IMODEII: an Improved IMODE algorithm based on the Reinforcement Learning

Karam M. Sallam *, Mohamed Abdel-Basset†, Mohammed El-Abd ‡,Ali Wagdy§, ¶
*School of IT and Systems, University of Canberra, ACT 2601, Australia,
†The faculty of Computers and Information, Zagazig University, Egypt,
‡College of Engineering and Applied Sciences, American University of Kuwait, Kuwait,
§Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt,
¶Department of Mathematics and Actuarial Science School of Sciences Engineering, The American University in Cairo, Cairo 11835, Egypt
*karam.sallam@canberra.edu.au, † mohamedbasset@ieee.org, ‡melabd@auk.edu.kw, §aliwagdy@gmail.com

***Abstract*—The success of differential evolution algorithm depends on its offspring breeding strategy and the associated control parameters. Improved Multi-Operator Differential Evolution (IMODE) proved its efficiency and ranked first in the CEC2020 competition. In this paper, an improved IMODE, called IMODEII, is introduced. In IMODEII, Reinforcement Learning (RL), a computational methodology that simulates interaction-based learning, is used as an adaptive operator selection approach. RL is used to select the best-performing action among three of them in the optimization process to evolve a set of solution based on the population state and reward value. Different from IMODE, only two mutation strategies have been used in IMODEII. We tested the performance of the proposed IMODEII by considering 12 benchmark functions with 10 and 20 variables taken from CEC2022 competition on single objective bound constrained numerical optimisation. A comparison between the proposed IMODEII and the state-of-the-art algorithms is conducted, with the results demonstrating the efficiency of the proposed IMODEII.**

***Index Terms*—reinforcement learning, differential evolution, evolutionary algorithms, unconstrained optimisation.**

I. Introduction

In many practical decision-making procedures, optimisation is a critical component of the process. Optimization has caught the attention of many scholars and practitioners for many decades because of its potential to solve planning, scientific, and technical design challenges that emerge in industry, the public sector, and the private sector. Optimisation problems identify the optimal option from a set of candidate solutions that maximises or minimises the intended results [1], [2]. These optimisation problems may be categorised in a variety of ways depending on the number and type of the involved variables, the type and number of objective functions to optimise, the presence of constraints, and a variety of other characteristics [3]. The paper's primary objective is to tackle bound-constrained optimisation problems that have a variety of mathematical features that traditional optimisation techniques cannot solve while Swarm Intelligence (SI) and Evolutionary Algorithms (EAs) techniques can.

Differential Evolution (DE) [4], a simple but effective EA, has been employed to tackle several optimisation problems. DE, like all other EAs, uses three basic operators to steer a group of solutions to acceptable results. Many scholars and practitioners are interested in DE because of its simplicity and resilience [2], [5]. DE and its variants have showed excellent performance on solving bound-constrained and real-application optimisation problems [6]–[10]. According to the no free lunch theorem [11], no single operator (or parameter) or EA is deemed to be the best for all optimisation test functions. To take advantage of the many parameters, operators, and algorithms available, practitioners and researchers have proposed algorithmic frameworks, that use more than one operator/algorithm to evolve a population of individuals towards the optimal solution. In the same direction, Sallam et al. [10] proposed an Improved Multi-Operator DE (IMODE) that won the CEC2020 competition on bound constrained optimisation competition problems. In spite of their benefits, these frameworks do not guarantee consistent results for a broad range of situations [9], [12], which indicates that new designs are required.

Using machine and deep learning, i.e., Reinforcement Learning (RL) [13], can help build efficient optimisation methods. RL is a prominent machine-learning method that performs appropriate action to maximise reward [14]. It works well in many fields, including computer science and engineering. Recent research using RL with DE [15]–[17] has certain limitations. They employed offline Neural Network training to learn the states of the algorithm and the reward values. The trained Neural Network was then employed to anticipate the next action. As a result, the algorithm's capacity to obtain high-quality solutions may have been hampered. Also,the algorithm may not have been able to handle new problems due to its offline training. An AMA algorithm, which an algorithm that combines global search (DE) with local refining (RL), has been proposed in [18]. A DE-RLFR algorithm is developed by Li et al. [19] to solve multiobjective optimisation problems. In DE-RLFRin, the population's evolution direction was adjusted dynamically while each solution in the population used a Q-learning agent.

The major goal of the current research is to propose a better optimisation method that uses (a) RL to pick the most-

performing DE search operator Neural Network during the search process; (b) the effectiveness of using more than one crossover operator and (c) the use of more than one boundary constrained handling technique. When combining RL and DE, the following questions must be addressed; (1) How to form alternative states of continuous problems? (2) How to define the viable actions? and (3) How to compute each action's reward function? For (1), the quality of solutions is considered, which are then segmented. The reward function is the average improvement rate in the objective function value achieved by each DE mutation strategy.

In this paper, RL is combined with IMODE, the winner of the CEC2020 competition on single-objective bound-constrained optimisation problems. RL is used to select one of the actions (DE mutation operator) during optimisation process.

The rest of the paper is divided as follows: Section II presents the literature review of the study. The details of the proposed IMODEII is presented in section III. Section IV presents the obtained results for 10D and 20D and also a comparison with the rival algorithms. Finally, the conclusion amd future work are presented in Section V.

II. Literature Review

In this section, the related literature review are presented.

According to prior research, the DE algorithm's performance is highly dependent on its parameters (population size (NP), crossover rate (Cr), and scaling factor (F). Despite the literature's abundance of DE variants, investigations show that none of them work well for all problems. A recent surge in interest in multi-operator DE variants has been attributed to the difficulties of basic DE variants (i.e., one mutation-based algorithm) for tackling all types of optimisation problems [3], [20]. The multi-operator/multi-method-based algorithms use a selection mechanism to select the most important operator/algorithm during the evolutionary process to evolve a set of solutions in order to reach optimal or best solutions.

