Received October 10, 2020, accepted October 21, 2020, date of publication October 26, 2020, date of current version November 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033593

Evolutionary Framework With Reinforcement Learning-Based Mutation Adaptation

KARAM M. SALLAM[1], SABER M. ELSAYED[2], (Member, IEEE), RIPON K. CHAKRABORTTY[3], (Member, IEEE), AND MICHAEL J. RYAN[3], (Senior Member, IEEE)

[1]Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt
[2]School of Engineering and IT, UNSW Canberra at ADFA, Canberra, ACT 2612, Australia
[3]Capability Systems Center, School of Engineering and IT, UNSW Canberra at ADFA, Canberra, ACT 2612, Australia

Corresponding author: Karam M. Sallam (karam_sallam@zu.edu.eg)

ABSTRACT Although several multi-operator and multi-method approaches for solving optimization problems have been proposed, their performances are not consistent for a wide range of optimization problems. Also, the task of ensuring the appropriate selection of algorithms and operators may be inefficient since their designs are undertaken mainly through trial and error. This research proposes an improved optimization framework that uses the benefits of multiple algorithms, namely, a multi-operator differential evolution algorithm and a co-variance matrix adaptation evolution strategy. In the former, reinforcement learning is used to automatically choose the best differential evolution operator. To judge the performance of the proposed framework, three benchmark sets of bound-constrained optimization problems (73 problems) with 10, 30 and 50 dimensions are solved. Further, the proposed algorithm has been tested by solving optimization problems with 100 dimensions taken from CEC2014 and CEC2017 benchmark problems. A real-world application data set has also been solved. Several experiments are designed to analyze the effects of different components of the proposed framework, with the best variant compared with a number of state-of-the-art algorithms. The experimental results show that the proposed algorithm is able to outperform all the others considered.

INDEX TERMS Adaptive method, bound constrained optimization, differential evolution, evolutionary algorithms, reinforcement learning.

I. INTRODUCTION

Many real-world decision-making processes in computer science, engineering and other domains are concerned with finding the best solution among a number of candidate ones that optimizes (maximizes or minimizes) the desired outcome [1]. These optimization problems can be classified in many ways based on the numbers and types of their decision variables, types and numbers of their fitness functions that needs to be optimized, the existence of constraints, and many other factors [2]. The main focus in this paper is to solve bound-constrained optimization problems which incorporate different mathematical properties that conventional optimization solvers cannot handle but evolutionary algorithms (EAs) and swarm intelligence (SI) approaches are capable of solving.

Of the EAs, differential evolution (DE) [3], which is a simple yet effective algorithm, has been used to solve several types of optimization problems. As with all other EAs, DE uses three main operators (mutation, crossover and selection) to guide a set of solutions to obtain acceptable results. DE has attracted the attention of many researchers and practitioners because of its ease of use, its simple structure, and its robustness [4], [5]. Covariance matrix adaptation evolution strategy (CMA-ES) [6] has also attracted the attention of researchers [4], [7]. Both of these algorithms have shown very good performance in solving the bound-constrained and real-application optimization problems [1], [2], [8]. However, there is no single operator (or parameter) and EA that is considered to be the best for all kinds of optimization test functions [2], [4], [9], this is in line with the no free lunch

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Gaggero.

theory [10]. This has motivated and encouraged practitioners and researchers to propose algorithmic frameworks that utilize the benefits and strengths of various parameters, operators and algorithms. Even though these frameworks commonly boost the performance of the evolutionary task, they still do not guarantee consistent outcomes for a wide variety of problems [1] reflecting the need for improved designs.

One possible way of designing efficient optimization algorithms is to use the power of machine learning, i.e., reinforcement learning (RL) [11], [12]. RL is a very popular machine-learning mechanism which takes a suitable action that maximizes the reward in a particular situation [13]. It has shown its effectiveness in many domains including, but not limited to, engineering and computer science. Although some recent studies adopted RL with DE [14]–[17], there are some limitations in their approaches. They used offline training (neural network (NN) training) to build a knowledge base about the DE's states and reward values. This NN was then used to predict the future action to be implemented. As a consequence, the considerable amount of computational resources required may have affected the algorithm's capability to converge to high-quality solutions. Also, due to its offline training, the algorithm may not have been capable of dealing with new problems not covered in the training phase. Cao *et al.* [16] used RL with a cuckoo search algorithm to solve scheduling problems but reported that their algorithm was computationally expensive. In addition, the states they considered to describe the current situations of the evolutionary process did not represent the algorithm's capability to improve solutions over generations. Furthermore, measuring diversity in the objective rather than decision space might have been inappropriate.

Motivated by the above-mentioned issues, the main aim of this paper is to propose an improved optimization algorithm by using the benefits of (1) RL to automatically emphasize the best-performing DE mutation operator and (2) the search capability of multiple algorithms. When using RL with DE, a few important points to consider are how to: (a) represent the possible states of continuous problems; (b) determine a list of possible actions; and (c) calculate the reward function of each action. For (a), we consider both the diversity and quality of solutions which are then partitioned to form some discrete segments. The possible actions are choosing the DE operators to use, with the average improvement rate in the fitness value obtained by each operator considered a reward function. In line with this, Reinforcement learning-based mutation adaptation (MARL with CMA (MARLwCMA)), which starts with a randomly generated population with a default Q-table initialized, is introduced. Then, at each generation, based on the maximum Q-value, the best action is chosen and used to evolve the whole population with the quality of solutions and population diversity recorded, the immediate reward calculated, and the Q-table updated. This process continues until the maximum number of fitness evaluations (FEs) is reached. We believe that this way of adapting ML in the proposed framework is the core novelty of this work, in which the framework is able to do learning during the optimization process as well as the representation of states and actions on the context of automatic-configuration of algorithms for real-valued optimization problems.

To use the strengths of multiple algorithms, MARL and CMA-ES guide two different sub-populations to obtain better solutions for a number of epochs (CS generations). Then, based on their effectiveness, one algorithm is chosen to evolve its sub-population for the next number of epochs (CS epochs = cycle). After two cycles ($2CS$), an information-sharing procedure is used and then both algorithms work in parallel for a certain number of epochs. These steps are repeatedly executed until a stopping criterion is met. A local search is also used to boost the exploitation capability of the proposed algorithm.

This framework was tested on 73 benchmark optimization problems with 10, 30 and 50 dimensions, 28 from the CEC2013 [18], 30 from the CEC2014 [19] and 15 from the CEC2015 [20]) competitions for single-objective optimization. The results demonstrate that, in terms of the quality of solutions and statistical comparisons, this algorithm can outperform 16 others. The proposed algorithm components are also evaluated to provide further insight into the proposed framework.

The main contributions of this paper are:

- A multi-method evolutionary algorithm is proposed which utilizes the benefits of multi-operator DE algorithm and the CMA-ES.
- A reinforcement learning is used to choose the best performing algorithm (MODE or CMA-ES) during the optimization process.
- The performance of MARLwCMA has been assessed by solving 73 problems with 10D, 30D, 50D collected from CEC2013, CEC2014 and CEC2015. Its performance has been also tested for solving test problems with higher dimensions (100D) from CEC2014 and CEC2017.
- The performance of the proposed algorithm is further tested by solving real-world application problems taken from CEC2011 benchmark problems.

The rest of this paper is organized as follows: a brief literature review on related research studies and RL is presented in Section II; the proposed approach and details of its components are provided in Section III; the experimental results and analyses are in Section IV; and, finally, conclusions and future work are in V.

II. LITERATURE REVIEW

This section introduces literature review and related work for single operator DE-based algorithms, multi-operator DE-based algorithms and the use of RL with an EA.

A. DE AND ITS VARIANTS

Storn and Price [3] proposed a DE which is one of the most powerful EA variants, due to its simplicity, fast convergence, and the same values of the parameter can be used for dealing with a range of problems. It has been also widely used and successfully utilized to solve real-world application problems in many engineering and scientific fields [7], [9]. Initially, the algorithm begins with a population of individuals and then a mutant solution is created for each individual (target solution) in the entire population, by summing the weighted difference vector between two individuals to another individual. A new offspring is produced by joining the mutant and target solutions and, finally, a selection operator is carried out to decide which of the parent and offspring vectors will enter the next population.

In the literature, many variants of the DE algorithm have been proposed. In this section we will review the most recent DE variants. Liu and Lampinen [21] proposed fuzzy adaptive DE algorithm (FADE) to solve optimization problems. It used fuzzy logic controllers to manage the values of F and Cr parameters. Zhang *et al.* [22] proposed an adaptive DE algorithm that used an external archive (JADE). JADE automatically adapts Cr and F based on previously successful experience. It also uses the "DE/current-to-pbest" mutation strategy to generate new solutions. Recently, Tanabe and Fukunaga [23] introduced an improved version of JADE, called SHADE, in which F and Cr values are tuned by a history-based parameter adaptation method. Later the same authors improved the SHADE algorithm by using a linear population size reduction technique (LPSR), that linearly reduces the population during the optimization process [24]. Their new proposed algorithm is called LSHADE, which won the CEC2014 competition.

Since the proposal of LSHADE algorithm, many improvements have been proposed to the original LSHADE. Brest *et al.* [25] proposed an improved version of LSHADE, called iLSHADE, with the modifications mainly to the memory update technique. This algorithm was further improved by the same author and they introduced jSO [26] that used a new weighted version of mutation operator. To maintain an effective balance between exploitation and exploration, Awad *et al.* [27] proposed a new way to adapt LSHADE control parameters (i.e., F and Cr). This is done by using a new ensemble sinusoidal mechanism that automatically tunes F and Cr values. Their proposed algorithm was called LSHADE-EpSin, which it was later enhanced by using a mixture of a Cauchy distribution and two sinusoidal formulas, a restart mechanism that is used at the later generations and a new way to adapt the population size [28]. Mohamed *et al.* [29] proposed a new adaptation technique, semi-parameter adaptation approach, to tune the values of F and Cr in LSHADE algorithm. Their proposed algorithm was called, LSHADE-SPA. A new modified LSHADE algorithm, called LSHADE-RSP, that uses a rank-based selective strategy was proposed by Stanovov *et al.* [30].

Meng *et al.* [31] proposed a novel parameter adaptive DE (PaDE) algorithm to solve optimization problems. PaDE uses a novel adaptation mechanism to tune F and Cr values, a novel parabolic population size reduction mechanism to update the population size, and an enhanced mutation operator. A novel DE variant, called DE-NPC, was proposed by Meng *et al.* [32] that used a novel way to tune the F and Cr parameters based on the position information of the population. As previously mentioned, no single DE operator or algorithm was able to solve a wide range of optimization problems [7], [33]–[35]. As a result, researchers have developed many other algorithms that used more than one operators or algorithm. These algorithms are reviewed in the next sections.

1) MULTI-OPERATOR DE VARIANTS

The efficiency and performance of DE algorithms is greatly influenced by its mutation operators, which encourages many practitioners and researchers to try to find the most effective one [36]. Despite the numerous DE variants that exist in the literature, multiple research papers have shown that a single operator may not work well for a wide range of optimization problems. Moreover, given the weakness of single-operator DE algorithms (i.e., the variants that used a single mutation operator) to solve all forms of test problems, research on multi-operator DE variants have recently gained much more interest [4]. Fan *et al.* [37] introduced an auto-selection mechanism (ASM) to choose an appropriate DE variant to solve combinatorial optimization problems. They also integrated a learning strategy and selection probability to reinforce the probability of a particular DE variant being chosen and their results for selective data sets demonstrated the efficacy of their ASM method. In a similar analysis conducted in [4], an adaptive operator selection (AOS) technique was proposed. This AOS used the information of the problem's landscape and performance histories to automatically select the most appropriate strategy from a set of many. Because of their unstable performances during the optimization process, all the potential mutation strategies were used in each stage to evolve all the solutions in the entire population. The effectiveness of this methodology was revealed by solving 45 bound-constrained problems presented in the CEC2014 and CEC2015 competitions. Elsayed *et al.* [1] proposed a multi-method evolutionary framework that used CMA-ES and multi-operator DE in a single framework. Their proposed framework uses a fuzzy rule-based heuristic to emphasize the appropriate algorithm during the optimization task. It was demonstrated to be superior to many other algorithms through the solution of a number of bound-constrained optimization problems.

Instead of being limited to only bound constrained problems, the authors in [5] developed a new mechanism for automatically choosing the best combinations of DE parameters to solve constrained optimization problems. In addition, their proposed approach was also shown to be successful for selecting the search operators and handling multiple constraints. In the search to design high-quality DE algorithms, both low- and high-level ensembles of mutation strategies have also been gaining wide attention recently [38], [39]; an ensemble of DE algorithms has been proposed by Elsayed *et al.* [40] that used 16 different combinations of crossover and mutation operators, and a constraint-handling technique. An ensemble of multiple DE variants (EDEV) has been recently proposed by Wu *et al.* [38], in which three popular DE algorithms, namely JADE [22], CoDE [41] and EPSDE [42] are used. Their proposed approach inherits the merits from its constituents.

Zhang *et al.* [43] introduced a multi-layer competitive-cooperative (MLCC) framework to boost the performance of DE algorithm. In contrast to many other multi-method DE approaches, it could simultaneously implement a multi-population-based parallel structure to evolve an entire population. By integrating a scheme for selecting individual preference-based layers, they showed that their method could outperform many state-of-the-art DE algorithms. To boost the exploitation and exploration capabilities of the DE algorithm, an improved variant that uses three mutation operators with their corresponding self-adaptive strategy to tune the parameters was developed in [44]. In order to generate a new candidate solution, only one mutation operator was automatically chosen to produce that new solution.

Recently Sallam *et al.* [34] proposed an algorithmic framework that uses a number of mutation strategies is proposed to solve constrained optimization problems (COPs). In their proposed algorithm, the problem landscape information is used to select the best-performing DE mutation strategies during the optimization process. The computational experimental revealed the effectiveness of their proposed methodology in solving COPs. An improved variant of a multi-operator DE algorithm was also proposed by Sallam *et al.* [35], in which three mutation strategies and two crossover operators have been used, with two indicators, the diversity of population and quality of solutions, are used to automatically select the better-performing operator during the evolutionary task, which has been adapted to solve real-world constrained optimization problems [45]. A two-stage MODE, called TS-MODE, was recently proposed to solve combinatorial optimization problems [46], in which the first stage is responsible for exploration while the second one is responsible for exploitation.

In addition to choosing the best mutation mechanism, using an appropriate adaptive selection of DE operators method is also important as it can enhance the performance of DE for a constrained optimization problem [47]. Without such a scheme, a DE algorithm may be trapped in a local optimum which could also lead to premature convergence when solving hard constraints. To overcome such premature convergence, Lin *et al.* [48] proposed an adaptive immune-inspired multi-objective algorithm which was shown to be competitive in terms of improving its population diversity and exploration capabilities. In 2019, a multi-operator DE algorithm is proposed by Chen *et al.* [49], with an interior-point technique used as a local search.

