Worst	Median	Mean	st.dev
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	**0.0000E+00**	2.0020E+01	2.0002E+01	1.6082E+01	8.0215E+00
F04	9.9496E-01	4.9748E+00	2.9849E+00	2.7703E+00	9.8121E-01
F05	2.4982E-01	1.2204E+02	7.0173E+00	1.2486E+01	1.8063E+01
F06	**0.0000E+00**	2.6143E+00	4.1629E-01	6.1416E-01	6.5245E-01
F07	**0.0000E+00**	9.3356E-01	3.6544E-02	5.5299E-02	1.2892E-01
F08	4.5244E-06	8.6137E-01	3.1264E-01	3.0355E-01	2.5852E-01
F09	1.0010E+02	1.0022E+02	1.0014E+02	1.0014E+02	2.5251E-02
F10	2.1654E+02	2.1654E+02	2.1654E+02	2.1654E+02	**0.0000E+00**
F11	7.1239E-01	3.0008E+02	2.1361E+00	1.2442E+02	1.4843E+02
F12	1.0034E+02	1.0126E+02	1.0071E+02	1.0073E+02	2.2200E-01
F13	3.0424E-02	3.0791E-02	3.0530E-02	3.0504E-02	6.2835E-05
F14	1.0000E+02	6.9948E+03	2.9345E+03	4.6487E+03	1.9886E+03
F15	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	**0.0000E+00**

Table II
RESULTS FOR 30D

	Best	Worst	Median	Mean	st.dev
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	2.0000E+01	2.0135E+01	2.0036E+01	2.0046E+01	3.6110E-02
F04	4.9750E+00	1.7909E+01	1.0945E+01	1.0841E+01	2.8854E+00
F05	7.5217E+02	1.8245E+03	1.2726E+03	1.2799E+03	2.5717E+02
F06	4.4798E+01	4.0962E+02	1.8250E+02	1.7380E+02	7.5535E+01
F07	3.6485E+00	7.3566E+00	5.1978E+00	5.0576E+00	8.0555E-01
F08	2.3365E+00	1.4619E+02	2.7406E+01	3.2389E+01	2.7103E+01
F09	1.0221E+02	1.0278E+02	1.0256E+02	1.0254E+02	1.2557E-01
F10	3.3222E+02	8.3164E+02	5.9810E+02	5.9533E+02	1.1493E+02
F11	4.0000E+02	4.4260E+02	4.0000E+02	4.0501E+02	1.3862E+01
F12	1.0295E+02	1.0418E+02	1.0349E+02	1.0350E+02	2.4624E-01
F13	2.5584E-02	2.7150E-02	2.5892E-02	2.5993E-02	3.4691E-04
F14	3.1070E+04	3.3532E+04	3.2411E+04	3.2187E+04	1.0502E+03
F15	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	**0.0000E+00**

Table III
RESULTS FOR 50D

	Best	Worst	Median	Mean	st.dev
F01	8.3503E+01	9.5705E+03	1.7466E+03	2.1192E+03	2.0795E+03
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	2.0000E+01	2.0038E+01	2.0002E+01	2.0005E+01	7.3266E-03
F04	9.9686E+00	3.7811E+01	2.0089E+01	2.1481E+01	6.8383E+00
F05	2.0854E+03	3.9092E+03	3.0262E+03	3.0053E+03	3.4951E+02
F06	5.3644E+02	2.3648E+03	1.6147E+03	1.6709E+03	4.1502E+02
F07	4.6974E+00	4.2111E+01	3.9850E+01	3.4763E+01	1.1868E+01
F08	1.0974E+01	7.1816E+02	2.6315E+02	2.6880E+02	1.4394E+02
F09	1.0366E+02	1.0429E+02	1.0393E+02	1.0394E+02	1.2485E-01
F10	8.0372E+02	1.4333E+03	9.7905E+02	1.0153E+03	1.5383E+02
F11	4.0000E+02	4.4939E+02	4.1100E+02	4.1210E+02	8.4109E+00
F12	1.0429E+02	2.0027E+02	1.0516E+02	1.1815E+02	3.3082E+01
F13	7.1016E-02	8.3831E-02	7.7432E-02	7.7568E-02	3.9649E-03
F14	4.9516E+04	7.3063E+04	6.8480E+04	6.3670E+04	8.6931E+03
F15	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	**0.0000E+00**

Considering the composition functions ($F09 - F15$), the algorithm provided near optimal solutions for $F13$, however it was difficult to reach the optimal solution for the remaining problems.