Fan et al. [21] presented an Auto-Selection Mechanism (ASM) to pick an appropriate DE variant during the evolutionary process. An Adaptive Operator Selection is introduced by Sallam et al. [20] that combines DE mutation operator performance records and function landscape information to select the best DE mutation operator automatically from a pool of strategies. This algorithm solved 45 unconstrained optimisation problems from the CEC2015 and CEC2014 competitions, proving its superiority. Elsayed et al. [1] used a fuzzy rule-based heuristic to emphasise the best-performing algorithm during the evolutionary process. Their framework worked well for a variety of bound-constrained benchmark problems.

Tasgetiren et al. [22] developed a method to select the optimal parameter settings that improves the performance of DE algorithm. Elsayed et al. [5] proposed an ensemble in a DE algorithm with 16 alternative mutation and crossover operator combinations and a constraint-handling method. A multi-population-based approach for extracting possible DE variants from ensemble features has recently been proposed in [23]. The performance of their approach was demonstrated by solving a set of standard optimisation problems.

Zhang et al. [24] developed a Multi-Layer Competitive-Cooperative (MLCC) architecture to improve DE performance. An improved DE algorithm that use three mutation strategies and a self-adaptive mechanism to set the control parameters values in order to improve the DE exploration and exploitation properties [25]. In this algorithm, one mutation technique was only used to generate new individuals, which automatically picked during the optimisation process. Yu et al. [26] proposed a new algorithm to handle both feasible and infeasible solutions at the same time, thus accelerating convergence while maintaining diversity. The work employed greedy constraint handling to handle the infeasible solutions and to guide them to the feasible ones. The work in [27] introduced a triangle mutation technique to balance the DE's exploration and exploitation capabilities. It outperformed other algorithms on regular benchmarks.

Elsayed et al. [28] presented a United Multi-Operator Evolutionary Algorithms (UMOEAs). In UMOEAs an initial population was divided into equal-sized groups, which were then evolved utilising distinct MOEAs. UMOEA adaptively varied population sizes and selected the superior multi-operator EA based on success rate. To enhance the under-performing ones, an information sharing and exchange method was used to share the best solutions among them. Due to their success in obtaining competitive results, the same authors [29] developed UMOEAsII, which an improved version of UMOEA. UMOEAII framework used many search operators in each algorithm, rather than just one. For further details on multi-operator algorithms, please see Das et al. [30] and Wu et al. [31], as well as Wu et al. [32].

Studies have coupled Q-learning with EAs to improve the performance of RL in real-application problems [33]. Das et al. [34] devised an enhanced Particle Swarm Optimisation (PSO) approach using Q-learning to reduce the computational complexity of the standard RL technique. Similarly, Zhou [35] devised a fuzzy RL-based Genetic Algorithm (GA) that has the ability to escape the local minimum problem in an actor-critic framework. To solve the Travelling Salesman Problem (TSP), Liu and Zeng [36] used reinforcement mutation. Their GA's convergence rate was faster than many known algorithms by using an RL mutation method. Similarly, Alipour et al. [37] proposed a framework that uses GAs and multi-agent RL to solve Traveling Salesman Problems (TSPs). An application of reinforcement neural fuzzy surrogate-assisted multi-objective evolutionary optimisation was recently introduced by Juang and Bui [38], which was used to optimise an ant colony optimisation algorithm.

The field of Deep and Reinforcement learning is rich with several techniques that can be used to select the most-appropriate operator/algorithm during the optimisation process. In this direction, Tan et al. [39] proposed a DE algorithm based on deep Q-network (DEDQN) in which the deep Q-network (DQN) is used to pick the mutation strategy from a mixed pool of DE strategies. The proposed DEDQN applied in two steps, training and prediction. Primarily, the DQN is trained offline by gathering data about the fitness landscape and the benefit (risk) of applying various mutation strategies to the training functions. Second, the trained DQN predicts

the mutation strategy for each generation based on the test function's fitness landscape.

III. Proposed Algorithm (IMODEII)

This section describes the proposed algorithm (IMODEII). It is shown in Algorithm 1 where FES represents the number of fitness evaluations and MAX_{FES} the maximum number of fitness evaluations, NS denotes the number of states and K represents the number of and actions. γ and α are the discount and learning rates.

IMODEII starts with setting the parameters of the algorithm, then randomly generating an initial population of size NP and evaluating it with Q-table elements set to zeros. in IMODEII, three DE mutation oprators (DE weighted-rand-toϕbest and DE/current-to-ϕbest with and without archive) were chosen for their great performance in solving optimization problems. These mutation strategies are utilised to construct three actions (See Subsection). During the optimization process, the RL then selects the optimal action from the action's list to guide the set of solutions toward the optimum. To enhance the obtained solution, an SQP local search is applied in the last few iterations (See Subsections). IMODEII main steps are carried out until the maximum number of fitness evaluations is reached.

The following subsections explain the proposed IMODEII in detail.

Algorithm 1 IMODEII algorithm

```
Define NP, number of states NS, Number of actions K,
Prob_ls ← 0.1, MAX_FES, NP, t ← 1 and FES ← 0;
Generate an initial population (X⃗) randomly with size of
NP solutions;
Evaluate the objective function (f(X⃗)), and update number
of fitness evaluations FES ← FES + NP;
Chose an initial random state s_t;
while FES ≤ MAX_FES do
    For state (s^t), select the action that has the maximum
    Q-value (best action) a^t.
    Use a^t to update the population;
    Apply the Crossover operation as in Section III-C;
    Evaluate the updated population;
    Use Equation 9 to compute the reward (RW^{t+1});
    Update the number of fitness evaluations FES =
    FES + NP;
    Update the number of solutions using Equation 2;
    Update the elements of the Q-table by Equation 11;
    Calculate the population state (s^t = s^{t+1};
    t ← t + 1;
    if FES ≥ 0.85 × MAX_FES then
        Conduct the SQP as described in subsection III-D;
        Update FES
    end if
end whileReturn the best solution.
```

The components of the proposed IMODEII algorithm is discussed in detail in the following subsections.