B. MULTI-METHODS ALGORITHMS (MMA)

Fundamentally, in the case of low-level ensembles, different search strategies, operators or constraint handling techniques are combined, while amalgamating multi-operators or algorithms to solve optimization problems is referred to as a multi-methods algorithm (MMA) [50]. Although, multi-methods are more pertinent to low-level ensembles, they are also sometimes attributed as a special case of high-level ensembles [39]. To investigate different MMAs, Grobler *et al.* [51] adapted three alternative multi-methods approaches from the literature. After rigorous analyses with the same set of constituent algorithms, they showed that the heterogeneous meta-hyper-heuristic (HMHH) [52] algorithm was the most competitive. Strategically, the HMHH deals with selecting population-based meta-heuristic approaches, which can run in parallel, merely based on algorithmic logic [52]. Later on, Grobler *et al.* [53] integrated four meta-heuristic algorithms with HMHH to control heuristic space diversity, showing that their algorithm was competetive with a few state-of-the-art approaches. Vrugt *et al.* [54] proposed a co-variance matrix adaptation strategy, while they coalesced genetic algorithm and particle swarm optimization. They attempted to introduce a self-adaptive MMA which can control the generated number of offspring at each generation. After executing a good number of experiments with benchmark problems, they demonstrated the efficacy of their self-adaptive approach, particularly when the optimization problem is complex and high-dimensional.

Similar kind of study can be obtained from the work of Elsayed *et al.* [40], in which they developed a united multi-operator EA (UMOEA). UMOEA begins by splitting the initial population into a number of sub-populations with equal sizes. UMOEA consists of a number of multi-operators algorithms and based on the improvement rate, the best-performing one was chosen. UMOEA used an information sharing mechanism, which further facilitated superior searching. Later, Elsayed *et al.* [55] proposed an improved variant UMOEA, called UMOEAsII. For more information about multi-operator and multi-method frameworks, readers are requested to read research articles [39], [56], [57].

C. RL IN EAs

There has been an increasing trend to use RL concepts in optimization approaches. Although RL and EAs seem very different, they are both optimization techniques as RL seeks to maximize an agent's reward and EA aims to optimize a fitness function [58]. To boost the performance of a gray wolf optimization (GWO), Emary *et al.* [14] proposed a combination of a RL principle that could select the best combination of parameters for the GWO to obtain better exploitation and exploration rates, and a NN. An experience repository built to map each agent's state was later updated during the evolutionary process. Similarly, Sadhu *et al.* [15] used a Q-learning technique to adapt the firefly algorithm's parameters to ensure a balance between its exploration and exploitation rates. In it, the Q- learning strategy set the optimal parameter values for each firefly in a population during the learning phase and then applied it during execution. Based on numerous analyses, this strategy was shown to be competitive for solving real-world constrained optimization problems.

The literature includes research studies that combined Q-learning and EAs to boost the performance of RL for solving real-world application problems [59]. A Q-learning-based improved particle swarm optimization (PSO) algorithm for reducing the computational complexities of the classic RL technique was proposed by Daset al. [60]. Similarly Zhou [61] developed a fuzzy RL-based genetic algorithm (GA) which, with the global optimization capability of the GA, was capable of solving the local minimum problem in an actor-critic structure. Later, Liu and Zeng [62] options for reinforcement mutation to solve the traveling salesman problem (TSP). By integrating a RL mutation strategy, the convergence rate of their GA was shown to be faster than, and competitive with, many existing algorithms. Similarly, Alipour *et al.* [63] proposed a framework in which a combination of multi-agent RL and GAs was used to solve TSPs. Recently, Juang and Bui [64] introduced a reinforcement neural fuzzy surrogate-assisted multi-objective evolutionary optimization approach for a robot-learning control application. It was applied to an ant colony optimization algorithm to improve its learning efficiency.

To control the mutation strategies of DE, Sharma *et al.* [65] developed a double-deep Q-learning method as an AOS process. They used NN to predict which mutation operator should be used with DE at each generation for each parent. Later, Bora *et al.* [66] considered solving a multi-objective optimization problem using RL integrated with a Non-dominated sorting genetic algorithm (NSGA-II) algorithm which was implemented parameter-free self-tuning during the evaluation process. Also, Ning *et al.* [67] introduced a multi-objective EA in which RL was a generic parameter controller. More possible directions for RL-integrated EAs are highlighted in the review papers of Drugan [58], Cunha *et al.* [68] and Barto [69].

In [70], RL was used to choose one of the lower heuristics from 30 actions, with three non-overlapping ranges as the number of states. In its process, the given computational time is divided into a number of equal epochs ($|E|$), which was determined by applying the algorithm on the instances taken from CHeSC 2011. At each epoch, the process iteratively employed a selected action (heuristic selection - move acceptance combination) in an ILS framework and then applied Variable Neighborhood Descent. Although the algorithm showed good performance in three of the six CHeSC problem domains (SAT, FS, TSP), its performance was poor for the remaining three domains.

The above literature survey illustrates the considerable amount of research undertaken on DE and RL. However, little attention has been paid to integrating RL concepts in DE. Moreover, research on obtaining better DE mutation strategies for solving continuous optimization problems is still an open area. Therefore, as introducing RL (i.e., Q-learning) for selecting optimal DE mutation operators will certainly be a valuable addition to the existing knowledge base, it acts as a motivation for this research.

III. PROPOSED ALGORITHM (MARLwCMA)

In this section, details of the proposed algorithm (MARLwCMA) are presented. It should be noted that the proposed algorithm is generic and can be adopted with other evolutionary operators and/or algorithms.

The main steps of the proposed MARLwCMA are shown in Algorithm 1 in the flowchart in Figure 1, where NS and K are the number of states and actions, respectively. FES and MAX_{FES} are the number and maximum number of function evaluations, respectively. α and γ are the learning rate and discount factors.

MARLwCMA starts with defining the values of the parameters, and then an initial population of size NP is randomly generated and then evaluated with the probability of each algorithm being performed set to 1, i.e., $Prob_i = 1, \forall i = 1, 2$ and Q-table elements to zeros. Then, the whole population is divided into n_{alg} sub-population, (X_1, X_2) which have NP_1, NP_2 solutions, respectively. In each epoch, n_{alg} random numbers between 0 and 1, i.e., $rand_i \in [0, 1] \forall i = 1, 2$ are generated. If $rand_i \leq Prob_i$, then X_i is evolved by alg_i. Subsequently, the diversity and quality of solutions of X_1 are calculated and recorded, based on which the state of the population is determined as described in Section III-A1. Note that, each algorithm is used based on its assigned probability with the Q-table in MARL updated as discussed in Section III-A3

MARL and CMA-ES are applied in parallel to evolve their assigned sub-populations for CS epochs (cycle). Once a cycle is completed, their probabilities $Prob_{alg}$ are dynamically updated considering both their levels of diversity and performances, as discussed in Subsection III-D. Subsequently, for the next cycle and in each epoch (t), two random numbers are generated, i.e., $rand_{alg} \in [0, 1]$, $\forall alg = 1, 2$, based on which each algorithm is re-applied., i.e., if $rand_1 \leq Prob_1$, X_1 is evolved using MARL (see Section III-A). Similarly, if $rand_2 \leq Prob_2$, X_2 is evolved using CMA-ES. Note $rand_1$ and $rand_2$ are generated in a way that assures that at least one of them is less than one of the two probabilities, in order to make sure that at least one of algorithm is used in each epoch.

At the end of every second cycle, an information sharing scheme is conducted, in which the best solution from the superior sub-population replaces the worst one from the inferior sub-population. Also, as the performance of one algorithm is good at the early stages of the evolutionary process and bad at latter generations, both probabilities are re-set to 1; hence, the poor-performing algorithm may get a chance for improvement. This is done to give equal chance for both algorithms to rerun again. Therefore, both MARL and CMA-ES are executed for another CS generation and the whole process repeated.

An SQP is employed in the last generations of the optimization process in order to boost the convergence of the

Algorithm 1 MARLwCMA Algorithm

```
1: Define $NS$, $K$, $\alpha$, $\gamma$, $nop$, $Prob_{ls} \leftarrow 0.1$, $MAX_{FES} \leftarrow 10,000D$, $prob_1 \leftarrow 1$, $prob_2 \leftarrow 1$, $NP \leftarrow NP_1 + NP_2$, $c \leftarrow 0$ and $FES \leftarrow 0$;
2: Generate random initial population ($X$) of size $NP$;
3: Compute the objective function value of each solution in the initial population, $f(X)$;
4: Compute the number of fitness evaluations, $FES \leftarrow FES + NP$;
5: Randomly assign $NP_1$ and $NP_2$ individuals from $X$ to $X_i$, $\forall i = 1, 2$, respectively;
6: Calculate diversity and quality of $X_1$ and then determine its state, as described in Section III-A1;
7: Define $Q(s_t, a_t) \leftarrow 0$ for every state ($s_t$) and action ($a_t$) and randomly select one of the states from the set of states $S$;
8: **while** $FES \leq MAX_{FES}$ **do**
9:     $c \leftarrow c + 1$;
10:    **if** $c == CS$ **then**
11:        Update $prob_1$ and $prob_2$ as discussed in Section III-D;
12:    **end if**
13:    **if** $c == 2 \times CS$ **then**
14:        Apply information sharing between both algorithms as in Section III-E and reset $prob_1$, $prob_2$ and $c$;
15:    **end if**
16:    **if** $rand \in [0, 1] \leq Prob_1$ **then**
17:        Select best action ($a_t$) for current state ($s_t$) from the Q-table;
18:        Apply MARL to evolve entire $X_1$ as described in Section III-A
19:        Update $FES$, $FES = FES + NP_1$;
20:        Calculate immediate reward $r_t$ using Equation 8;
21:        Obtain maximum Q value for next state $s_{t+1}$;
22:        Update Q-table using Equation 10 and update current state $s_t \leftarrow s_{t+1}$;
23:    **end if**
24:    **if** $rand \in [0, 1] \leq Prob_2$ **then**
25:        Evolve $X_2$ using CMA-ES;
26:        update FES, ($FES = FES + NP_2$);
27:    **end if**
28:    **if** $FES \geq 0.75 \times MAX_{FES}$ **and** $FES \leq MAX_{FES}$ **then**
29:        Run SQP as discussed in Section III-F;
30:        Update $FES$
31:    **end if**
32: **end while**
```

proposed MARLwCMA, as discussed in Subsection III-F. MARLwCMA's steps are carried out until the maximum number of fitness evaluations are exhausted.

The main components of the proposed MARLwCMA are discussed in detail in the following sub-sections.

A. MARL

As previously stated, MARL is applied to evolve the first sub-population (X_1) with NP_1 individuals. It uses two mutation operators (DE/current-to-ϕbest with archive/1/bin and DE/current-to-ϕbest without archive/1/bin), which are selected because of their highly ranked performances for solving bound constrained optimization problems, as stated in [4]. They are used to construct three actions, as described in Section III-A2. Then, the RL is used to choose the best action from the list of actions to evolve the entire sub-population (X_1). To do this, the main points are: (1) how to represent the possible states of continuous problems; (2) which possible actions to consider; and (3) how to calculate the reward function of each action.

1) STATE REPRESENTATION

The two criteria used to determine the states of RL are the population diversity and improvement in the fitness function.

Let Div^t denote the population diversity in generation t calculated as

$$Div^t = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}(x_{i,j}^t - \bar{x}_{i,j})^2} \quad (1)$$

where NP is the population size, t the generation number and $\bar{x}_{i,j}$ the mean of the solutions' variables in the j^{th} dimension in generation t calculated by

$$\bar{x}_{i,j} = \frac{1}{NP}\sum_{i=1}^{NP} x_{i,j}^t \quad (2)$$

The quality ($Qual$) of solutions is expressed as the absolute difference between optimal value and the best fitness value, computed as:

$$Qual_k^t = |fit^* - fit_{t,k}^{best}|, \quad \forall k = 1, 2, \cdots, K \quad (3)$$

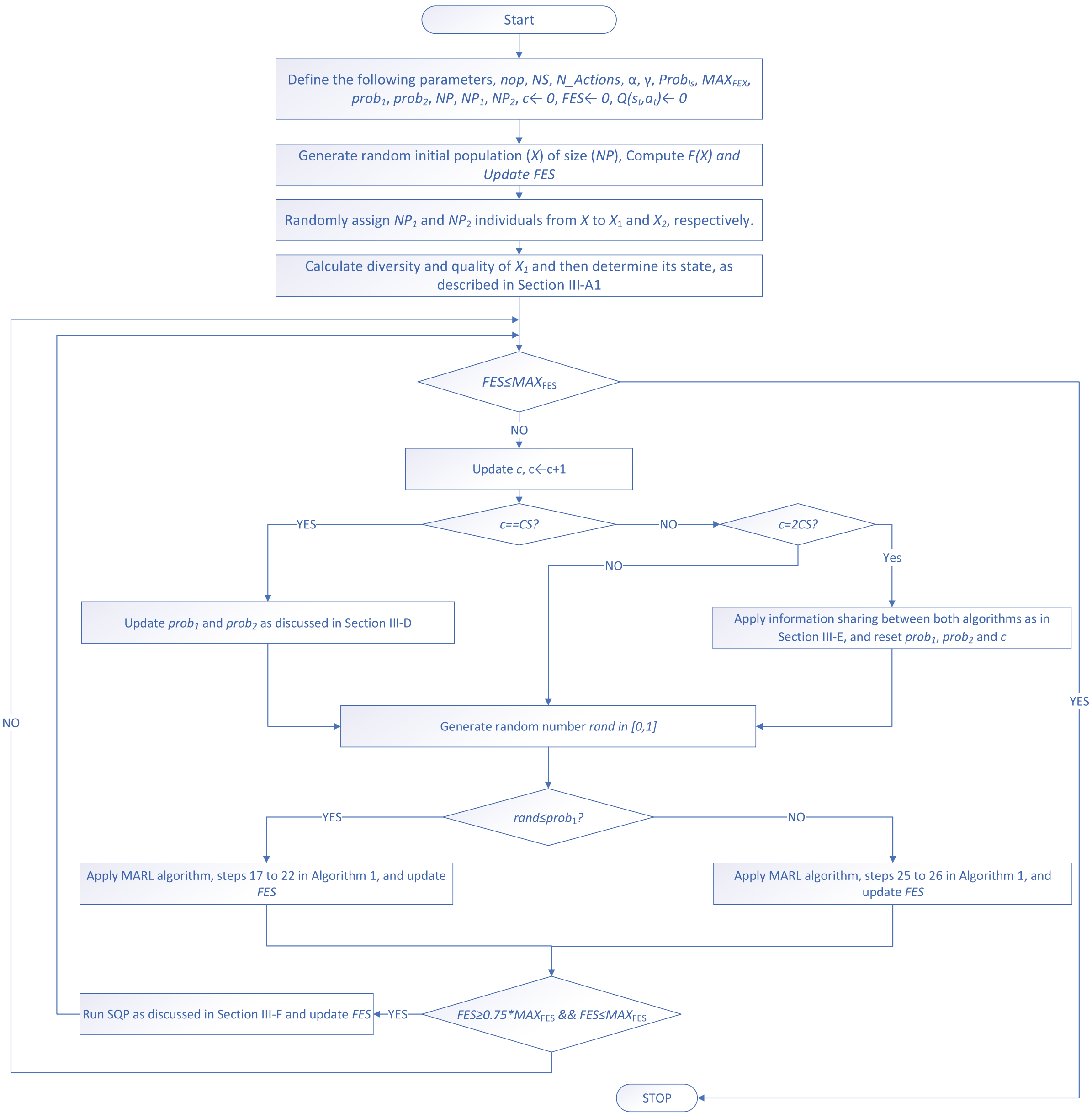

Start
Define the following parameters, nop, NS, N_Actions, α, γ, Prob_ls, MAX_FEX, prob1, prob2, NP, NP1, NP2, c← 0, FES← 0, Q(st,at)← 0
Generate random initial population (X) of size (NP), Compute F(X) and Update FES
Randomly assign NP1 and NP2 individuals from X to X1 and X2, respectively.
Calculate diversity and quality of X1 and then determine its state, as described in Section III-A1
FES≤MAX_FES
NO
Update c, c←c+1
c==CS?
YES
NO
c=2CS?
Yes
Update prob1 and prob2 as discussed in Section III-D
Apply information sharing between both algorithms as in Section III-E, and reset prob1, prob2 and c
Generate random number rand in [0,1]
rand≤prob1?
YES
NO
Apply MARL algorithm, steps 17 to 22 in Algorithm 1, and update FES
Apply MARL algorithm, steps 25 to 26 in Algorithm 1, and update FES
Run SQP as discussed in Section III-F and update FES
YES
FES≥0.75*MAX_FES && FES≤MAX_FES
NO
YES
STOP

FIGURE 1. **The general structure of the proposed MARLwCMA Algorithm.**

where fit^* is the function value of the optimal solution or best known solution, $fit^{best}_{t,k}$ the best fitness function value obtained by action k and K is the number of used actions.