C. Results for 30D

The computational results of the proposed algorithm for the $30D$ test problems are shown in Table II. From the results, LSHADE-ND provided the optimal solution for the unimodal functions. For the multimedia functions, it was able to reach near optimal solution for $F03$ and $F04$, while for $F05$ the result obtained from the proposed algorithms is far from the optimal solution. For the hybrid functions, the algorithm obtained near optimal solutions for $F07$ and $F08$ in regard to both the best and mean results, but the result was far from the optimal for $F06$. In case of the composition functions, the proposed algorithm was able to achieve very close results to the optimal solutions for $F13$, but it was difficult to obtain good results for the remaining problems.

D. Results for 50D

Table III shows the computational results for the $50D$ problems. From table III, LSHADE-ND provided the optimal solution for $F02$ and it was able to reach near optimal for $F01$. For the multimedia functions, it was able to reach near optimal for two of them

+($F03$ and $F04$), but it was not easy to get the optimal for $F05$. For the hybrid functions, the algorithm obtained near optimal solutions for $F07$ and $F08$, but the results were far from the optimal for $F06$. For the composition functions, the proposed algorithm was able to obtain near optimal solutions for $F13$, but it was hard to obtain good results for the remaining problems.

E. Results for 100D

Table IV shows the computational for the $100D$ problems. From the results, LSHADE-ND provided the optimal solution for one unimodal functions ($F02$), but it was not easy to get the optimal for $F01$. For the multimedia functions, it was able to reach near optimal for $F03$ and $F04$, but it was difficult to obtain the optimal for $F05$. For the hybrid functions, the algorithm obtained near optimal solutions for $F07$, but was far from the optimal for $F06$ and $F08$. Considering the composition functions, LSHADE-ND was able to obtain close results to the optimal solution for $F13$, but its performance was not good for the remaining problems.

F. Complexity of the algorithm

To this end, the computational complexity of the proposed algorithm was calculated based on all the problem dimensions following the guidelines in [10]. A summary of the results is shown in Table V. From table V, the computational time is reasonably small and is linearly increased in reference to the problem dimensions.

G. Comparison with other state-of-the-art algorithms

To judge the performance of LSHADE-ND, we have compared it with four state-of-the-art algorithms, based on the solutions of CEC'2015 competition problems. The algorithms are: LSHADE [8], SHADE [9], UMOEAs [29], and JADE [15]. We have discussed JADE, SHADE and LSHADE in an earlier section. UMOEA is a united multi-operator based

Table IV
RESULTS FOR 100D

	Best	Worst	Median	Mean	st.dev
F01	1.0567E+05	3.3620E+05	2.0516E+05	2.1089E+05	6.2564E+04
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	2.0000E+01	2.0019E+01	2.0000E+01	2.0001E+01	3.4587E-03
F04	2.3879E+01	2.0099E+02	4.9725E+01	7.3379E+01	4.5409E+01
F05	9.5541E+03	1.3350E+04	1.1550E+04	1.1502E+04	7.6655E+02
F06	3.7779E+03	6.1532E+03	5.0947E+03	5.0588E+03	5.7630E+02
F07	2.2173E+01	1.3618E+02	9.8064E+01	9.0062E+01	2.9626E+01
F08	1.4584E+03	3.5144E+03	2.4524E+03	2.4183E+03	5.1533E+02
F09	1.0570E+02	1.0684E+02	1.0629E+02	1.0630E+02	2.0623E-01
F10	3.1667E+03	5.2661E+03	3.8933E+03	4.0245E+03	5.4148E+02
F11	4.0560E+02	5.8692E+02	4.7437E+02	4.8499E+02	4.8021E+01
F12	1.1160E+02	2.0041E+02	1.1201E+02	1.2243E+02	2.8758E+01
F13	6.0644E-02	6.4745E-02	6.2682E-02	6.2596E-02	1.0701E-03
F14	1.0883E+05	1.0891E+05	1.0888E+05	1.0887E+05	1.7782E+01
F15	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	**0.0000E+00**

Table V
COMPUTATIONAL COMPLEXITY

	T_0	T_1	$\hat{T_2}$	$\frac{(\hat{T_2}-T_1)}{T_0}$
D=10	0.1051	0.1002	0.2864	1.7716
D=30		0.3822	0.9358	5.2674
D=50		0.8082	2.2225	13.4567
D=100		2.3527	5.2628	27.6889

evolutionary algorithm that was the top competitor in the CEC'2014 real parameter single objective optimization competition. We have executed these algorithms using the same parameters suggested by the authors in their papers and the other conditions are the same as the competition guides. Table VI shows a comparison summary of the results obtained from LSHADE-ND and the other four algorithms for $10D$, $30D$, $50D$, and $100D$ problems. In table VI last column(+ means there is a significant difference between the proposed algorithm and the other algorithm, and $\approx$ means there is no significant difference).