A. Generation of initial Population and updating method

IMODEII starts with generation a random initial population of size NP as:

$$\overrightarrow{X}_i = \overrightarrow{X}_{min} + (\overrightarrow{X}_{max} - \overrightarrow{X}_{min}) \times rand_i \quad i \in NP \tag{1}$$

where $\overrightarrow{X}_{min}$ is a vector representing lower bounds, $\overrightarrow{X}_{min}$ is a vector denotes the upper bounds and $rand$ is vector or random number between 0 and 1.

The proposed algorithm uses a linear reduction mechanism to dynamically reduce the population through the optimization process. This technique is used in order to preserve the population diversity at the start of the search process and to boost the convergence at the later ierations [40]. This process is described by the following equation:

$$NP^{t+1} = round[(\frac{NP_{min} - NP_{init}}{MAX_{FES}}) \times FES + NP_{init}] \tag{2}$$

where t represents the iteration number, NP_{min} and NP_{init} are the smallest number of solutions required for IMODEII to run and the number of solutions in the initial population, respectively, (FES) the current and maximum numbers of fitness evaluations, respectively.

B. RL main components

In this paper, RL is used to select the best-performing DE mutation strategy (actions), that maximises a reward function in a given state or situation. To do this, we need to determine (1) how the states are represented, (2) what are the possible actions to use; (3) how the reward is calculated.

1) State Representation: In order to represent the population state, in this paper, two metrics (the diversity of the population and the improvement in the objective function values) have been used.

The diversity (div_t), represents the difference between each solution and the average of the decision variables of the population at generation t, computed by Equation 3.

$$div_t = \sqrt{\frac{1}{NP} \times \sum_{i=1}^{NP} (x_{i,j} - \bar{x}_{i,j})^2} \tag{3}$$

where t represents the iteration number, NP the size of the population, $\bar{x}_{i,j}$ the average of decision variables at iteration t for the j^{th} dimension computed as

$$\bar{x}_{i,j} = \frac{\sum_{i=1}^{NP} x_{i,j}^t}{NP} \tag{4}$$

The improvement in the objective function (IOF) value is computed by Equation 5

$$IOF = \frac{|f^* - f_{t,a}^{best}|}{f_{t-1,a}^{best}}, \forall a = 1, 2, ..., AC \tag{5}$$

where f^*, $f_{t,a}^{best}$ and $f_{t-1,a}^{best}$ represent the optimum or best known solution, the best objective function value achieved by

a at iteration t and $t-1$, respectively. AC represent the total number of actions.

Given the above two criteria, the state (s) of the algorithm population is computed by

$$s = \left[\frac{div_t}{div_0}, \frac{IOF_t}{IOF_0}\right] \tag{6}$$

where div_0 and IOF_0 are the diversity and improvement of the objective function of the initial population.

Because the nature of the state variables are continuous, it is essential to divide the space to establish a number of discrete states. It is worth noting that the intervals (div_t/div_0) and (IOF_t/IOF_0) are both [0, 1] and may be subdivided into n_d and n_e sub-intervals. Thus, the population's states cand be categorized to $n_d \times n_e$ intervals. This means that although more n_d and n_i may result in a better control scheme, it also increases the number of created states. Thus, the model's performance and computational complexity must be balanced by choosing an appropriate number of states.

2) Set of Actions: The proposed IMODEII algorithm uses two mutation operators, from which three actions have been used in the Reinforcement Learning. IMODEII evolves the whole population using the following two DE mutation operators and the third action which is a combination of the two since they are effective at addressing unconstrained optimisation issues [10], [20].

- DE/current-to-ϕbest/1 - archive

$$\overrightarrow{V}_i = \overrightarrow{X}_i + F_i \times (\overrightarrow{X}_\phi - \overrightarrow{X}_i + \overrightarrow{X}_{r_1} - \overrightarrow{X}_{r_2}) \tag{7}$$

- DE/current-to-ϕbest/1 - without archive

$$\overrightarrow{V}_i = \overrightarrow{X}_i + F_i \times (\overrightarrow{X}_\phi - \overrightarrow{X}_i + \overrightarrow{X}_{r_1} - \overrightarrow{X}_{r_3}) \tag{8}$$

where $r_1 \neq r_2 \neq r_3 \neq i$ are random integer numbers, $\overrightarrow{X}_{r1}$ and $\overrightarrow{X}_{r3}$ picked objective from the full population, $\overrightarrow{X}_\phi$ from the best ϕ% of individuals in the entire population, and $\overrightarrow{X}_{r2}$ from the entire population and archive combined, respectively. The Proposed IMODEII algorithm uses an archive to sustain population diversity, adding new solutions that are worse than their children solution [41]. If the archive exceeds its predetermined size, the worst solutions are deleted to create room for new solutions. The set

3) Reward function: In this paper, the reward $RW(s,a)$ is computed based on the improvement in the number of solutions in the current generation (t). Equation 9 is used to calculate the reward given to each a action in a given s state.

$$RW(s,a) = \frac{\sum_{i=1}^{NP}((NI_{i,a}=1) - (NI_{i,a}=0))}{NP} \tag{9}$$

where NI_a represents the number of individuals updates when using action a, is calculated by

$$NI_{i,a} = \begin{cases} 0 & fit(\overrightarrow{X}_i, a) - fit(\overrightarrow{X}_i) \geq 0 \\ 1 & fit(\overrightarrow{X}_i, a) - fit(\overrightarrow{X}_i) < 0 \end{cases} \tag{10}$$

where $fit(\overrightarrow{X}_i, a)$ denotes the objective function value of the i^{th} individual by action a.