Simultaneously, the improvement criteria (IC) is computed by:

$$IC^t_k = \frac{Qual^t_k}{fit^{best}_{t-1,k}}, \quad \forall k = 1, 2, \ldots, K \tag{4}$$

where $fit^{best}_{t-1,k}$ is the best fitness function value obtained by action k at generation $t-1$.

Subsequently, based on the above two criteria, the state of the population (s) can be defined by:

$$s = \left[\frac{Div^t}{Div^0}, \frac{IC^t}{IC^0}\right] \tag{5}$$

where Div^0 is the initial population's diversity and IC^0 its quality of the solutions.

As the state variables are continuous, it is crucial to partition the space to form some discrete spaces. It should be noted that the intervals of$(Div^t/Div^0) \in [0, 1]$ and (IC^t/IC^0) are $[0, 1]$ and it is possible to divide them into N_D and N_I sub-intervals. Therefore, the population's states can be classified into the $N_D \times N_I$. The performance of the RL method is highly dependent on the number of N_D and N_I, i.e, although the larger of their values may obtain a better control scheme, the number of state spaces will also be very large. Therefore, it is necessary to maintain a balance between the model's computational complexity and its performance. The analysis of the number of states is explained in Section IV-B2.

2) ACTIONS AND REWARD FUNCTION

In the proposed MARLwCMA algorithm, the DE mutation strategies represent the actions in the RL technique. As previously stated, in this paper, DE/current-to-ϕbest/1/bin with archive (equation 6) and DE/current-to-ϕbest/1/bin without archive (equation 7), are used as they perform well in solving bound constrained optimization problems [4]. Note that MARLwCMA algorithm is salable for the use of more DE mutation strategies (actions), with Section IV-B1) summarizes the effect of the number of operators in the performance of MARLwCMA.

- DE/current-to-ϕbest/1/bin with archive

$$u_{i,j} = \begin{cases} x_{i,j} + F_i.(x_{\phi,j} - x_{i,j} + x_{r1,j} - x_{r2,j}) \\ \qquad \text{if } (rand \leq Cr_i \text{ or } j = j_{rand}) \\ x_{i,j} \qquad \text{otherwise} \end{cases} \tag{6}$$

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

$$u_{i,j} = \begin{cases} x_{i,j} + F_i.(x_{\phi,j} - x_{i,j} + x_{r1,j} - x_{r3,j}) \\ \qquad \text{if } (rand \leq Cr_i \text{ or } j = j_{rand}) \\ x_{i,j} \qquad \text{otherwise} \end{cases} \tag{7}$$

where F and Cr are the scaling factor and crossover rate. r_1, r_2, r_3, are mutual random numbers and not equal to i. $\vec{x}_{r1}$ and $\vec{x}_{r3}$ are two solutions randomly chosen from the entire population, while $x_{\phi,j}$ are selected from the best 10% of individuals in the entire populations and $x_{r2,j}$ from the union of the archive and the entire population. In the proposed MARLwCMA, an archive is used to maintain the diversity of the population, with new individuals worse than their parents inserted into the archive [22]. To make a space for newly produced individuals, if the archive size is greater than its predefined size, the worst individuals are deleted from it.

Therefore, the number of actions will be three, and are represented by:

- the first uses the DE/current-to-ϕbest/1/bin with archive to evolve the entire population;
- the second applies the DE/current-to-ϕbest/1/bin without archive to produce new individuals from existing ones; and
- the third action uses the two DE mutation strategies to guide the whole population towards the optimal by splitting the whole population into two equal sub-populations each of them is evolved bu one the DE's operators.

Although a simple way of selecting a reward function $R(s, a)$ is based on the fitness function value, this may initially cause the RL algorithm to be trapped in a local solution. Therefore, $R(s, a)$ is calculated according to the improvement rate, that is the number of improved solutions in the current generation t, by:

$$R(s, a) = \frac{1}{NP} \sum_{i=1}^{NP} ((N_{i,a} = 1) - (N_{i,a} = 0)) \tag{8}$$

where N_a, the number of solutions evolved when applying action a, is computed as:

$$N_{i,a} = \begin{cases} 1, & f(x_i, a) - f(x_i) < 0 \\ 0 & f(x_i, a) - f(x_i) \geq 0 \end{cases} \tag{9}$$

where $f(x_i, a)$ is the fitness function value of the i^{th} solution using action a.

3) Q-LEARNING ALGORITHM

Q-learning is a model-free technique which considers the most efficient and productive RL method. Its main working principle depends on the reward or penalty fed back from the environment based on a state transition. In this study, Q-learning is adopted as representative of RL in a state-action table (Q-table) which is an $NS \times K$ matrix used as a reference when facing a similar situation in the future [15]. Table 1 shows its structure.

TABLE 1. The structure of Q-table.

State	Action			
	a_1	a_2	$\cdots$	a_K
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$\cdots$	$Q(s_1, a_K)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$\cdots$	$Q(s_2, a_K)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_{NS}	$Q(s_{NS}, a_1)$	$Q(s_{NS}, a_2)$	$\cdots$	$Q(s_{NS}, a_K)$

Let $A = [a_1, a_2, \ldots, a_K]$ represent the set of actions a learning agent can execute, $S = [s_1, s_2, \ldots, s_{NS}]$ the learning agent's set of states and r_{t+1} the immediate reward obtained from performing action a. Then, the total cumulative reward value ($Q(s_t, a_t)$) the learning agent gains at time t is computed by:

$$Q_{t+1}(s_t, a_t) = Q(s_t, a_t) + \alpha\, [r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)] \tag{10}$$

where $\alpha \in [0, 1]$ is the learning rate and $\gamma \in [0, 1]$ the discount factor, with the aim of γ to penalize future rewards. If $\gamma = 0$, the Q-learning algorithm considers only the current reward, but if $\gamma = 1$, it looks for a higher long-term one. Full statistical analyses of α and γ are presented in Sections IV-B4 and IV-B5, respectively.

The main steps in the MARL algorithm are presented in Algorithm 2.

B. CMA-ES

Over time, CMA-ES has verified its aptitude for dealing with different kinds of optimization problems [6], [71]. Following the concept of the self-adaptation of an evolutionary strategy, the CMA adopts the co-variance matrix of a multi-variable normal distribution. In CMA-ES, a Gaussian distribution is used to generate new solutions taking into account the path they take over as an alternative to using a single mutation step. Algorithm 3 presents the main steps in CMA-ES.

Algorithm 2 MARL Algorithm

```
1:  Define state $s_t \in S = s_1, s_2, \ldots, s_{NSs}$, action $a \in A = a_1, a_2, \ldots, A_{N\_actions}$ and $Q(s_t, a_t) \leftarrow 0$;
2:  Randomly select an initial state $s_t$;
3:  while $FES \leq MAX_{FES}$ do
4:      Choose best action $(a_t)$ (the one with maximum Q value), for state $(s_t)$ from Q-table;
5:      Evolve population using selected action$(a_t)$;
6:      Compute the fitness function value;
7:      Calculate immediate reward $(r_{t+1})$ using Equation (8);

8:      Update $FES \leftarrow FES + NP_1$;
9:      Update population $X_1$ using Equation (12);
10:     Update Q-table elements by Equation (10);
11:     Update state $(s_t \leftarrow s_{t+1})$.
12: end while
```

Algorithm 3 CMA-ES

```
1:  Generate initial individuals $(X_2)$ of size $NP_2$ and evaluate the objective function;
2:  Calculate weighted mean of population $(\overrightarrow{x}_m = \sum_{i=1}^{NP_2} w_i \overrightarrow{x}_{2,i})$, where $\sum_{i=1}^{NP_2} w_i = 1$ and $w_i = \frac{1}{NP_2}$, $\forall i = 1, 2, \ldots, NP_2$;
3:  Generate new solutions by sampling from Gaussian distribution as: $\overrightarrow{x}_{i,t+1} = N(\overrightarrow{x}_{m,t}, \sigma_t^2 C_t) = \overrightarrow{x}_m + \sigma_t N(0, C_t)$, $\forall i = 1 : NP_2$;
4:  Compute the objective function values of new solutions;
5:  Sort new solutions with respect to their objective function values;
6:  Select best $(\mu)$ solutions as a parental solutions, and compute their mean as $\overrightarrow{x}_{m,t+1} = \sum_{k=1}^{\mu} w_k \overrightarrow{x}_{k,t}$ where $\sum_{k=1}^{\mu} = 1$ and $w_1 \geq w_2 \geq \cdots \geq w_\mu$;
7:  Update evolution path $(p_{t+1}^s)$ and $(p_{t+1}^\sigma)$;
8:  Adapt co-variance matrix $(C_{t+1})$;
9:  Update global step size $(\sigma_{t+1})$;
10: Continue steps 3 to 7 until a stopping condition met.
```

C. POPULATION INITIALIZATION AND UPDATING TECHNIQUE

MARLwCMA starts with an initial population that is randomly generated by:

$$x_{i,j} = x_{i,j}^{min} + rand \times (x_{i,j}^{max} - x_{i,j}^{min}) \qquad i = 1, 2, \ldots, NP \text{ and } j = 1,\ 2,\ \ldots,\ D \tag{11}$$

where *rand* is a function used to produce random numbers in the range [0, 1], D the problem dimensions, $x_{i,j}^{min}$ and $x_{i,j}^{max}$ the lower and upper bounds of each decision variable.

During the optimization process, a linear population reduction size mechanism is used to dynamically decrement the population size of MARL (NP_1) as in Equation . This is done to maintain the diversity at the early stage of the optimization process and a fast convergence rate at latter stages [24].

$$NP_{1,t+1} = round[(\frac{NP_1^{min} - NP_1^{init}}{MAX_{FES}}) \times FES + NP_1^{init}] \tag{12}$$

where NP^{min} is the least number of solutions MARL can use, FES the current number of fitness evaluations (FES), MAX_{FES} the largest number of FES and NP^{init} is the size of the initial population.

D. UPDATING PROBABILITIES ($Prob_1$ AND $Prob_2$)

At the end of each cycle, the proposed MARLwCMA automatically selects which algorithm (MARL or CMA-ES) will be applied in the next cycle based on the values of two probabilities ($Prob_1$ and $Prob_2$). To update values of these probabilities, two indicator are used, the sub-population's diversity and the solutions' quality.

Concerning sub-population's diversity (D) (X_{alg}), which is the average deviation of each individual in it from the best individual, i.e.,

$$D_{CS,alg} = \frac{1}{NP_{alg}} \left(\sum_{i=1}^{NP_{alg}} dis(\overrightarrow{x}_{CS,alg,i} - \overrightarrow{x_{CS}}^{best}) \right), \quad \forall alg = 1, 2 \tag{13}$$

where $dis(\overrightarrow{x}_{CS,alg,i} - \overrightarrow{x_{CS}}^{best})$ is the Euclidean distance between best solution $(\overrightarrow{x_{CS}}^{best})$ and the i^{th} solution $(\overrightarrow{x}_{CS,alg,i})$ in X_{alg} at cycle (CS). Also the normalized diversity (DR_{alg}) for each sub-population is calculated by

$$DR_{CS,alg} = \frac{D_{CS,alg}}{\sum_{alg=1}^{2} D_{CS,alg}}, \quad \forall alg = 1, 2 \tag{14}$$

For the solutions' quality, at the end of each cycle (CS), the best solution in each sub-population is employed, based on which the normalized value is calculated as:

$$QR_{alg} = \frac{f(x_{CS,alg}^{best})}{\sum_{alg=1}^{2} f(x_{CS,alg}^{best})}, \quad \forall alg = 1, 2 \tag{15}$$

where $f(x_{CS,alg}^{best})$ is the objective value of the best solution obtained by algorithm *alg* at the end of the cycle (Cs).

The improvement rate value (IRV_{alg}) is computed using the above-mentioned equations as:

$$IRV_{alg} = (1 - QR_{alg}) + DR_{alg}, \quad \forall alg = 1, 2 \tag{16}$$

Note: the value of QR_{alg} is subtracted from one in order to meet the goal maximizing the IRV_{alg}.

Finally, the probability of using each algorithm is updated by:

$$Prob_{alg} = max\left(0.1, min\left(0.9, \frac{IRV_{alg}}{\sum_{alg=1}^{2} IRV_{alg}}\right)\right), \quad alg = 1, 2 \tag{17}$$

$Prob_{alg}$ are changed dynamically based on both diversity and quality of the solutions, with a minimum value (0.1) used to prevent any of them to be zero, so that the algorithm may have get improved in latter generations. Also, to ensure that at least one algorithm is running in each epoch, if the total sum of $IRV_{alg} = 0$, the value of both probabilities are set to 1.0 [2].

E. INFORMATION SHARING

As reported in [], sharing information among sub-populations improves the algorithm's performance. So, in this paper, a simple information sharing mechanism is used in which the best solution from the superior sub-population replaces the worst one from the inferior sub-population after every two cycles ($c = 2 \times CS$) in order to to encourage more effective searching [1], [2]. The values of the CMA-ES parameters are set to their default values, excluding σ which is computed by:

$$\sigma = \sigma \times \left(1 - \frac{FES}{FES_{max}}\right) \tag{18}$$

F. LOCAL SEARCH

In order to accelerate the proposed MARLwCMA convergence, sequential quadratic programming (SQP) is applied with a probability of $P_{ls} = 0.1$ to the best solution. SQP runs for up to CFE_{ls} fitness evaluations, in order to reduce the maximum number of fitness evaluations (CFE_{ls}) consumed. Therefore, it is applied in each epoch in the last 25% of the evolutionary task with a small probability (0.1) and, if there is no improvement in the objective function value, the probability of applying local search was set to a very small value, as discussed in steps 5-9 in Algorithm 4.

Algorithm 4 Local Search (SQP) Algorithm

```
1:  Required: the best obtained individual from the entire population $\overrightarrow{x}_{best}$;
2:  Generate a number randomly between 0 and 1 ($rnd \in [0, 1]$);
3:  if $rnd \leq P_{ls}$ then
4:      Run SQP to $\overrightarrow{x}_{best}$ for $CFE_{ls}$ objective function evaluations;
5:      if $f(\overrightarrow{x}_{sqp}) < f(\overrightarrow{x}_{best})$ then
6:          Update the value of $P_{ls}$, ($P_{ls} \leftarrow 0.1$);
7:          $\overrightarrow{x}_{best} \leftarrow \overrightarrow{x}_{sqp}$ and $f(\overrightarrow{x}_{best}) \leftarrow f(\overrightarrow{x}_{sqp})$;
8:      else
9:          Update the value of $P_{ls}$, ($P_{ls} \leftarrow 0.0001$);
10:     end if
11:     Update $FES$, ($FES \leftarrow FES + CEF_{ls}$);
12: end if
```

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In order to assess the effectiveness and performance of MARLwCMA, several experiments were carried out on 73 optimization functions (28 from the CEC2013 [18], 30 from the CEC2014 [19] and 15 from the CEC2015 [20]) competitions for single-objective optimization) with 10, 30 and 50 dimensions and a search space of $[-100, 100]^D$. MARLwCMA is also used to find the optimal solutions to problems with higher (100) dimensions taken from CEC2014 and CEC2017. Its performance is further tested by solving 22 real-application test problems taken from CEC2011.