From the results, it is clear that our proposed algorithm outperformed all algorithms for most of the test functions. Based on the Wilcoxon test [30], the performance of the proposed algorithm was statistically better than LSHADE, SHADE, and JADE in regard to the mean results for the 10D, while there was no significant difference between the proposed algorithm in regard to the best results for the $10D$. Also, there was no significant difference between the proposed algorithm and UMOEAs in regard to the best and mean results in $10D$. For $30D$, LSHADE-ND was statistically better than JADE and SHADE in regard to the best and mean results. For LSHADE, there was no significant difference in regard to the best solution, but LSHADE-ND showed its superiority in regard to the mean values obtained. For UMOEAs there was no significant difference in both the best and mean results in $30D$. For $50D$, LSHADE-ND was statistically better LSHADE, SHADE and JADE in reference to the best and mean results, while there was no statistical difference in comparison with UMOEAs. For the $100D$, we noticed that there was statistical difference between LSHADE-ND and SHADE, considering the average results, and JADE based on both the best and average results achieved.

Furthermore, we have ran all algorithms on the same computer and recorded the computation time of each algorithm. The average computational times are reported in table VII. From the results obtained, it is clear that our proposed algorithm outperforms all algorithms considering all dimensions.

V. CONCLUSIONS

EAs are capable of solving a wide range of unconstrained and constrained optimization problems, but they may suffer from low convergence rates to the optimal solution. Recently, the computational intelligence field has witnessed the birth and growth of ND optimization methods for solving many optimization problems (linear, nonlinear and quadratic). ND has the ability to converge fast to the optimal solutions for certain types of problems. To utilize the strength of both methods, in this research, we embedded ND with modified L-SHADE by using a self-adaptive mechanism that was able to appropriately chose among both methods.

The performance of the proposed algorithm was tested on the CEC'2015 single objective real-parameter competition problems, which have different mathematical properties, with 10, 30, 50 and 100 dimensions. From the results obtained, the proposed algorithm showed consistent performance to achieve good quality solutions, and the results were better than other state-of-the-art algorithms considered in this paper.

In the future work, we will develop an approximate model to reduce the number of fitness evaluations that is consumed by ND, and hence to reduce the fitness evaluations consumed by the proposed algorithm.

REFERENCES

[1] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Memetic multi-topology particle swarm optimizer for constrained optimization," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[2] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers & operations research*, vol. 38, no. 12, pp. 1877–1896, 2011.

[3] K.Deb, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[4] L. Rutkowski, *Computational intelligence: methods and techniques*. Springer, 2008.

[5] R. Sarker, J. Kamruzzaman, and C. Newton, "Evolutionary optimization (evopt): a brief review and analysis," *International Journal of Computational Intelligence and Applications*, vol. 3, no. 04, pp. 311–330, 2003.

[6] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial intelligence through simulated evolution," 1966.

[7] Y. Xia and J. Wang, "A general projection neural network for solving monotone variational inequalities and related optimization problems," *Neural Networks, IEEE Transactions on*, vol. 15, pp. 318–328, 2004.

[8] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1658–1665.

[9] R. Tanabe and A. Fukunaga, "Evaluating the performance of shade on cec 2013 benchmark problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 1952–1959.

[10] J. Liang, B. Qu, P. Suganthan, and Q. Chen, "Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization," *Computational Intelligence Laboratory*, 2014.

[11] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Training and testing a self-adaptive multi-operator evolutionary algorithm for constrained optimization," *Applied Soft Computing*, vol. 26, pp. 515–522, 2015.