4) Q-learning Algorithm: Q-learning is a model-free RL algorithm. Its basic functioning premise is the environment's reward or penalty depending on a state change. We used Q-learning to express RL in a state-action table (Q-table) as shown in Equation III-B4. This matrix/table is an $NS \times K$ matrix utilised as a future reference in case of facing the same situation [15]. Here's how it's laid out

$$\mathbf{Q} = \begin{array}{c} \begin{matrix} \mathrm{a}_1 & \mathrm{a}_2 & \dots & \mathrm{a}_K \end{matrix} \\ \begin{bmatrix} Q(s_1,a_1) & Q(s_1,a_1) & \dots & Q(s_1,a_K) \\ Q(s_2,a_1) & Q(s_2,a_2) & \dots & Q(s_2,a_K) \\ \vdots & \vdots & \ddots & \vdots \\ Q(s_{NS},a_1) & Q(s_{NS},a_2) & \dots & Q(s_{NS},a_K) \end{bmatrix} \end{array} \begin{matrix} \mathrm{s}_1 \\ \mathrm{s}_2 \\ \vdots \\ \mathrm{s}_{NS} \end{matrix}$$

The Q-table values are calculated using the following equation.

$$\begin{aligned} Q^{t+1}(s^t,a^t) = Q^t(s^t,a^t) + \\ \alpha \times \left[RW^{t+1} + \gamma \times \max_a(Q(s^{t+1},a^t) - Q(s^t,a^t))\right] \end{aligned} \tag{11}$$

where α and γ are the learning rate and discount factor and their values lie between 0 and 1.

C. Crossover operators

In IMODEII algorithm, after mutation (action) operation, a random selection between exponential and binomial crossover is performed, by the following Algorithm 2.

Algorithm 2 Crossover

Randomly generate r between 0 and 1;
if $r \leq 0.4$ **then**
  Apply binomial Crossover
  $\mathrm{u}_{i,j} = \begin{cases} v_{i,j} & \text{if } (rand \leq Cr_i \text{ or } j = j_{rand}) \\ x_{i,j} & \text{otherwise} \end{cases}$
else
  Apply exponential Crossover
  $\mathrm{u}_{i,j} = \begin{cases} v_{i,j} & \text{for } j = \langle l \rangle_D, \langle l+1 \rangle_D, ..., \langle l+L-1 \rangle_D \\ x_{i,j} & \text{for all other } j \in [1, D] \end{cases}$
end if

It is worth to mention that, the proposed IMODEII is flexible to use more mutation and crossover operators.

D. Local Search

To aid in the proposed IMODEII's convergence, sequential quadratic programming (SQP) is used to find the optimum individual in every iteration during the last 15% of the optimisation process. The probability of applying the local search ($Prob_{ls}$) was set to 0.1 and SQP runs for up to CFE_{ls} fitness evaluations. As mentioned in Steps 5-9 of Algorithm 3, the likelihood of using this local search is constantly updated.

Algorithm 3 SQP

Input: the best individual from the entire population $\overrightarrow{X}^{best}$;
Randomly generate r between 0 and 1;
if $r \leq Prob_{ls}$ **then**
  Perform the sequential quadratic search (SQP) to $\overrightarrow{X}^{best}$ for CFE^{ls} objective functions evaluations;
  if $f(\overrightarrow{X}^{sqp}) < f(\overrightarrow{X}^{best})$ **then**
    Update $Prob_{ls} \leftarrow 0.1$;
    Replace $\overrightarrow{X}^{best}$ by $\overrightarrow{X}^{sqp}$ ($\overrightarrow{x}^{best} \leftarrow \overrightarrow{X}^{sqp}$) and replace $f(\overrightarrow{X}^{best})$ by $f(\overrightarrow{X}^{sqp})$ ($f(\overrightarrow{X}^{best}) \leftarrow f(\overrightarrow{X}^{sqp})$);
    else
    Decrease the value of $Prob_{ls}$ ($Prob_{ls} \leftarrow 0.0001$);
  end if
  Update the number of objective function evaluations ($FES \leftarrow FES + CFE_{ls}$);
end if

IV. Experimental Results

The proposed IMODEII method was tested on 12 optimisation problems with 10, and 20 variables with a search space of $[-100, 100]^D$ taken from CEC2022 competition on bound-constrained single objective optimisation problems. The results obtained from IMODEII were compared to a number of the state-of-the-art algorithms.

IMODEII was coded in MATLAB R2018b. We ran the proposed algorithm on a laptop with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz16, 32 GB RAM, and Windows 10. All other comparing algorithms utilised the same seed to guarantee fair comparisons. The parameters' values of rival algorithms were obtained from their original papers. Following the competition guideline, each algorithm was run 30 times for 200,000 FES and 1,000,000 FES for functions with 10D and 20D, respectively, or if the distance from the best solution ($|f(\overrightarrow{x}_{best} - f(\overrightarrow{x^*})|$) less than $1E-08$.

We used two non-parametric tests to compare the algorithms (Friedman ranking and Wilcoxon signed-rank tests [42]). The proposed IMODEII algorithm's performance was also graphically compared with the rival algorithms by plotting performance profiles [43]. The performance profile (ρ_A) of an algorithm (A) is determined as

$$\rho_A(\tau) = \frac{1}{n_p} \times |p \in P : r_{p,A} \leq \tau| \tag{12}$$

where $\rho_A(\tau)$ is the chance that the performance rate $r_{p,A}$ is within a value $\tau \in R$ for the best possible probability and rho_A is a function that provides the cumulative distribution of the $r_{p,A}$.

A. Parameter Setting and Analysis

In this paper, the proposed IMODEII algorithm parameters were set based on the some experimental analysis. For RL α was set to a value of 0.25 and γ to a value of 0.85. For the SQP, FES_{LS} was set to $0.85 \times FES_{max}$ and $Prob_{ls}$ to a value of 0.1. For the IMODEII algorithm, the initial population NP_{init} was set to a value of $10 * D$ individuals, the minimum population size NP_{min} to 4, the the archive rate (A) to 1.4 and the memory size (H) to $18 \times D$.The values of the scale parameter (F) and the crossover rate (Cr) were set following the same method used in [10].