The results obtained from the proposed algorithm (MARLwCMA) for the test problems from the CEC2014 and CEC2015 competitions were compared with those from the following existing-algorithms.

1) DE-based single-operators algorithms: the proposed MARLwMA is compared with Success History based DE with linear population size reduction (LSHADE) [24], Success History based DE (SHADE) [23], Adaptive DE with optional external archive (JADE) [22].
2) multi-operator-based algorithms: MARLwCMA is compared against DE with composite trial vector generation strategies (CoDE) [41], DE algorithm with an adaptation strategy (SaDE) [72], DE with an ensemble of parameters and mutation strategies (EPSDE) [42], multi-population ensemble DE (MPEDE) [57] and Landscape adaptive operator selection DE (LSAOSDE) [4].
3) multi-method-based algorithms: the proposed algorithm is compared with United multi-operator evolutionary algorithm (UMOEAs) [40], and A Multialgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO) [54];

Note that the results of other algorithms are obtained from [4].

Also for CEC2013 test problems, the results obtained from MARLwCMA were compared with those from the following existing algorithms.

1) single algorithms and operators: the proposed algorithm is compared with Self-adaptive of DE control parameters (jDE) [73], A sinusoidal differential evolution algorithm (SinDE) [74], Co-variance matrix adaptation evolution strategy with re-sampled Inheritance Search (CMA-ES-RIS) [75], DE with an individual-dependent mechanism (IDE) [76], DE with automatic parameter configuration (DE-APC) [77], A self-adaptive differential evolution with PBX crossover (MDE-pBX) [78] and SHADE [23].
2) multi-operator-based algorithms: the proposed algorithm is compared with CoDE [41], EPSDE [42] and SaDE [72].

Note that existing approaches which employed RL have been used to solve a different set of problems with different mechanisms used for defining the actions and states. Consequently, it was hard to compare them against our approach.

The proposed MARLwCMA algorithm was developed using Matlab R2018b and run on a computer with Windows 10 that has a core I7 processor with 3.4 GHz processor and 16 GB RAM. To ensure fair comparisons, the proposed algorithm and all other competing algorithms used the same stopping condition as stated in the competition guidelines. Based on the competition rules, the evaluation process of each generated solution is summarized as follows:

1) for each generated solution (x), the objective function f(x) value should be calculated at the generated solution (x).

2) the error, which measures how far the fitness of each solution from the known optimal value, is calculated using the following equation error=$|f(\overrightarrow{x}) - f(\overrightarrow{x^*})$, where $f(x^*)$ is the known optimal value.

Also, all the algorithms were run 51 times with stopping condition equal to $10,000 \times D$ fitness function evaluations or $|f(\overrightarrow{x}_{best}) - f(\overrightarrow{x^*})| \leq 1E-08$, where x_{best} the best solution achieved by the algorithm and x^* is the known optimal solution. For each single run, the difference between the obtained solution and the known optimal solution is set to zero, if its value is less than or equal to $1E-08$.

To statistically compare between algorithms test, Two non-parametric tests (Friedman ranking tests and the Wilcoxon signed-rank [79]) are used. Also, the performance of the proposed algorithm were also graphically assessed by plotting the performance profiles graph [80]. The performance profiles graph is a tool used to compare many algorithms (M) performance using set of test problems (P) and a comparison objective (i.e., the average number of FES or computational time) to achieve specific level of the performance condition (i.e., optimal objective function value). For an algorithm (A), the value of the performance profile Rho_A is computed by:

$$Rho_A(\tau) = \frac{1}{n_p} \times |p \in P : r_{p,A} \leq \tau| \tag{19}$$

where $Rho_A(\tau)$ is the ratio of $A \in M$ that the performance ratio $r_{p,m}$ is within a factor $\tau \in R$ for the best possible probability. Rho_A is a function that returns the cumulative distribution for the $r_{p,A}$.

A. PARAMETER SETTING AND ANALYSIS

Regarding the algorithms' parameters: for MODE NP_1^{init} was set to $18D$ individuals, NP^{min} to 4 individuals, the memory size (H) to 5 and archive rate (A) to 1.4 [4]; for CMA-ES, $\sigma = 0.3$, $\mu = NP_2/2$ and $NP_2 = 4 + \lfloor(3log(D))\rfloor$ [6]; for MARL, CS was set to 50, 100 and 150 epochs for test problems with 10, 30 and 50 dimensions [55], respectively. Regarding the local search, the maximum number of fitness evaluation that the SQP can run (FES_{LS}) was set to $0.2FES_{max}$ [1]. The technique proposed in [24] is used to manage the values of F and Cr.

B. ANALYSES OF ALGORITHM's COMPONENTS AND PARAMETERS

Variants of the proposed MARLwCMA using different values of its components and parameters were analyzed to determine the algorithm's final version. These analyses were carried out on 30D CEC2014 benchmark problems classified in four groups: $F01 - F_{03}$ are unimodal, $F_{04} - F_{16}$ simple multimodal, $F_{17} - F_{22}$ hybrid and $F_{23} - F_{30}$ composite.

1) EFFECT OF NUMBER OF DE OPERATORS (*nop*)

This effect was analyzed by conducting different experiments using variants with $nop = 2, 3$ and 4. Detailed results are presented in Table 1 in the supplementary material and the results of statistical comparisons obtained from a Wilcoxon test in Table 2. As there were no significant differences among all the variants, those with the fewest operators were preferred. The results obtained from a Friedman test shown in Table 3 demonstrated that the variant with $nop = 2$ obtained the best rank. Therefore, the number of actions was three (i.e., A1 = DE1, A2 = DE2 and A3 a combination of DE1 and DE2).

TABLE 2. Summary of comparisons of MARLwCMA variants with different ***nop*** values obtained from Wilcoxon test.

Algorithms	Better	Similar	Worse	Dec.
$nop = 2$ vs. $nop = 3$	11	10	9	$\approx$
$nop = 2$ vs $nop = 4$	13	9	8	$\approx$
$nop = 3$ vs $nop = 4$	11	9	10	$\approx$

TABLE 3. Average rankings of MARLwCMA variants with different ***nop*** values for all functions obtained by Friedman test.

$nop = 2$	$nop = 3$	$nop = 4$
1.88	2.02	2.10

Finally, one unimodal function ($F2$), two simple multimodal functions ($F06$ and $F12$), one hybrid function ($F19$) and one composition function ($F25$) are randomly selected to demonstrate the effectiveness of each operators (A_1, A_2 and A_3) as represented in Figure 2. As an example, for $F2$ A1 and A3 are selected many times during the evolutionary process, while A2 is selected less number of times.

2) EFFECT OF NUMBER OF STATES (*NS*)

This effect was checked by conducting different experiments with $NS = 4, 9, 16, 25$ and 36 for solving the 30D CEC2014 problems, with the values of the other parameters the same as those in Subsection IV-B1. Details of the obtained results are presented in Table 2 in the supplementary material and a summary is given in Table 4.

Finally, to rank these four different variants, the Friedman test was conducted, with the mean rankings shown in Table 5. It can be concluded from Tables 4 and 5 that the variant with $NS = 25$ was slightly better than that with $NS = 36$ and also better than all the others.

3) EFFECT OF PER

This effect was analyzed by running the proposed MARLwCMA with different Per values (i.e., 0.15, 0.25 and 0.5) to solve 30D problems. Detailed obtained results provided in Table 3 in the supplementary material. Subsequently, the Wilcoxon and Friedman tests were conducted, the results from which are summarized in Tables 6 and 7, respectively. Based on them, it was concluded that the variant with $Per = 0.25$ was the best.

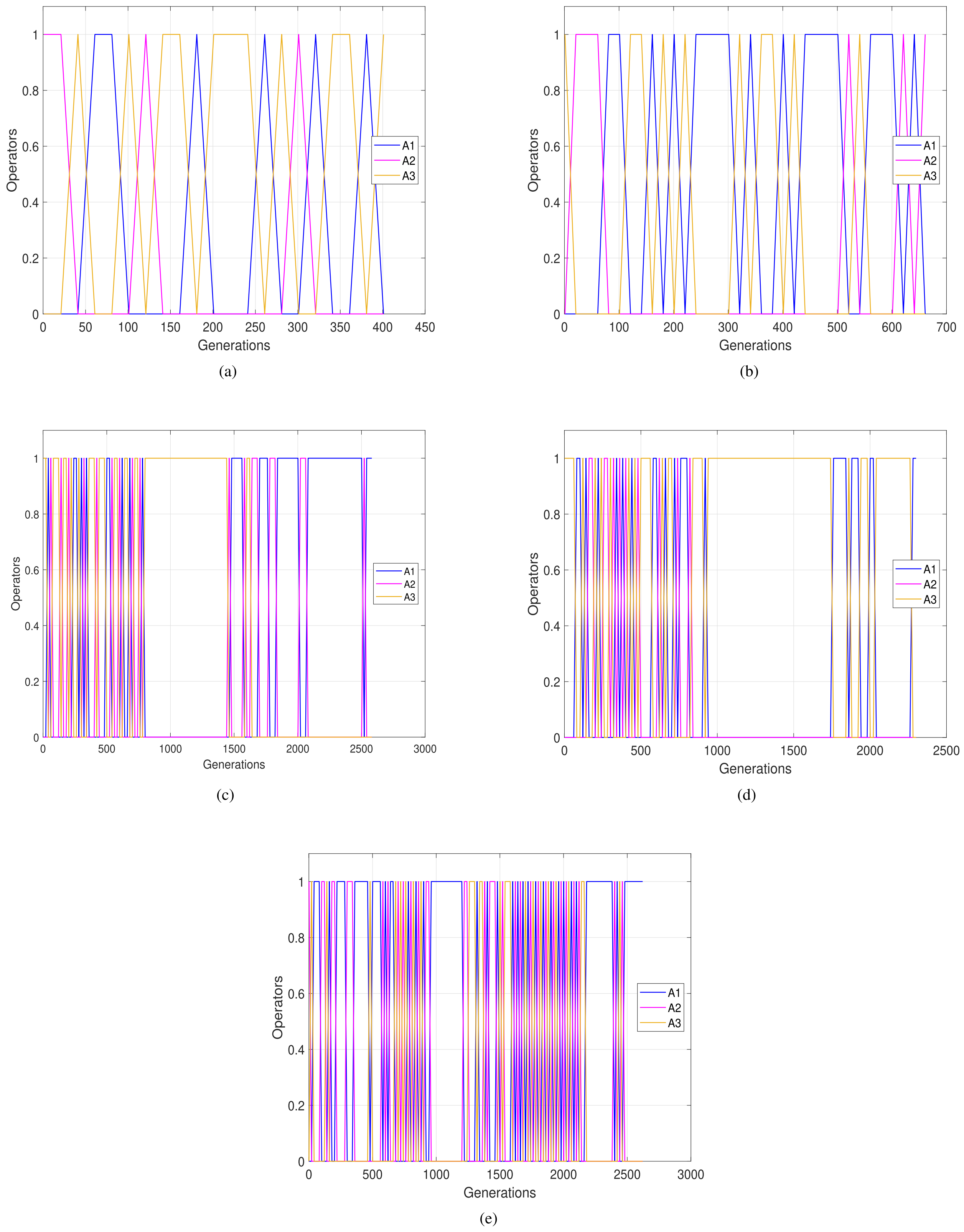


FIGURE 2. **Effect of each action on the performance of the optimization process for (a) F2; (b) F06; (c) F12; (d) F19; and (e) F25.**

4) EFFECT OF LEARNING RATE (α)

This effect was analyzed by conducting several experiments with $\alpha = 0.15, 0.25$ and 0.5, and setting the other parameters to their values stated in Section IV-B3. Table 4 in the supplementary material provides detailed results.

TABLE 4. Summary of comparisons of MARLwCMA variants with different *NS* values.

Algorithms	Better	Similar	Worse	Dec.
$NS = 4$ vs. $NS = 9$	12	11	7	$\approx$
$NS = 4$ vs. $NS = 16$	12	11	7	$\approx$
$NS = 4$ vs. $NS = 25$	10	11	9	$\approx$
$NS = 4$ vs. $NS = 36$	9	11	10	$\approx$
$NS = 9$ vs. $NS = 16$	12	11	7	$\approx$
$NS = 9$ vs. $NS = 25$	6	11	13	$\approx$
$NS = 9$ vs. $NS = 36$	6	11	13	+
$NS = 16$ vs. $NS = 25$	5	11	14	$\approx$
$NS = 16$ vs. $NS = 36$	6	11	13	$\approx$
$NS = 25$ vs. $NS = 36$	10	11	9	$\approx$

TABLE 5. Average rankings of MARLwCMA with different *NS* values for all functions obtained by Friedman test.

$NS = 4$	$NS = 9$	$NS = 16$	$NS = 25$	$NS = 36$
2.83	3.23	3.43	**2.73**	2.77

The Wilcoxon test results presented in Table 8 demonstrate that the variant with $\alpha = 0.25$ was better than all the others, a conclusion confirmed by the results obtained from the Friedman test shown in Table 9.

5) EFFECT OF DISCOUNT FACTOR (γ)

This effect was analyzed by conducting several experiments with $\gamma = 0.80, 0.85, 0.90$ and 0.95, with detailed results presented in Table 5 in the supplementary material. The Wilcoxon test was conducted to compare the variants and the results provided in Table 10 demonstrate that the one with $\gamma = 0.85$ was the best, as do the average rankings obtained from the Friedman test presented 11.

C. COMPARISONS OF RESULTS OBTAINED FROM MARLwCMA AND ITS CONSTITUENT ALGORITHMS (MARL AND CMA-ES)

The comparison of the obtained results from the proposed MARLwCMA and its constituents (MARL and CMA-ES) is conducted in this section. The best parameter values determined, that is, $nop = 2$, $NS = 25$, $Per = 0.25$, $\alpha = 0.25$ and $\gamma = 0.85$ and those of the other parameters presented in Section IV-A were considered, with details of the best, average and standard deviation results reported in the supplementary material.

1) 10D RESULTS

Details of the average fitness errors ($|f(\overrightarrow{x}_{best}) - f(\overrightarrow{x^*})$) and their standard deviations (Std.) achieved by running MARLwCMA are provided in Table 6 in the supplementary material.

For the unimodal test functions (F01-F03), MARLwCMA worked very well for all of them, obtaining the optimal optimal best and mean results. Considering the simple multimodal test functions (F04-F16), MARLwCMA able to obtain optimal solutions for 7 test functions, and very close results

TABLE 6. Summary of comparisons of MARLwCMA variants with different *Per* values.

Algorithms	Better	Similar	Worse	Dec.
$Per = 0.15$ vs. $Per = 0.25$	5	13	12	$\approx$
$Per = 0.15$ vs $Per = 0.5$	10	12	8	$\approx$
$Per = 0.25$ vs $Per = 0.5$	12	13	5	+

TABLE 7. Average rankings of MARLwCMA variants with different *Per* values for all functions obtained by Friedman test.