Table VI
A SUMMARY BETWEEN LSHADE-ND AND OTHER STATE-OF-THE-ART ALGORITHMS

	LSHADE-ND		Better	Equal	Worse	Dec.
10D	vs. **LSHADE**	Best	9	4	2	≈
		Mean	9	3	3	+
	vs. **SHADE**	Best	10	3	2	≈
		Mean	8	6	1	+
	vs. **UMOEAs**	Best	4	6	5	≈
		Mean	8	3	4	≈
	vs. JADE	Best	8	4	3	≈
		Mean	9	6	0	+
30D	vs. **LSHADE**	Best	8	4	3	≈
		Mean	10	3	2	+
	vs. **SHADE**	Best	11	3	1	+
		Mean	12	2	1	+
	vs. **UMOEAs**	Best	4	7	4	≈
		Mean	9	3	3	≈
	vs. JADE	Best	12	2	1	+
		Mean	12	2	1	+
50D	vs. **LSHADE**	Best	8	3	4	+
		Mean	9	4	2	+
	vs. **SHADE**	Best	12	2	1	+
		Mean	11	2	2	+
	vs. **UMOEAs**	Best	6	4	5	≈
		Mean	11	2	2	≈
	vs. JADE	Best	11	2	2	+
		Mean	13	2	0	+
100D	vs. **LSHADE**	Best	8	2	5	≈
		Mean	8	5	2	≈
	vs. **SHADE**	Best	9	2	4	≈
		Mean	12	2	1	+
	vs. **UMOEAs**	Best	7	3	5	≈
		Mean	8	2	5	≈
	vs. JADE	Best	11	2	2	+
		Mean	13	0	2	+

Table VII
THE AVERAGE COMPUTATIONAL TIME FOR10D, 30D,50D, AND 100D

	LSHADEND	LSHADE	SHADE	UMOEAs	JADE
10D	**1.56E+00**	2.75E+00	1.84E+00	5.04E+00	1.64E+00
30D	**1.02E+01**	1.18E+01	1.15E+01	3.14E+01	1.19E+01
50D	**1.89E+01**	2.29E+01	1.94E+01	5.23E+01	2.39E+01
100D	**8.43E+01**	9.19E+01	8.88E+01	1.04E+02	9.73E+01

[12] D. B. Fogel, "Evolutionary algorithms in theory and practice," 1997.

[13] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced engineering informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[14] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 1041–1048.

[15] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 5, pp. 945–958, 2009.

[16] L.-Z. Liao, H. Qi, and L. Qi, "Neurodynamical optimization," *Journal of Global Optimization*, vol. 28, pp. 175–195, 2004.

[17] D. Tank and J. J. Hopfield, "Simple'neural'optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit," *Circuits and Systems, IEEE Transactions on*, vol. 33, no. 5, pp. 533–541, 1986.

[18] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *Circuits and Systems, IEEE Transactions on*, vol. 35, no. 5, pp. 554–562, 1988.

[19] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J. L. Huertas, and E. Sanchez-Sinencio, "Nonlinear switched capacitorneural'networks for optimization problems," *Circuits and Systems, IEEE Transactions on*, vol. 37, pp. 384–398, 1990.

[20] C.-Y. Maa and M. Schanblatt, "A two-phase optimization neural network," *Neural Networks, IEEE Transactions on*, vol. 3, pp. 1003–1009, 1992.

[21] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *Neural Networks, IEEE Transactions on*, vol. 7, no. 6, pp. 1544–1548, 1996.

[22] W. E. Lillo, S. Hui, and S. H. ?ak, "Neural networks for constrained optimization problems," *International journal of circuit theory and applications*, vol. 21, pp. 385–399, 1993.

[23] X.-B. Liang and J. Wang, "A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints," *Neural Networks, IEEE Transactions on*, vol. 11, pp. 1251–1262, 2000.

[24] Y. Leung, K.-Z. Chen, and X.-B. Gao, "A high-performance feedback neural network for solving convex nonlinear programming problems," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 1469–1477, 2003.

[25] Y. Xia and J. Wang, "A recurrent neural network for solving nonlinear convex programs subject to linear constraints," *Neural Networks, IEEE Transactions on*, vol. 16, no. 2, pp. 379–386, 2005.

[26] Z. Yan, J. Wang, and G. Li, "A collective neurodynamic optimization approach to bound-constrained nonconvex optimization," *Neural Networks*, vol. 55, pp. 20–29, 2014.

[27] J. Wang, Y. Zhou, and J. Yin, "Combining tabu hopfield network and estimation of distribution for unconstrained binary quadratic programming problem," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14 870–14 881, 2011.

[28] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*. Siam, 1996, vol. 14.

[29] S. M. Elsayed, R. A. Sarker, D. L. Essam, and N. M. Hamza, "Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1650–1657.

[30] B. Rosner, R. J. Glynn, and M.-L. T. Lee, "The wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics*, vol. 62, no. 1, pp. 185–192, 2006.