TABLE I: Results for 10D problems.

Functions	Best	Worst	Median	Mean	Std.	Avg. FES
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	7.3304E+04
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	1.0800E+05
F03	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	5.9982E+04
F04	6.9647E+00	1.5919E+01	1.1442E+01	1.1243E+01	2.6666E+00	2.0000E+05
F05	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	1.0348E+05
F06	1.3229E-02	4.6697E-01	1.7255E-01	2.0227E-01	1.3345E-01	2.0000E+05
F07	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	1.1870E+05
F08	9.7302E-05	2.9157E+00	3.3668E-02	2.0629E-01	5.4770E-01	2.0000E+05
F09	**0.0000E+00**	2.2928E+02	2.2928E+02	2.2164E+02	4.1861E+01	1.9686E+05
F10	**0.0000E+00**	1.0025E+02	3.1228E-02	1.4989E+01	3.4305E+01	1.7765E+05
F11	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	7.3509E+04
F12	1.5937E+02	1.6349E+02	1.6144E+02	1.6167E+02	1.0002E+00	2.0000E+05

TABLE II: Results for 20D problems.

Functions	Best	Worst	Median	Mean	Std.	Avg. FES
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	3.6652E+05
F02	**0.0000E+00**	4.9085E+01	4.4896E+01	4.0440E+01	1.5862E+01	9.8515E+05
F03	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	2.5356E+05
F04	4.7758E+01	8.5566E+01	7.2544E+01	6.9074E+01	1.0084E+01	1.0000E+06
F05	4.2130E+01	1.3256E+03	3.7400E+02	4.8258E+02	3.8779E+02	1.0000E+06
F06	1.6661E-01	1.0058E+01	2.2187E+00	3.4516E+00	3.0748E+00	1.0000E+06
F07	8.0662E-03	1.3433E+01	2.3120E+00	2.9704E+00	2.6592E+00	1.0000E+06
F08	8.7051E-01	2.0801E+01	2.0256E+01	1.8101E+01	5.6632E+00	1.0000E+06
F09	1.8078E+02	1.8078E+02	1.8078E+02	1.8078E+02	8.6723E-14	1.0000E+06
F10	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	4.0985E+05
F11	**0.0000E+00**	3.0000E+02	3.0000E+02	2.8000E+02	7.6112E+01	9.6380E+05
F12	2.3374E+02	2.4250E+02	2.3739E+02	2.3757E+02	2.0748E+00	1.0000E+06

B. Detailed results obtained from IMODEII

This section presents comprehensive results obtained from the proposed IMODEII and other competing algorithms.

1) 10D results: Table I shows the best, worst, median, average and standard deviation of the fitness error values ($|f(\overrightarrow{x}_{best}) - f(\overrightarrow{x^*})$) obtained from the proposed IMODEII and the last column in this table represent the average fitness evaluations to reach optimal solution.

For the unimodal function (F01), the proposed IMODEII obtained the optimal result for both best and the mean results. For the basic functions (F02-F05), IMODEII obtained the optimal solutions for F02, F03 and F05, while the performance deteriorated for F04. For the hybrid functions (F06-F08), IMODEII achieved the optimum value for F07, and very close results to the optimal for both F06 and F08. Finally, for the composition functions (F09-F12), IMODEII achieved the optimum value for F09 and F011, very close results to the optimal for F10, while its performance is degraded for F12, as it stuck in local solution.

2) 20D results: Table II presents the best, worst, median, average and standard deviation of the fitness error values ($|f(\overrightarrow{x}_{best}) - f(\overrightarrow{x^*})$) obtained from the proposed IMODEII and the last column in this table represent the average fitness evaluations to reach optimal solution. The proposed IMODEII is able to obtain the optimum for F03 and F05 of the basic functions, while it stuck in local solution for F02 and obtained a close result to the optimum for F04. For the hybrid functions (F06 - F08), the proposed IMODEII achieved a very good results. In case of composition functions (F09 - F12), IMODEII obtained the optimum for F11, while it stuck in local solutions for the remaining functions.

TABLE III: IMODEII algorithm run time complexity

	T_0	T_1	$\hat{T_2}$	$(\hat{T_2}-T_1)/T_0$
$D = 10$		0.041321	2.32E+00	24.194
$D = 20$	0.094263	0.079156	3.50E+00	36.290

3) Complexity of the proposed algorithm: The IMODEII complexity is computed for 10D and 20D following the competition guideline, with the results' summary presented in Table III. It can be concluded from Table III that, the complexity of the proposed IMODEII is reasonably small for 10D and linearly increased when the dimension of the problem increase.

C. Comparison with the state-of-the-art algorithms

To evaluate the proposed IMODEII's efficacy, its performance is compared to that of the following state-of-the-art algorithms:

1) SSA: Spherical Search algorithm [44];
2) UMOEA: United Multi-operator EA algorithm [28];
3) IMODE-AGSK: Improved Multi-operator DE with Adaptive GSK algorithm [8];
4) MadDE [45]:
5) IMODE: Improved Multi-operator DE Algorithm [10].

As indicated before, these algorithms were run using the settings specified by their authors in their papers, and all other circumstances are identical to those specified in the competition rules.

1) 10D results: According to Table IV and in terms of best obtaine results, the proposed IMODEII outperformed SSA, UMOEA, IMODE-AGSK, MadDE, and IMODE in 4, 5, 3, 3 and 3 test problems, was equal in 7, 5, 7, 7, and 7 test problems, and inferior in 1, 2, 2, 2 and 2 test functions. In terms of average outcomes, IMODEII outperformed SSA, UMOEA, IMODE-AGSK, MadDE, and IMODE in 4, 9, 3, 3 and 6 test functions, respectively, and was comparable in 5, 1, 6, 6 and 5 test functions, but was worth than them in 3, 2, 3, 3 and 1 test problems.