$Per = 0.15$	$Per = 0.25$	$Per = 0.5$
2.08	**1.77**	2.15

TABLE 8. Summary of comparisons of MARLwCMA variants with different α values.

Algorithms	Better	Similar	Worse	Dec.
$\alpha = 0.15$ vs. $\alpha = 0.25$	6	13	11	$\approx$
$\alpha = 0.15$ vs $\alpha = 0.5$	12	12	6	$\approx$
$\alpha = 0.25$ vs $\alpha = 0.5$	14	13	3	+

TABLE 9. Average rankings of MARLwCMA variants with different α values for all functions obtained by Friedman test.

$\alpha = 0.15$	$\alpha = 0.25$	$\alpha = 0.5$
1.98	**1.73**	2.28

TABLE 10. Summary of comparisons of MARLwCMA variants with different γ values.

Algorithms	Better	Similar	Worse	Dec.
$\gamma = 0.80$ vs. $\gamma = 0.85$	6	13	11	$\approx$
$\gamma = 0.80$ vs $\gamma = 90$	10	13	7	$\approx$
$\gamma = 0.80$ vs $\gamma = 95$	8	14	8	$\approx$
$\gamma = 0.85$ vs $\gamma = 90$	9	13	8	$\approx$
$\gamma = 0.85$ vs $\gamma = 95$	10	13	7	$\approx$
$\gamma = 0.90$ vs $\gamma = 95$	7	13	10	$\approx$

TABLE 11. Average rankings of MARLwCMA variants with different γ values for all functions obtained by Friedman test.

$\gamma = 0.80$	$\gamma = 0.85$	$\gamma = 0.90$	$\gamma = 0.95$
2.53	**2.35**	2.62	2.50

to optimal for the remaining test functions. It also performed very well for the hybrid functions but converged to local solutions for the composite ones.

In terms of quality of solutions presented in Table 12, MARLwCMA obtained better results than was CMA-ES and MARL and, regarding the average results, was better than CMA-ES and MARL for 23 and 18 test functions, respectively, but was worse to them for 2 and 7 test functions, respectively.

From the Wilcoxon test results, MARLwCMA was statistically significantly superior to CMA-ES but there was no significant differences among the best results obtained. As a further analysis, the Friedman test was carried out to rank all the algorithms based on the obtained mean results, with the rank values depicted in Table 13, which show that MARLwCMA was ranked first.

TABLE 12. Summary of comparisons of MARLwCMA against CMA-ES and MARL for 10D, 30D and 50D test functions taken from CEC2014 competition based on Wilcoxon test results.

Algorithms	Criteria	10D				30D				50D			
		Better	Similar	Worse	(P-value, Dec.)	Better	Similar	Worse	(P-value, Dec.)	Better	Similar	Worse	(P-value, Dec.)
MARLwCMA vs. CMA-ES	Best	21	8	1	(0.000,+)	21	6	3	(0.001, +)	17	9	4	(0.009, +)
	Average	23	5	2	(0.000, +)	23	5	2	(0.000, +)	21	5	4	(0.004, +)
MARLwCMA vs. MARL	Best	10	10	10	(0.263,≈)	17	6	7	(0.024 +)	17	5	8	(0.021, +)
	Average	18	5	7	(0.009, +)	20	4	6	(0.007, +)	20	3	7	(0.003, +)

TABLE 13. Average rankings of MARLwCMA, CMA-ES and MARL for all dimensions of CEC2014 benchmark problems obtained by Friedman test.

Algorithm	10D		30D		50D	
	Best	Mean	Best	Mean	Best	Mean
CMA-ES	2.62	2.63	2.43	2.57	2.32	2.48
MARL	1.72	1.90	2.03	2.02	2.05	2.02
MARLwCMA	**1.67**	**1.47**	**1.53**	**1.42**	**1.63**	**1.50**

2) 30D RESULTS

Details of the results obtained from MARL-wCMA are presented in Table 7 in the supplementary material. For unimodal functions, MARLwCMA obtains the optimal solutions for all them for both best and mean results. For simple simple multimodal test functions, it obtains optimal solutions five test functions (F04, F06, F07, F08 and F10), very close solutions to optima for F12 to F15, but its performance decreased for F5 and F11. For the hybrid test functions functions, although MARLwCMA achieved very close results to the optimal for F18 and F22, its result for F17 were slightly worse. Similar to the 10D test functions, MARLwCMA became trapped in local solutions when solving composite test functions.

The quality of the solutions obtained are shown in Table 12. Considering the best results, MARLwCMA was superior, similar and inferior to CMA-ES for 21, 6 and 3 test problems, respectively and to MARL for 17, 6 and 7, respectively. In terms of the average results, it was better and worse than CMA-ES for 21 and 4, respectively, and than MARL for 20 and 7, respectively.

Also to check the statistical differences among the competing algorithms, the Wilcoxon test was carried out, with the results recorded in Table 12 showing that MARLwCMA is statistically better than its constituent parts. As a further analysis, the Friedman test is conducted to rank all algorithms with the obtained results presented in Table 13. From Table 13, it can be concluded that thr proposed MARLwCMA was ranked first.

Furthermore, the performance profiles graph was plotted to compare MARLwCMA, CMA-ES and MARL, with the results presented in Figure 3. It is obvious that MARLwCMA was better than its constituent algorithms as it first reached a probability of 1.0 at $\tau \approx 37$.

3) 50D RESULTS

Table 8 in the supplementary material presents the details results obtained by the proposed algorithm. For the unimodal test functions, MARLwCMA obtained the optimal solutions for F02 and F03 and very close to an optimal one for F01. Considering the simple multimodal test problems, it achieved optimal results and very close results for 3 and 6 test functions, respectively, of them, close to optimal for F16 and poor for only F11 and F17 while its performances for the hybrid functions were reasonable. For the composite test functions, similar to the 10D and 30D problems, it became trapped in local minima close to the optimal solutions but its mean results for F29 and F30 deviated more from the global solutions.

Regarding the quality of solutions achieved, MARLwCMA obtained significantly better results than CMA-ES and MARL, as confirmed by the results presented in Table 12. In terms of the best results, it was superior, inferior and similar to CMA-ES for 17, 4 and 9 problems, respectively, and to MARL for 17, 5 and 8, respectively. Its average results were better, worse and similar to those of CMA-ES for 21, 4 and 5 test problems, respectively, and to those of MARL for 20, 7 and 3 test functions, respectively.

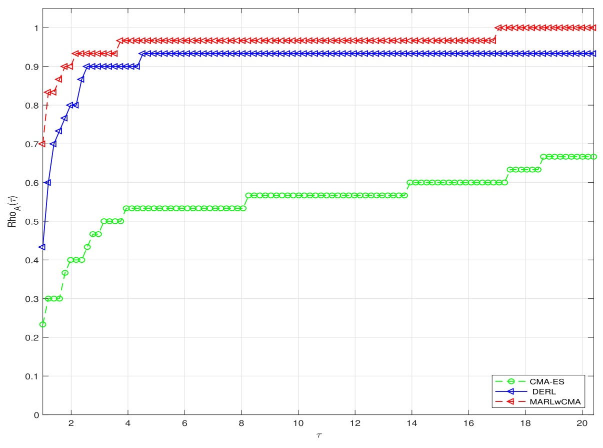
Considering the Wilcoxon results, MARLwCMA was statistically superior to other algorithms. Also, it was ranked first according to the Friedman test results presented in Table 13. As a further comparison, a graph of the performance profiles was plotted to compare the algorithms (Figure 3). It is clear that MARLwCMA reached a probability of 1 first at $\tau \approx 95$.

D. COMPARISONS OF MARL AND ITS DE VARIANTS

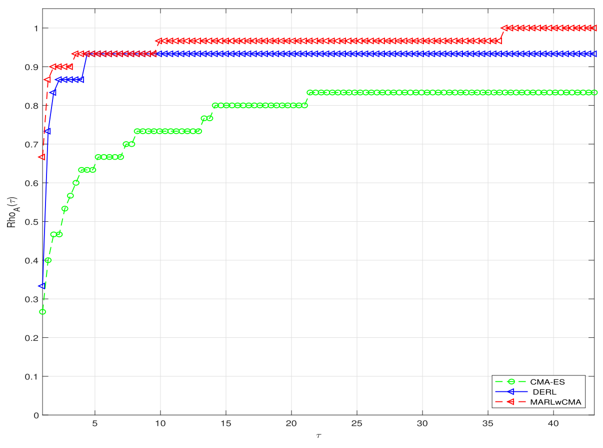
To test the effectiveness of using RL to select the best DE operators, the computational results obtained by MARL were compared. The 30D CEC2014 test problems were solved using the three different actions described in Section III-A2, with Var1, Var2 and Var3 denoting the first, second and third actions, respectively.

The summary of the results obtained from MARL and its different variants is presented in Table 14, from which MARL was superior, equal and inferior to Var1 for 12, 10 and 8 test functions, respectively, to Var2 for 13, 10 and 7, respectively, and to Var3 for 12, 10 and 8, respectively. For the mean obtained results, MARLwCMA was superior to Var1, Var2 and Var3 for 17, 15 and 16, respectively, similar for 5, 7 and 6, respectively, and worse for 8, 8 and 8 test problems, respectively.

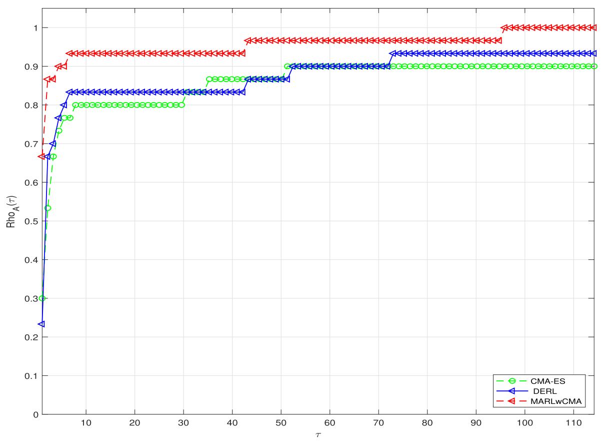
Regarding the obtained average results, MARL performed significantly better than all its variants, based on the Wilcoxon test as shown in Table 14. Further analysis was conducted using the Friedman rank test, with the results presented in Table 15 showing that MARL performed the best for both best and mean results.



(a) 10D test problems

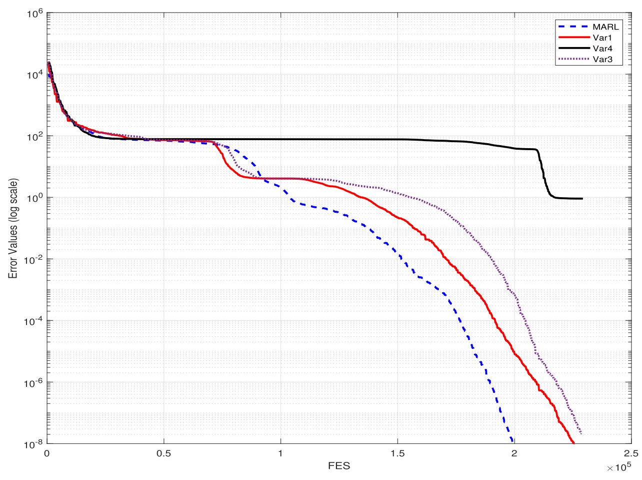


(b) 30D Test Problems

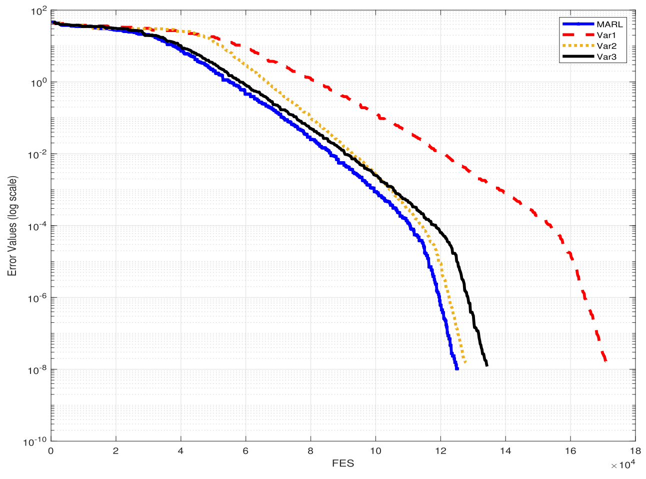


(c) 50D test Problems

FIGURE 3. Performance profiles of MARLwCMA, CMA-ES and MARL based on CEC2014 unconstrained problems for (a)10D problems; (b) 30D problems; and (c) 50D problems.



(a) F04



(b) F06

FIGURE 4. Convergence graphs for MARL, Var1, Var2 and Var3 on CEC2014 with 30D problems; (b) F04; and (c) F06.

TABLE 14. Summary of comparisons of performances of MARL and its DE variants for 30D CEC2014 test problems.

Algorithms	Criteria	Better	Similar	Worse	Dec.
MARL vs. Var1	Best	12	10	8	≈
	Average	17	5	8	+
MARL vs. Var2	Best	13	10	7	≈
	Average	15	7	8	+
MARL vs. Var3	Best	12	10	8	≈
	Average	16	6	8	+

Also, the graphs plotted to show the convergences of the proposed MARL and its constituent DE algorithms are shown in Figure 4, from which it is clear that MARL converges faster than other algorithms.

The results confirmed that MARL is statistically superior to its constituent DE algorithms which demonstrated the capability of RL to select the appropriate action during the optimization process, with another reason maybe the use of good DE operators.

TABLE 15. Average rankings of MARL, Var1, Var2 and Var3 for 30D CEC2014 problems obtained by Friedman test.

Algorithms	Var1	Var2	Var3	MARL
Best	2.55	2.70	2.50	**2.25**
Mean	2.60	2.75	2.53	**2.12**

E. COMPARISONS OF PERFORMANCES OF MARLwCMA AND STATE-OF-THE-ART ALGORITHMS FOR CEC2014 PROBLEMS

The performance of the proposed MARLCMA was compared with those of several rival algorithms, with detailed results for the best, average and standard deviation values reported in the supplementary material.

1) 10D RESULTS

The average fitness errors $|f(\overrightarrow{x}_{best})-f(\overrightarrow{x^*})$ and the standard deviations obtained from the proposed MARLwCMA and the competing algorithms are provided in Table 9 in the supplementary material. As it is clear from that table the proposed MARLwCMA, LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO and LSAOSDE are able to obtain better results for 17, 8, 5, 6, 8, 6, 5, 5, 4, 4 and 10 test functions, respectively. MARLwCMA obtains the optimal solutions for all the unimodal functions (F_1 to F_3). For the simple multimodal functions ($F04$ to $F16$), the proposed algorithm achieves better results for 6 test functions. It obtains the optimal solution to three functions ($F06$, $F07$ and $F08$) and a very close value to the optimal for $F04$, $F12$, $F15$ and $F16$. For the hybrid functions ($F17$ to $F22$), the performance of the proposed MARLwCMA is not good in comparison to CoDE, and LSAOSDE. CoDE is better than the MARLwCMA in 5 test functions, while it is worse than it in 1 test functions. LSAOSDE is superior to MARLwCMA in 4 test functions and inferior to it in 2 test functions. Considering the composition functions ($F23$ to $F30$), the proposed MARLwCMA obtains better results than other algorithms in 6 test functions. To sum up, the proposed algorithm is suitable for unimodal, multimodal and composition functions, while its performance deteriorates when solving hybrid functions.