According to the Wilcoxon test, IMODEII was not significantly better than other competing algorithms for both best and average results obtained, however, there is a bias towrds the proposed IMODEII in terms of the number of better results obtained. Also, the Friedman rank test was carried out to rank all the rival algorithm with the results presented in Table V. It is clear that the proposed IMODEII algorithms is ranked first for both best and the mean obtained results, followed by IMODE-AGSK, while MadDE comes in the third place.

Figure 1 shows the performance profiles graph for the test functions with 10 and 20 dimensions, which compare all the competing algorithms. The Friedman and Wilcoxon tests yielded consistent findings, as IMODEII first achieved a ratio of 1.0 at $\tau = 1.25$ and $\tau = 1.75$ for best and average results for 10D.

2) 20D results: The summary of the results obtained from IMODEII and the rival algorithms is presented in Table VI. According to Table IV and in terms of best obtaine results, the proposed IMODEII outperformed SSA, UMOEA, IMODE-AGSK, MadDE, and IMODE in 5, 5, 5, 4 and 7 test problems, was equal in 5, 5, 4, 5 and 2 test problems, and inferior in 2, 2, 3, 3, and 3 test functions, respectively. In terms of average outcomes, IMODEII outperformed SSA, UMOEA, IMODE-AGSK, MadDE, and IMODE in 4, 6, 7, 4 and 8 test functions, respectively, and was comparable in 5, 2, 3, 5 and 1 test functions, but was worth than them in 3, 4, 2, 3 and 3 test problems, respectively.

TABLE IV: Summary of comparisons of performance of IMODEII, UMOEA, IMODE-AGSK, MadDE, and IMODE obtained by the Wilcoxon test

Criteria	Algorithms	Better	Equal	Worth	(P-value, Dec.)
best	IMODEII vs. SSA	4	7	1	(0.225, ≈)
	IMODEII vs. UMOEA	5	5	2	(0.499, ≈)
	IMODEII vs. IMODE-AGSK	3	7	2	(0.893, ≈)
	IMODEII vs. MadDE	3	7	2	(0.893, ≈)
	IMODEII vs. IMODE	3	7	2	(0.893, ≈)
Average	IMODEII vs. SSA	4	5	3	(0.612, ≈)
	IMODEII vs. UMOEA	9	1	2	(0.061, ≈)
	IMODEII vs. IMODE-AGSK	3	6	3	(0.917, ≈)
	IMODEII vs. MadDE	3	6	3	(0.917, ≈)
	IMODEII vs. IMODE	6	5	1	(0.237, ≈)

TABLE V: Average Rankings of the IMODEII, SSA, UMOEA, IMODE-AGSK, MadDE and IMODE algorithms based on the best results for 10D (Friedman)

Algorithm	Ranking for Best	Ranking for Average
IMODEII	2.9583	2.75
SSA	3.9167	3.50
UMOEA	4	4.88
IMODE-AGSK	3.2083	2.96
MadDE	3.2917	3.04
IMODE	3.625	3.88

According to the Wilcoxon test, IMODEII was not significantly better than other competing algorithms for both best and average results obtained, however, there is a bias towrds the proposed IMODEII in terms of the number of better results obtained. Also, the Friedman rank test was carried out to rank all the rival algorithm with the results presented in Table V. It is clear that the proposed IMODEII algorithms is ranked first for both best and average obtained results, followed by IMODE-AGSK in regards to best and SSA in regards to averag, while SSA comes in the third place in terms of best ranking and IMODE-AGSK in terms of average obtained results. This finding was supported by the performance profiles shown in Figure 1 (c) and (d), where the proposed algorithm achieved a value of 1 first $\tau = 2.2$ and $\tau = 175$ for for best and average results for 20D, respectively.

TABLE VI: Comparisons summary of performance of IMODEII, UMOEA, IMODE-AGSK, MadDE, and IMODE obtained by the Wilcoxon test for 20D test problems

Criteria	Algorithms	Better	Equal	Worth	(P-value, Dec.)
best	IMODEII vs. SSA	5	5	2	(0.499, ≈)
	IMODEII vs. UMOEA	5	5	2	(0.063, ≈)
	IMODEII vs. IMODE-AGSK	5	4	3	(0.401, ≈)
	IMODEII vs. MadDE	4	5	3	(0.866, ≈)
	IMODEII vs. IMODE	7	2	3	(0.169, ≈)
Average	IMODEII vs. SSA	4	5	3	(0.866, ≈)
	IMODEII vs. UMOEA	6	2	4	(0.646, ≈)
	IMODEII vs. IMODE-AGSK	7	3	2	(0.260, ≈)
	IMODEII vs. MadDE	4	5	3	(0.612, ≈)
	IMODEII vs. IMODE	8	1	3	(0.110, ≈)

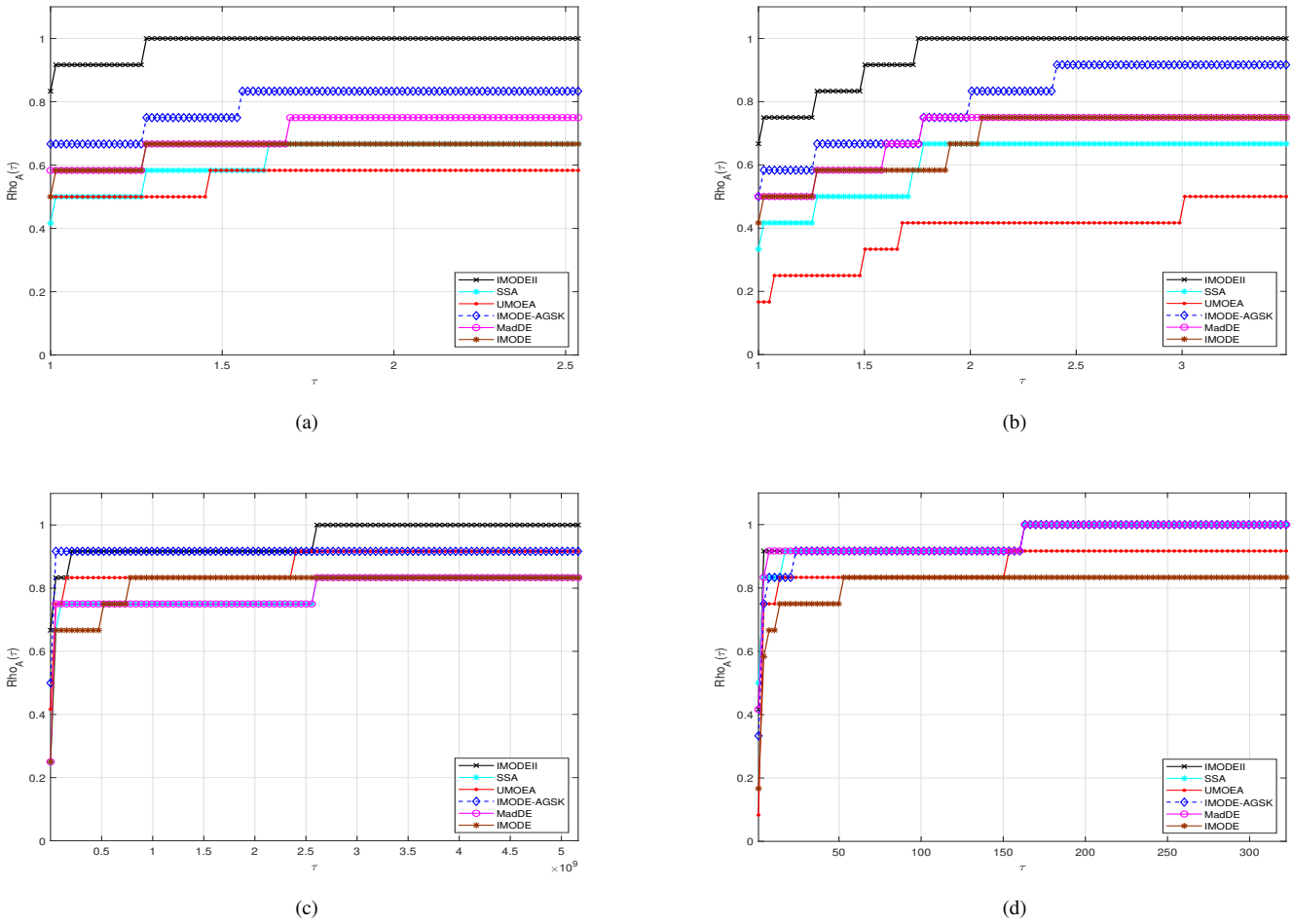

Fig. 1: Performance profiles graphs comparing the performance of IMODEII, SSA, UMOEA,IMODE-AGSK, MadDE and IMODE based on the best and average results for (a) 10D best results; (b) 10D Average results; (c) 20D best Results; and (d) 20D Average results

TABLE VII: Average Rankings of the IMODEII, SSA, UMOEA, IMODE-AGSK, MadDE and IMODE algorithms based on the best results for 10D (Friedman)

Algorithm	Ranking for Best	Ranking for Average
IMODEII	2.79	2.92
SSA	3.38	3.13
UMOEA	3.71	3.54
IMODE-AGSK	3.00	3.50
MadDE	3.71	3.46
IMODE	4.42	4.46

V. Conclusion

Many evolutionary algorithms (EAs) have been created to solve optimisation problems, however no single algorithm/operator can handle all optimisation problems. Due to this, multi-operator and/or multi-method-based algorithms have been developed in the literature. In this paper, we proposed a new multi-operator DE that using more than one DE mutation operators, with SQP local search employed in last optimisation iterations. In the proposed IMODEII, the most appropriate mutation operator was chosen using RL based on solution diversity and quality. The proposed algorithm's performance was evaluated by solving 24 optimisation problems with 10,and 20 variables. The results obtained from the proposed IMODEII ware compared against five state-of-the-art algorithms namely, SSA, UMOEA, IMODE-AGSK, MadDE and IMODE. The results demonstrated the efficiency of the proposed IMODEII algorithm by using Friedman and Wilcoxon tests and the performance profile graph.

In the future, the proposed algorithm will be tested by solving more optimisation problems. Also, it will be extended to be applied in the energy domain, to determine the solar systems unknow parameters. Further, we will adopt the algorithm to solve other optimisation problems, i.e., multi-objective, constrained and/or integer optimisation problems.

References

[1] S. Elsayed, R. Sarker, and C. A. C. Coello, "Fuzzy rule-based design of evolutionary algorithm for optimization," *IEEE transactions on cybernetics*, vol. 49, no. 1, pp. 301–314, 2017.

[2] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-assisted multi-operator differential evolution for solving constrained optimization problems," *Expert Systems with Applications*, p. 113033, 2019.

[3] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Improved united multi-operator algorithm for solving optimization problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[4] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[5] S. Elsayed, R. Sarker, C. C. Coello, and T. Ray, "Adaptation of operators and continuous control parameters in differential evolution for constrained optimization," *Soft Computing*, vol. 22, no. 19, pp. 6595–6616, 2018.

[6] K. M. Sallam, R. A. Sarker, D. L. Essam, and S. M. Elsayed, "Neurodynamic differential evolution algorithm and solving cec2015 competition problems," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 1033–1040.

[7] A. A. Samir, A. R. Rashwan, K. M. Sallam, R. K. Chakrabortty, M. J. Ryan, and A. A. Abohany, "Evolutionary algorithm-based convolutional neural network for predicting heart diseases," *Computers & Industrial Engineering*, vol. 161, p. 107651, 2021.