In terms of the solutions' quality presented in Table 16, MARLwCMA was better than LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO and LSAOSDE for 16, 22, 21, 18, 20, 23, 18, 24, 25 and 14 test functions, respectively, and worse to them for 8, 3, 3, 7, 5, 2, 6, 2, 3 and 10 test functions, respectively. In terms of the Wilcoxon test, the proposed MARLwCMA algorithm is statistically better than all the other algorithms except LSAOSDE for which there was no significant difference but there was a bias towards MARLwCMA in terms of its number of better results. Considering the Friedman test results in Table 17.MARLwCMA is ranked first followed by LSAOSDE.

Plots of the performance profiles are depicted in Figure 5, in which it can be seen that MARLwCMA was better than all the competing algorithms because it able to attain a probability of 1.0 at $\tau = 500$.

2) 30D RESULTS

Table 10 in the supplementary material presents details of the results obtained from MARLwCMA and othe competing algorithms. It is clear from that table that the proposed MARLwCMA obtains the best results for 16 test functions, while LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO, and LSAOSDE achieve the best results for 8, 5, 6, 8, 6, 6, 6, 4, 5 and 10 test functions, respectively. The proposed MARLwCMA obtains the optimal solutions for the unimodal test functions ($F01$to $F03$). Considering the simple multimodal functions ($F04$ to $F16$), MARLwCMA achieves the optimal solutions for $F06$, $F07$ and $F08$ and a very close results to the optimal for $F04$, $F10$, $F12$, $F13$, $F14$, and $F15$. It can be observed that, MARLwCMA attains the best results in 6 test functions. For the hybrid functions ($F17$ to $F22$), similar to 10D, the proposed MARLwCMA is performing poorly in comparison to CoDE and LSAOSDE. In regards to the composition functions ($F23$ to $F30$), MARLwCMA achieves the best for 6 test functions, while the number of best results obtained from LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO, and LSAOSDE are 1, 1, 1, 1, 1, 1, 1, 1, 2 and 2 test functions, respectively. Obviously, for the 30D test problems, MARLwCMA demonstrates more superiority on unimodal, simple multimodal and hybrid functions.

TABLE 16. Summary of comparisons of MARLwCMA against LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO and LSAOSDE for 10D, 30D and 50D test problems taken from CEC2014 competition.

Algorithms	10D				30D				50D			
	Better	Similar	Worse	P-value, Dec.	Better	Similar	Worse	P-value, Dec.	Better	Similar	Worse	P-value, Dec.
MARLwCMA vs. LSHADE	16	6	8	(0.026,+)	17	6	7	(0.045,+)	18	5	7	(0.009,+)
MARLwCMA vs. SHADE	22	5	3	(0.001,+)	23	4	3	(0.000,+)	24	4	2	(0.000,+)
MARLwCMA vs. JADE	21	6	3	(0.001,+)	23	5	2	(0.000,+)	24	3	3	(0.000,+)
MARLwCMA vs. CoDE	18	5	7	(0.021,+)	24	5	1	(0.000,+)	25	2	3	(0.000,+)
MARLwCMA vs. SaDE	20	5	5	(0.009,+)	26	2	2	(0.000,+)	28	1	1	(0.000,+)
MARLwCMA vs. EPSDE	23	5	2	(0.000,+)	23	5	2	(0.000,+)	26	1	3	(0.001,+)
MARLwCMA vs. MPEDE	18	6	6	(0.027,+)	21	4	5	(0.000,+)	24	3	3	(0.000,+)
MARLwCMA vs UMOEAs	24	4	2	(0.000,+)	21	4	5	(0.000,+)	21	4	5	(0.000,+)
MARLwCMA vs. AMALGAM-SO	25	2	3	(0.000,+)	23	2	5	(0.003,+)	27	1	2	(0.000,+)
MARLwCMA vs. LSAOSDE	14	6	10	(0.265,≈)	16	5	9	(0.412,≈)	23	4	3	(0.002,+)

TABLE 17. Average rankings of all algorithms for all dimensions of CEC2014 benchmark problems obtained by Friedman test.

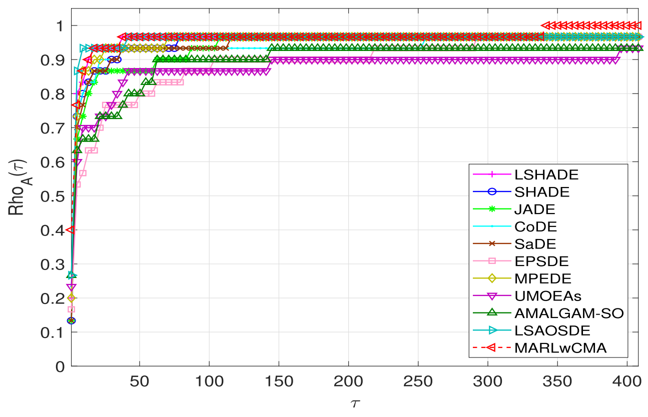
Algorithms	Rank for 10D	Rank for 30D	Rank for 50D
LSHADE	4.78	4.37	4.42
SHADE	6.45	6.55	6.27
JADE	6.35	6.83	6.68
CoDE	5.40	6.97	6.98
SaDE	6.97	8.43	8.43
EPSDE	7.82	7.85	7.78
MPEDE	5.72	4.92	5.03
UMOEAs	7.38	7.02	5.92
AMALGAM-SO	7.50	6.38	7.98
LSAOSDE	4.12	3.62	3.97
MARLwCMA	**3.52**	**3.07**	**2.53**

Regarding the quality of solutions, as shown in Table 16, MARLwCMA was superior to LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO and LSAOSDE for 17, 23, 23, 24, 26, 23, 21, 21, 23 and 16 test problems, respectively, similar for 6, 4, 5, 5, 2, 5, 4, 4, 2 and 3 and inferior for 7, 3, 2, 1, 2, 2, 5, 5, 5 and 9 test functions, respectively.

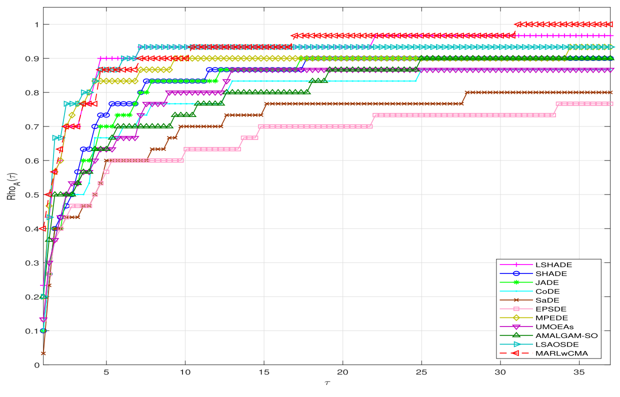
In terms of The Wilcoxon test, as presented in Table 16, the proposed MARLwCMA algorithm is statistically better than all other competing algorithms, except LSAOSDE for which there was no statistically significant difference but a bias towards MARLwCMA in terms of the number of better functions. The Friedman test was also carried out with the mean rank results presented in Table 17 revealed that MARLwCMA is ranked first and LSAOSDE second. Finally, the performance profiles graph is plotted in Figure 5, from which it can be seen that the proposed MARLwCMA algorithm is better than other algorithm, because it reaches a probability of 1.0 first at $\tau \approx 32$.

3) 50D RESULTS

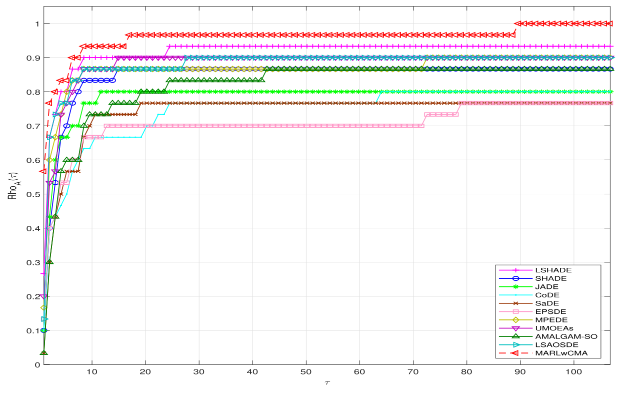
Table 11 in the supplementary material presents the detailed results obtained by MARLwCMA and the other algorithms. The last row of that table indicates the number of best solutions obtained by eavery algorithm in comparison to other algorithms. MARLwCMA attains the best results for 19 test functions, while the number of best solutions achieved by LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO, and LSAOSDE are 9, 4, 4, 1, 0, 3, 4, 7, 1 and 4 test functions, respectively. For unimodal functions, MARL obtains the optimal solutions for $F02$ and $F03$ and obtains a very close solution to the optimal for $F01$. UMOEAs obtains the optimal solutions for all the unimodal functions. Considering the simple multimodal test functions, the proposed MARLwCMA achieves the best results for 7 test functions, while LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO, and LSAOSDE obtain the best for 5, 1, 1, 0, 0, 0, 2, 3, 1 and 1 test functions respectively. Considering the hybrid functions, MARLwCMA is able to obtain the best results for 4 test functions, while the number of best solutions obtained by LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE,



(a) 10D test problems



(b) 30D Test Problems



(c) 50D test Problems

FIGURE 5. Performance profiles of MARLwCMA and the rival algorithms algorithms based on CEC2014 unconstrained problems for (a)10D problems; (b) 30D problems; and (c) 50D problems.

UMOEAs, AMALGAM-SO, and LSAOSDE are 2, 0, 0, 0, 0, 0, 0, 0, 0, and 0 test functions, respectively. Obviously, the superiority of the proposed MARLwCMA increased with the increase of problem dimensions when solving hybrid functions. Finally, for the composition functions, MARLwCMA achieves the optimal solutions for 6 test functions.

TABLE 18. Summary of comparisons of MARLwCMA against SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE- APC, SHADE and MDE-pBX for 10D, 30D and 50D test problems taken from CEC2013 competition.

Algorithms	10D				30D				50D			
	Better	Similar	Worse	P-value, Dec.	Better	Similar	Worse	P-value, Dec.	Better	Similar	Worse	P-value, Dec.
MARLwCMA vs. SinDE	18	6	4	(0.003,+)	24	3	1	(0.000,+)	21	5	2	(0.001,+)
MARLwCMA vs. CMA-ES-RIS	19	5	4	(0.026,+)	20	3	5	(0.000,+)	20	1	7	(0.002,+)
MARLwCMA vs. IDE	18	6	4	(0.011,+)	21	5	2	(0.001,+)	21	5	2	(0.000,+)
MARLwCMA vs. CoDE	25	2	1	(0.000,+)	23	3	2	(0.000,+)	23	3	2	(0.000,+)
MARLwCMA vs. EPSDE	19	5	4	(0.002,+)	22	5	1	(0.000,+)	20	5	3	(0.000,+)
MARLwCMA vs. jDE	19	4	5	(0.011,+)	21	5	2	(0.000,+)	22	5	1	(0.000,+)
MARLwCMA vs. SaDE	18	7	3	(0.006,+)	22	4	2	(0.000,+)	22	3	3	(0.000,+)
MARLwCMA vs DE-APC	19	6	3	(0.001,+)	24	3	1	(0.000,+)	24	2	2	(0.000,+)
MARLwCMA vs. SHADE	18	8	2	(0.001,+)	21	4	3	(0.000,+)	21	3	4	(0.001,+)
MARLwCMA vs. MDE-pBX	25	2	1	(0.000,+)	25	2	1	(0.000,+)	25	2	1	(0.000,+)

Regarding the quality of solutions achieved, MARLwCMA was the best of all the algorithms, as confirmed by the results provided in Table 16. It was superior, inferior and similar to LSHADE for 18, 7 and 5 test problems, respectively, to JADE for 24, 3 and 3, respectively, to SHADE for 24, 2 and 4, respectively, to CoDE for 25, 3 and 2, respectively, to SaDE for 28, 1 and 1, respectively, to EPSDE for 26, 3 and 1, respectively, to MPEDE for 24, 3 and 3, respectively, to UMOEAs for 21, 5 and 4, respectively, to AMALGAM-SO for 27, 2 and 1, respectively, and to LSAOSDE for 23, 3 and 4, respectively.

Also, to identify the statistical differences among the algorithms, the Wilcoxon test was conducted, with the results recorded in Table 16 indicating that MARLwCMA was statistically better than all the others. Moreover, it was ranked first based on the results obtained from the Friedman test presented in Table 17. As a further comparison of the algorithms, a graph of their performance profiles is shown in Figure 5 in which it is clear that MARLwCMA attained a ratio of 1.0 first at $\tau \approx 91$.

F. TESTING MARLwCMA ON ADDITIONAL BENCHMARK PROBLEMS

This section presents the solutions obtained by MARLwCMA for two other sets of 10D, 30D and 50D benchmark problems taken from the CEC2013 and CEC2015 competitions. Its performances were judged against those of many well-known algorithms, as previously stated in Section IV. The comparisons were conducted based on the quality of the obtained solutions, Friedman test, Wilcoxon test and the performance profiles graphs.

1) TESTING MARLwCMA ON CEC2013 PROBLEMS

This benchmark data set has 28 problems in the following three categories: $F_{01} - F_{05}$ are unimodal functions; $F_{06} - F_{20}$ basic multimodal ones; and$F_{21} - F_{28}$ composite ones. Tables 12, 13 and 14 in the supplementary material present the average and standard deviation values obtained from MARLwCMA and the other algorithms for 10D, 30D and 50D problems, respectively. Note that the results of the compared algorithms are taken from their original sources.

For 10D test functions, it can be seen that the proposed MARLwCMA obtains the best results in 18 test functions, while SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, and MDE-pBX obtain the best results for 7, 9, 4, 2, 5, 5, 3, 6, 6 and 2 test functions, respectively. Considering 30D test problems, the proposed MARLwCMA achieves the best results in 22 test functions, while the best number of solutions obtained by SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, and MDE-pBX are 2, 6, 3, 3, 4, 4, 3, 2, 5 and 2 test functions, respectively. Considering the 50D test functions, MARLwCMA obtains the best results for 18 test functions. While SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, and MDE-pBX attain the best results for 5, 6, 5, 3, 4, 4, 2, 2, 5 and 2 test functions, respectively.

As for 10D unimodal test problems ($F01$ and $F05$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA obtain the best results on 2, 4, 2, 2, 3, 2, 2, 5, 4, 2 and 4 test functions respectively. For the 10D basic multimodal test functions ($F06$ to $F20$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA achieve the best results on 3, 1, 2, 0, 2, 3, 1, 1, 2, 0 and 11 test functions, respectively. Considering the composition functions ($F21$ to $F28$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA are able to achieve the best results on 2, 4, 0, 0, 0, 0, 0, 0, 0, 0 and 3 test functions, respectively.

Concerning 30D unimodal test functions ($F01$ to $F05$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA obtain the best results on 2, 4, 2, 2, 2, 2, 2, 2, 2, 2 and 3 test functions, respectively. For 30D basic multimodal test functions ($F06$ to $F20$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA are able to attain the best results on 0, 0, 1, 1, 2, 2, 1, 0, 3, 0 and 13 test functions, respectively. Regarding 30D composition functions ($F21$ to $F28$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA achieve the best results on 0, 2, 0, 0, 0, 0, 0, 0, 0, 0 and 6 test functions, respectively.

Regarding to 50D unimodal test functions ($F01$ to $F05$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA are able to

TABLE 19. Average rankings of all algorithms for all dimensions of CEC2013 benchmark problems obtained by Friedman test.