[8] A. W. Mohamed, A. A. Hadi, P. Agrawal, K. M. Sallam, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm with adaptive parameters hybrid with imode algorithm for solving cec 2021 benchmark problems," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 841–848.

[9] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-assisted multi-operator differential evolution for solving constrained optimization problems," *Expert Systems with Applications*, vol. 162, p. 113033, 2020.

[10] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[11] Y.-C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of optimization theory and applications*, vol. 115, no. 3, pp. 549–570, 2002.

[12] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 372–379.

[13] Y. Hu, Y. Yao, and W. S. Lee, "A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs," *Knowledge-Based Systems*, vol. 204, p. 106244, 2020.

[14] Z. Ding, Y. Huang, H. Yuan, and H. Dong, "Introduction to reinforcement learning," in *Deep reinforcement learning*. Springer, 2020, pp. 47–123.

[15] A. K. Sadhu, A. Konar, T. Bhattacharjee, and S. Das, "Synergism of firefly algorithm and q-learning for robot arm path planning," *Swarm and Evolutionary Computation*, vol. 43, pp. 50–68, 2018.

[16] Z. Cao, C. Lin, M. Zhou, and R. Huang, "Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 825–837, 2018.

[17] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Evolutionary framework with reinforcement learning-based mutation adaptation," *IEEE Access*, vol. 8, pp. 194 045–194 071, 2020.

[18] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar, "Realization of an adaptive memetic algorithm using differential evolution and q-learning: A case study in multirobot path planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 814–831, 2013.

[19] Z. Li, L. Shi, C. Yue, Z. Shang, and B. Qu, "Differential evolution based on reinforcement learning with fitness ranking for solving multimodal multiobjective problems," *Swarm and Evolutionary Computation*, vol. 49, pp. 234–244, 2019.

[20] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-based adaptive operator selection mechanism for differential evolution," *Information Sciences*, vol. 418, pp. 383–404, 2017.

[21] Q. Fan, X. Yan, and Y. Zhang, "Auto-selection mechanism of differential evolution algorithm variants and its application," *European Journal of Operational Research*, vol. 270, no. 2, pp. 636–653, 2018.

[22] M. F. Tasgetiren, P. N. Suganthan, and Q.-K. Pan, "An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem," *Applied Mathematics and Computation*, vol. 215, no. 9, pp. 3356–3368, 2010.

[23] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, and P. N. Suganthan, "Ensemble of differential evolution variants," *Information Sciences*, vol. 423, pp. 172–186, 2018.

[24] S. X. Zhang, L. M. Zheng, K. S. Tang, S. Y. Zheng, and W. S. Chan, "Multi-layer competitive-cooperative framework for performance enhancement of differential evolution," *Information Sciences*, vol. 482, pp. 86–104, 2019.

[25] M. Attia, M. Arafa, E. Sallam, and M. Fahmy, "An enhanced differential evolution algorithm with multi-mutation strategies and self-adapting control parameters," 2019.

[26] X. Yu, X. Yu, Y. Lu, G. G. Yen, and M. Cai, "Differential evolution mutation operators for constrained multi-objective optimization," *Applied Soft Computing*, vol. 67, pp. 452–466, 2018.

[27] A. W. Mohamed, "A novel differential evolution algorithm for solving constrained engineering optimization problems," *Journal of Intelligent Manufacturing*, vol. 29, no. 3, pp. 659–692, 2018.

[28] 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 *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014, pp. 1650–1657.

[29] S. Elsayed, N. Hamza, and R. Sarker, "Testing united multi-operator evolutionary algorithms-ii on single objective optimization problems," in *2016 IEEE congress on evolutionary computation (CEC)*. IEEE, 2016, pp. 2966–2973.

[30] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[31] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Information Sciences*, vol. 329, pp. 329–345, 2016.

[32] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms–a survey," *Swarm and evolutionary computation*, vol. 44, pp. 695–711, 2019.

[33] P. Goyal, H. Malik, and R. Sharma, "Application of evolutionary reinforcement learning (erl) approach in control domain: A review," *Smart innovations in communication and computational sciences*, pp. 273–288, 2019.

[34] P. Das, H. Behera, and B. Panigrahi, "Intelligent-based multi-robot path planning inspired by improved classical q-learning and improved particle swarm optimization with perturbed velocity," *Engineering science and technology, an international journal*, vol. 19, no. 1, pp. 651–669, 2016.

[35] C. Zhou, "Robot learning with ga-based fuzzy reinforcement learning agents," *Information Sciences*, vol. 145, no. 1-2, pp. 45–68, 2002.

[36] F. Liu and G. Zeng, "Study of genetic algorithm with reinforcement learning to solve the tsp," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6995–7001, 2009.

[37] M. M. Alipour, S. N. Razavi, M. R. Feizi Derakhshi, and M. A. Balafar, "A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem," *Neural Computing and Applications*, vol. 30, no. 9, pp. 2935–2951, 2018.

[38] C.-F. Juang and T. B. Bui, "Reinforcement neural fuzzy surrogate-assisted multiobjective evolutionary fuzzy systems with robot learning control application," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 3, pp. 434–446, 2019.

[39] Z. Tan and K. Li, "Differential evolution with mixed mutation strategy based on deep reinforcement learning," *Applied Soft Computing*, vol. 111, p. 107678, 2021.

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

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

[42] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[43] H. J. Barbosa, H. S. Bernardino, and A. M. Barreto, "Using performance profiles for the analysis and design of benchmark experiments," in *Advances in Metaheuristics*. Springer, 2013, pp. 21–36.

[44] A. Kumar, R. K. Misra, D. Singh, S. Mishra, and S. Das, "The spherical search algorithm for bound-constrained global optimization problems," *Applied Soft Computing*, vol. 85, p. 105734, 2019.

[45] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, "Improving differential evolution through bayesian hyperparameter optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 832–840.