Algorithms	Rank for 10D	Rank for 30D	Rank for 50D
SinDE	5.68	5.52	4.93
CMA-ES-RIS	6.23	6.68	6.21
IDE	4.16	4.63	4.55
. CoDE	8.77	5.38	6.14
EPSDE	5.71	6.84	7.00
. jDE	6.30	6.68	6.84
SaDE	5.43	6.71	7.02
DE-APC	8.20	8.13	7.89
SHADE	4.48	4.29	4.45
MDE-pBX	8.02	8.79	8.39
MARLwCMA	**3.02**	**2.38**	**2.57**

attain the best results on 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2 and 3 test functions, respectively. Concerning 30D basic multimodal test functions ($F06$ to $F20$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA obtain the best results on 1, 1, 2, 0, 1, 1, 0, 0, 3, 0 and 10 test functions, respectively. For 30D composition functions ($F21$ to $F28$), SinDE, CMA-ES-RIS, IDE, CoDE, EPSDE, jDE, SaDE, DE-APC, SHADE, MDE-pBX and MARLwCMA achieve the best results on 2, 2, 1, 1, 1, 1, 0, 0, 1, 0 and 5 test functions, respectively. Obviously, it can be concluded from the above analysis, the proposed algorithm is Superior to other algorithms on basic multimodal and composition functions, while it comes in the second position when it used to solve unimodal test functions.

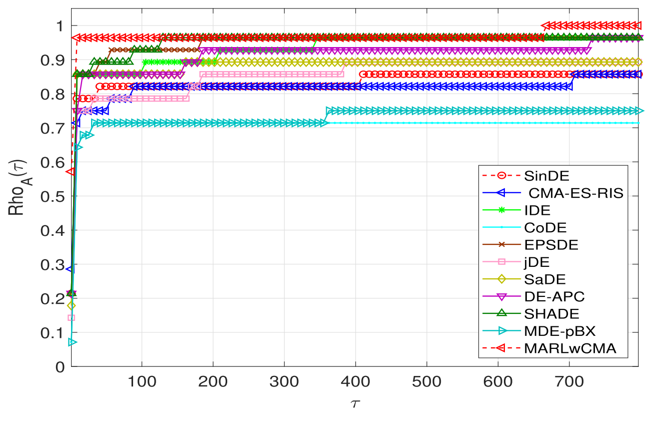
A summary of the comparisons based on the Wilcoxon test is presented in Table 18 from which it can be concluded that MARLwCMA achieved the best results for most of those problems and was also statistically better than all the other algorithms for all dimensions. Further analysis was conducted using the Friedman test, with the algorithms' mean rankings presented in Table 19. It is evident that the proposed MARLwCMA ranked first for all dimensions.

Considering the performance profiles tool, it is clear that MARLwCMA was better than all the rival algorithms, as depicted in Figure 6. It was confirmed that it reached the maximum probability at the beginning and attained probability values of 1.0 at $\tau \approx 190$, 85 and 325 for the 10D, 30D and 50D test problems, respectively.

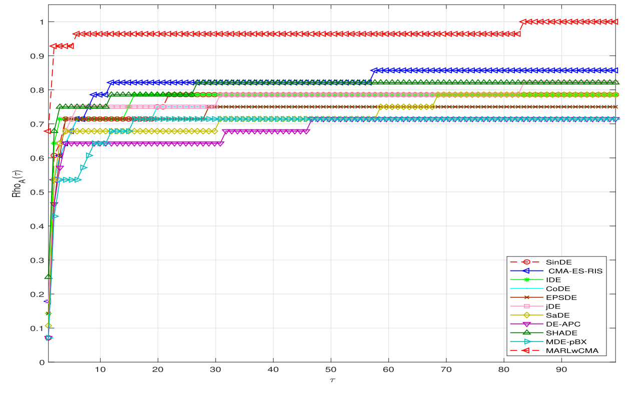
2) TESTING MARLwCMA ON CEC2015 PROBLEMS

Similar to the CEC2014 problems, the CEC2015 ones could be classified in four categories: 1) $F01 - F_{02}$ are unimodal; 2) $F_{03} - F_{05}$ simple multimodal; 3) $F_{06} - F_{08}$ hybrid; and 4) $F_{09} - F_{15}$ composite.

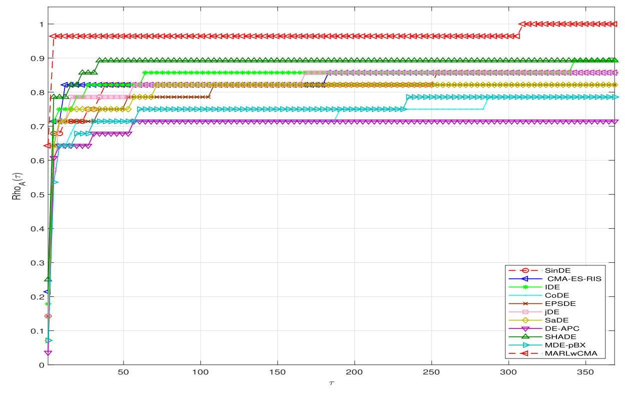
Tables 15, 16 and 17 in the supplementary material present the average and standard deviation results obtained from the proposed MARLwCMA for the 10D, 30D and 50D test functions, respectively. For 10D test functions, it can be seen that LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO, LSAOSDE and MARLwCMA are able to obtain the best results for 8, 6, 6, 7, 4, 4, 6, 6, 4, 6 and 7 test functions, respectively. In regards to 30D test



(a) 10D test problems



(b) 30D Test Problems



(c) 50D test Problems

FIGURE 6. Performance profiles of MARLwCMA and the rival algorithms algorithms based on CEC2013 unconstrained problems for (a)10D problems; (b) 30D problems; and (c) 50D problems.

problems MARLwCMA is able to obtain the best results for 8 test functions, while LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO and LSAOSDE are able to obtain the best results for 4, 2, 2, 2, 2, 2, 4, 4, 5, 6 and 8 test functions, respectively. Concerning the 50D test functions, LSHADE, SHADE, JADE, CoDE, SaDE,

TABLE 20. Summary of comparisons of MARLwCMA against LSHADE, SHADE, JADE, CoDE, SaDE, EPSDE, MPEDE, UMOEAs, AMALGAM-SO and LSAOSDE for 10D, 30D and 50D test problems taken from CEC2015 competition.

Algorithms	10D				30D				50D			
	Better	Similar	Worse	P-value, Dec.	Better	Similar	Worse	P-value, Dec.	Better	Similar	Worse	P-value, Dec.
MARLwCMA vs. LSHADE	5	4	6	(0.594,≈)	8	2	5	(0.196,≈)	10	2	3	(0.013,+)
MARLwCMA vs. SHADE	7	6	2	(0.110,≈)	12	2	1	(0.002,+)	12	2	1	(0.002,+)
MARLwCMA vs. JADE	7	6	2	(0.038,+)	12	2	1	(0.002,+)	12	2	1	(0.002,+)
MARLwCMA vs. CoDE	6	6	3	(0.139,≈)	11	2	2	(0.004,+)	11	3	1	(0.004,+)
MARLwCMA vs. SaDE	7	5	3	(0.285,≈)	13	1	1	(0.004,+)	14	1	0	(0.001,+)
MARLwCMA vs. EPSDE	9	4	2	(0.026,+)	11	2	2	(0.023,+)	10	2	3	(0.039,+)
MARLwCMA vs. MPEDE	4	7	4	(0.889,≈)	8	2	5	(0.294,≈)	12	2	1	(0.002,+)
MARLwCMA vs UMOEAs	8	7	0	(0.012,+)	10	3	2	(0.009,+)	10	3	2	(0.008,+)
MARLwCMA vs. AMALGAM-SO	11	3	1	(0.008,+)	10	2	3	(0.032,+)	10	2	3	(0.028,+)
MARLwCMA vs. LSAOSDE	4	7	4	(1.000,≈)	6	4	5	(0.695,≈)	9	4	2	(0.043,+)

EPSDE, MPEDE, UMOEAs, AMALGAM-SO, LSAOSDE and MARLwCMA achieve the best results for 3, 2, 2, 1, 1, 4, 2, 3, 3, 4 and 8 test functions, respectively.

Based on the number of best results, it can be concluded that, MARLwCMA performs well in all dimensions, especially for problems with 30D and 50D. While LSHADE, SHADE, JADE, CoDE, MPEDE, UMOEAs and LSAOSDE perform well in problems with 10D, but their performance deteriorate for the higher dimensions.

Regarding the quality of solutions, Table 20 provides a summary of the results obtained by the proposed MARLwCMA which shows that it outperformed all the other algorithms considered with respect to the number of better solutions. Based on the Wilcoxon test results, it was better than most of the other algorithms for the 10D and 30D test problems and superior to all of them for the 50D ones.

Regarding the attained solutions' quality, Table 20 presents the summary of the results obtained by the proposed MARLwCMA. The results showed that MARLwCMA outperformed all other algorithms considered in the comparison with respect to the number of better solutions obtained. Considering the Wilcoxon test, the proposed MARLwCMA was statistically better than most of the other algorithms in 10D and 30D, while it is superior to all other algorithms for 50D test problems.

Considering the Friedman test results, MARLwCMA was ranked first for the 30D and 50D test problems and second for the 10D ones, slightly behind the first-ranked algorithm (LSAOSDE).

Finally, it is clear from from the performance profiles graphs that MARLwCMA was better than all the other algorithms, as depicted in Figure 7. It was confirmed that it achieved the maximum probability at the start and reached probability values of 1.0 at $\tau \approx 17$, 4.5 and 7.5 for the 10D, 30D and 50D test functions, respectively.

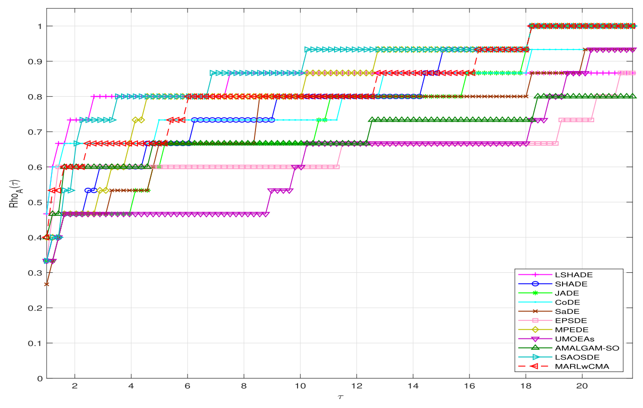
G. TESTING MARLwCMA ON HIGHER DIMENSIONS

In this section, the performance of the proposed MARLwCMA is tested by solving optimization problems 100D including, CEC2014 and CEC2017.

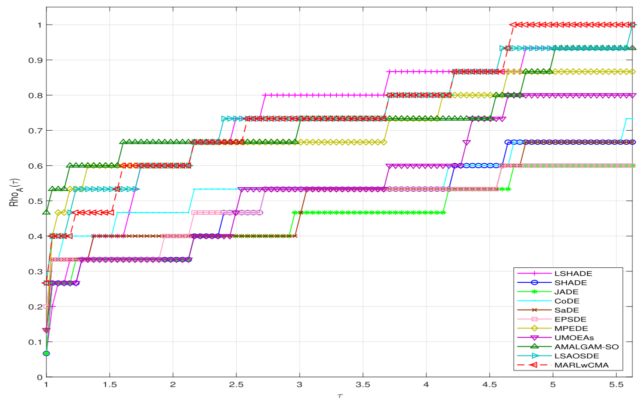
For the CEC2014 test problems, the proposed MARLwCMA is compared with the following algorithms:

- Self-adaptive of DE control parameters (jDE) [73];
- Success History based DE (SHADE) [23];
- Success History based DE with linear population size reduction (LSHADE) [24];
- DE with composite trial vector generation strategies (CoDE) [41];
- DE with an ensemble of parameters and mutation strategies (EPSDE) [42];
- Improved variant of the iL-SHADE algorithm (jSO) [26];
- Adap-tive DE with optional external archive with Auto Enhanced Population Diversity (AEPD-JADE) [81];
- Self-adaptive Differential Evolution with crossover neighborhood (DE-VNS) [82];
- A sinusoidal differential evolution algorithm (SinDE) [74];
- DE with an individual-dependent mechanism (IDE) [76];
- Enhanced fitness-adaptive differential evolution algorithm with novel mutation with elites regeneration (ERG-EFADE) [83];
- Improved L-SHADE algorithm (iL-SHADE) [25];
- L-SHADE with Competing Strategies (LSHADE44) [84];
- Multi-population ensemble DE (MPEDE) [57]; and
- Ensemble of multiple DE variants (EDEV) [38].

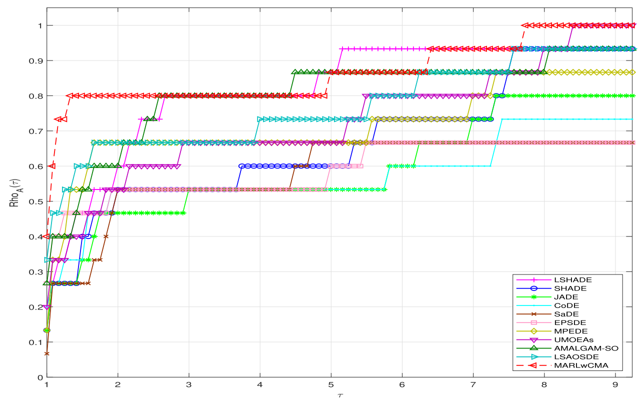
The mean of the error obtained by MARLwCMA, DE-VNS, APED-JADE, sinDE, MPEDE, EDEV, iLSHADE, LSHADE44, and LSHADE is presented in Tables 18 and 19 in the supplementary material file. From this table, it can be seen that, DE-VNS, APED-JADE, sinDE, MPEDE, EDEV, iLSHADE, LSHADE44, LSHADE and MARLwCMA obtain the best results for 0, 5, 4, 4, 4, 5, 1, 7 and 12 test functions, respectively. Concerning the unimodal functions ($F01$ to $F03$), MARLwCMA, DE-VNS, APED-JADE, sinDE, MPEDE, EDEV, iLSHADE, LSHADE44, and LSHADE obtain the best results for 0, 0, 0, 1, 1, 2, 0, 1 and 2 test functions, respectively. This confirms the superiority of the proposed algorithm for solving unimodal functions. For the simple multimodal functions ($F04$ to $F16$), the proposed MARLwCMA achieves the best results for 2 test functions, while DE-VNS, APED-JADE, sinDE, MPEDE, EDEV, iLSHADE, LSHADE44, and LSHADE obtain the best results for 0, 4, 2, 2, 1, 2, 1 and 6. It can be concluded that, LSHADE and APED-JADE are better than MARLwCMA when solving multimodal functions. In regards to



(a) 10D test problems



(b) 30D Test Problems



(c) 50D test Problems

FIGURE 7. **Performance profiles of MARLwCMA and the rival algorithms algorithms based on CEC2015 unconstrained problems for (a)10D problems; (b) 30D problems; and (c) 50D problems.**

hybrid functions ($F17$ and $F22$) and composition functions ($F23$ to $F30$), the proposed MARLwCMA attains the best results for 8 test function, while DE-VNS, APED-JADE, sinDE, MPEDE, EDEV, iLSHADE, LSHADE44, and LSHADE achieve the best results for 0, 1, 2, 1, 2, 1, 0, and 0. Obviously, from the above analysis, the proposed MARLwCMA is suitable to solve unimodal, hybrid and composition functions, while it performs poorly for the multimodal test functions.

TABLE 21. **Average rankings of all algorithms for all dimensions of CEC2015 benchmark problems obtained by Friedman.**

Algorithms	Rank for 10D	Rank for 30D	Rank for 50D
LSHADE	5.00	5.37	4.03
SHADE	5.80	6.70	6.83
JADE	7.00	7.70	7.50
CoDE	5.70	6.33	7.23
SaDE	6.90	7.60	7.73
EPSDE	7.20	8.07	7.83
MPEDE	5.10	4.67	5.67
UMOEAs	6.90	6.33	6.40
AMALGAM-SO	7.53	5.70	5.30
LSAOSDE	**4.23**	4.00	4.53
MARLwCMA	4.63	**3.53**	**2.93**

Considering the quality of solutions, Table 22 presents a summary of the results produced by the proposed MARLwCMA and the rival algorithms. It is clear from that table that the proposed algorithm outperformed all the competing algorithms. The proposed MARLwCMA is better than DE-VNS, AEPD-JADE, sinDE, MPEDE, EDEV, ILSHADE, LSHADE44, LSHADE, CoDE, EPSDE, jSO, jDE, IDE ERG-EFADE and SHADE for 24, 19, 18, 19, 17, 17, 21, 19, 29, 28, 18, 21, 22, 26 and 20 test problems, respectively, worse than them in 6, 10, 11, 8, 8, 9, 8, 10, 1, 1, 10, 9, 7, 4, and 9 test functions respectively.

In regards to the Wilcoxon test, the proposed algorithm is statistically better than DE-VNS, sinDE, MPEDE, LSHADE, LSHADE44, CoDE, EPSDE, jDE, IDE, ERG-EFADE and SHADE, while there is no siginicance difference with AEPD-JADE, EDEV, iL-SHADE and jSO. However, there is a bias toward the proposed algorithm (MARLwCMA) as it obtains better results than them in most of the test problems. As a further analysis, the Friedman test is conducted to rank all algorithms with the obtained results presented in Table 23. From Table 23, it is clear that the proposed MARLwCMA is ranked first followed by ILSHADE while CoDE comes on the last position.

For CEC2017 test problems, the performance of the proposed MARLwCMA has been tested and compared with the performance of the following algorithms:

- Effective butterfly optimizer with co-variance matrix adapted restart phase (EBOwCMAR) winner of CEC2017 competition [85];
- Improved DE with population size adaptation (IDEwPSA) [86];
- Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood (LSHADE-cnEpSin) [27];
- LSHADE with rank-based selective pressure algorithm (LSHADE-RSP) [30];
- LSHADE algorithm with an alternative adaptation approach for the selection of control parameters (LSHADE-SPA) [29];

TABLE 22. Summary of comparisons of MARLwCMA against DE-VNS, AEPD-JADE, sinDE, MPEDE, EDEV, ILSHADE, LSHADE44, LSHADE, CoDE, EPSDE, jSO, jDE, IDE, ERG-EFADE, and SHADE for 100D test problems taken from CEC2014 competition.

Algorithms	Better	Equal	Worse	P-value, Dec.
MARLwCMA vs. DE-VNS	24	0	6	0.003, +
MARLwCMA vs. AEPD-JADE	19	1	10	0.098, ≈
MARLwCMA vs. sinDE	18	1	11	0.045, +
MARLwCMA vs. MPEDE	19	3	8	0.024, +
MARLwCMA vs. EDEV	17	5	8	0.166, ≈
MARLwCMA vs. ILSHADE	17	4	9	0.112, ≈
MARLwCMA vs. LSHADE44	21	1	8	0.006, +
MARLwCMA vs. LSHADE	19	1	10	0.011, +
MARLwCMA vs. CoDE	29	0	1	0.000, +
MARLwCMA vs. EPSDE	28	1	1	0.000, +
MARLwCMA vs. jSO	18	2	10	0.339, ≈
MARLwCMA vs. jDE	21	0	9	0.004, +
MARLwCMA vs. IDE	22	1	7	0.001, +
MARLwCMA vs. ERG-EFADE	26	0	4	0.000, +
MARLwCMA vs. SHADE	20	1	9	0.012, +

TABLE 23. Average ranking of all algorithms for 100D of CEC2014 benchmark problems obtained by Friedman Test.

Algorithm	Rank
MARLwCMA	**5.05**
DE-VNS	9.32
AEPD-JADE	6.75
sinDE	7.38
MPEDE	7.77
EDEV	6.25
ILSHADE	5.93
LSHADE44	7.85
LSHADE	7.00
CoDE	14.95
EPSDE	12.48
jSO	6.28
jDE	9.17
IDE	9.42
ERG-EFADE	10.85
SHADE	9.55

- Multi-method based orthogonal experimental design algorithm (MM-OED) [9];
- SHADE with Distance based parameter adaptation (DISH) [87];
- Improved variant of the iL-SHADE algorithm (jSO) [26];
- An enhanced version of IPOP-CMA-ES (RB-IPOP-CMA-ES) [88];
- Parameter Adaptive Differential Evolution With Novel Parameter Control (PaDE-NPC) [89];
- Tow phase DE (TPDE) [90];

In CEC2017, there are 29 test problems, which they used to test the performance of the proposed algorithm. The average of the error and the standard deviations are presented in Table 20 in the supplementary material file. From this table, EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, TPDE and MARLwCMA achieve the best results on 2, 0, 7, 4, 4, 5, 3, 1, 3, 2, 6 and 10 test problems, respectively. Considering the unimodal ($F01$ and $F03$) and simple multimodal functions ($F04$ to $F10$), MARLwCMA achieves the best results for 6 test functions, while EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, TPDE obtain the best results on 1, 3, 4, 3, 2, 1, 1, 2 and 1 test functions, respectively. Considering the hybrid functions ($F11$ to $F20$) and composition functions ($F21$ to $F30$), the proposed MARLwCMA algorithm achieve the best results for 4 test functions, while EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, TPDE attain the best results for 1, 0, 4, 0, 1, 3, 1, 0, 1, 0, 4 test functions, respectively. It is obvious that, MARLwCMA, LSHADE-cnEpSin and TPDE have the same performance when solving the hybrid and composition functions, as the number of best solutions obtained from each of these algorithms are equal.

Table 24 presents the summary of the results obtained from MARLwCMA, EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, and TPDE. Considering the quality of solutions, it is clear from that table that the proposed algorithm outperformed all the competing algorithms. The proposed MARLwCMA is better than EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, and TPDE for 21, 29, 15, 15, 16, 22, 16, 18, 19, 22 and 17 test problems, respectively. While it is worse than them in 7, 0, 12, 13, 12, 7, 13, 11, 8, 7 and 12 test functions respectively. The proposed MARLwCMA obtains similar results to EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, and TPDE for 1, 0, 2, 1, 1, 0, 0, 0, 1, 0 and 0 test problems, respectively.

Considering the Wilcoxon test, the proposed MARLwCMA algorithm is statistically better than EBOwCMAR, IDEwPSA, MM-OED, jSO, RB-IPOP-CMA-ES and PaDE-NPC, while there is no significance difference with LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, DISH and TPDE. However, there is a bias toward the proposed algorithm (MARLwCMA) as it obtains better results than them in most of the test problems.

As a further analysis, the Friedman test is conducted to rank all algorithms with the obtained results presented in Table 25. From Table 25, it is clear that the proposed MARLwCMA is ranked first followed by DISH while IDEwPSA comes on the last position.

H. TESTING PERFORMANCE OF MARLwCMA ON REAL-WORLD APPLICATIONS

In this section, the performance of the proposed MARLwCMA is further examined by solving a benchmark test set that has 22 real-application problems taken from CEC2011. The proposed algorithm was run following the guideline of the competition that required 25 independent runs for each

TABLE 24. Summary of comparisons of MARLwCMA against EBOwCMAR, IDEwPSA, LSHADE-cnEpSin, LSHADE-RSP, LSHADE-SPA, MM-OED, DISH, jSO, RB-IPOP-CMA-ES, PaDE-NPC, and TPDE for 100D test problems taken from CEC2017 competition.

Algorithms	Better	Equal	Worse	P-value, Dec.
MARLwCMA vs. EBOwCMAR	21	1	7	0.036, +
MARLwCMA vs. IDEwPSA	29	0	0	0.000, +
MARLwCMA vs. LSHADE-cnEpSin	15	2	12	0.337, ≈
MARLwCMA vs. LSHADE-RSP	15	1	13	0.345, ≈
MARLwCMA vs. LSHADE-SPA	16	1	12	0.255, ≈
MARLwCMA vs. MM-OED	22	0	7	0.015, +
MARLwCMA vs. DISH	16	0	13	0.325, ≈
MARLwCMA vs. jSO	18	0	11	0.042, +
MARLwCMA vs. RB-IPOP-CMA-ES	19	1	8	0.005, +
MARLwCMA vs. PaDE-NPC	22	0	7	0.009, +
MARLwCMA vs. TPDE	17	0	12	0.304, ≈

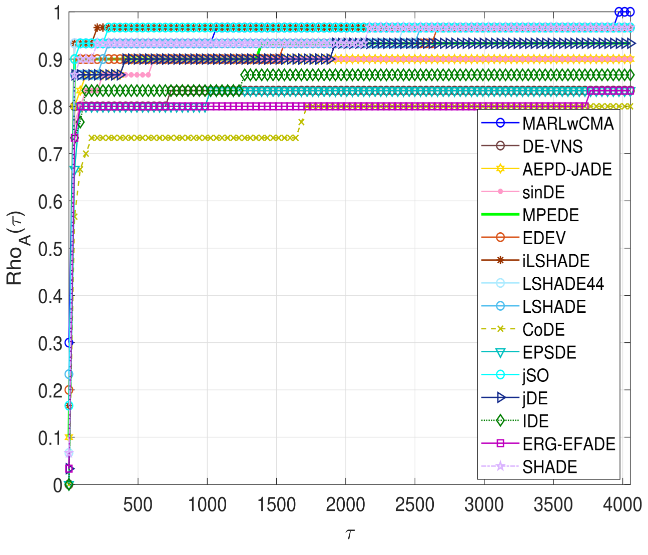
TABLE 25. Average ranking of all algorithms for 100D of CEC2017 benchmark problems obtained by Friedman Test.

Algorithm	Rank
MARLwCMA	**4.71**
EBOwCMAR	6.30
IDEwPSA	11.43
LSHADE-cnEpSin	5.64
LSHADE-RSP	5.66
LSHADE-SPA	5.86
MM-OED	6.27
DISH	5.14
jSO	6.96
RB-IPOP-CMA-ES	6.95
PaDE-NPC	7.59
TPDE	5.48

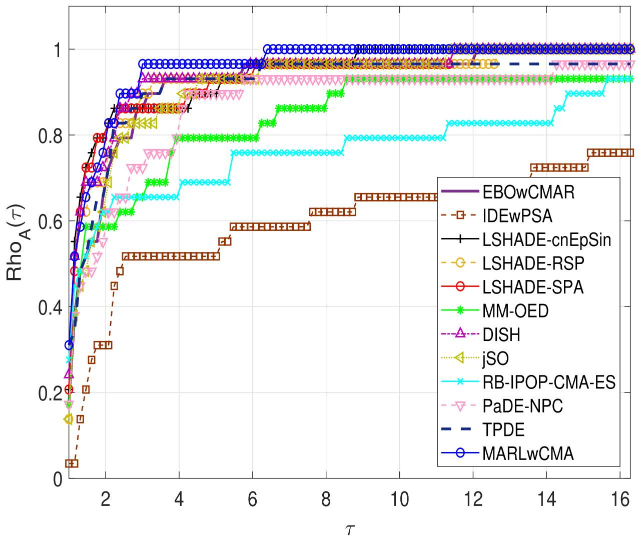
problem with a maximum number of fitness evaluations equal to 150000.

The performance of the proposed MARLwCMA algorithm is compared with the following algorithms:

- ADDE: An adaptive distributed DE algorithm [91];
- CDASA: A continuous DE ant-stigmergy algorithm [];
- CloudeDE: A distributed Cloude DE [92];
- AsAMP-dDE: An asynchronous adaptive multi-population model for distributed DE [93];
- DDE-SD: A distributed DE with space driven topology [94];
- DDEM-RCA: A distributed DE with migration strategy [95];
- CoDE: DE with composite trial vector generation strategies [41];
- SaDE: DE algorithm with an adaptation strategy [72];
- jDE: A self-adaptive of DE control parameters [73];
- DE-DPS: DE with dynamic parameters selection [96];
- JADE: An adaptive DE with optional external archive [22].
- JADE-sort: An adaptive DE with sorting crossover rate [97];
- SHADE: A success History based DE [23]
- EPSDE: DE with an ensemble of parameters and mutation strategies [42];
- IDE: DE with an individual-dependent mechanism [76]; and



(a) 100D test problems CEC2014



(b) 100D test problems CEC2017

FIGURE 8. Performance profiles of MARLwCMA and the rival algorithms algorithms for 100D problems for (a) CEC2014 and (b) CEC2017.

- jSO: an improved variant of the iL-SHADE algorithm [26];

Tables 21 and 22 in the supplementary file presents the results obtained from the proposed MARLwCMA and the competing algorithms. Concerning the quality of the results, Table 26 presents the summary of the results. It is clear from Table 26 that the proposed MARLwCMA obtain better results than ADDE, IBDDE, CloudeDE, AsAMP-bDE, DDE-SD, DDEM-RCA, CoDE, SaDE, jDE, DE-DPS, JADE, JADE-sort, SHADE, EPSDE, IDE and jSO for 13, 19, 18, 18, 15, 18, 16, 17, 17, 18, 15, 15, 14, 15, 16 and 15 test problems, respectively, it obtains worse results than them for 5, 1, 2, 2, 3, 2, 4, 3, 3, 2, 5, 5, 5. 4, 4, and 4 test problems, respectively.

TABLE 26. Summary of comparisons of MARLwCMA against ADDE, IBDDE, CloudeDE, AsAMP-BDE,DDE-SD, DDEM-RCA, CoDE, SaDE, jDE. DE-DPS, JADE, JADE-sort, SHADE, EPSDE, IDE and jSO for test problems taken from the CEC2011 competition.

Algorithms	Better	Equal	Worse	P-value, Dec.
MARLwCMA vs. ADDE	13	4	5	0.078, ≈
MARLwCMA vs. IBDDE	19	2	1	0.000, +
MARLwCMA vs. CloudDE	18	2	2	0.001, +
MARLwCMA vs. AsAMP-bDE	18	2	2	0.001, +
MARLwCMA vs. DDE-SD	15	4	3	0.004, +
MARLwCMA vs. DDEM-RCA	18	2	2	0.001, +
MARLwCMA vs. CoDE	16	2	4	0.004, +
MARLwCMA vs. SaDE	17	2	3	0.001, +
MARLwCMA vs. jDE	17	2	3	0.002, +
MARLwCMA vs. DE-DPS	18	2	2	0.001, +
MARLwCMA vs. JADE	15	2	5	0.008, +
MARLwCMA vs. JADE-sort	15	2	5	0.021, +
MARLwCMA vs. SHADE	14	3	5	0.024, +
MARLwCMA vs. EPSDE	15	3	4	0.008, +
MARLwCMA vs. IDE	16	2	4	0.011, +
MARLwCMA vs. jSO	15	3	4	0.008, +

TABLE 27. Average ranking of all algorithms for of CEC2011 benchmark problems obtained by Friedman Test.

Algorithm	Rank
MARLwCMA	**3.95**
ADDE	4.43
IBDDE	15.00
CloudDE	10.77
AsAMP-bDE	12.43
DDE-SD	8.64
DDEM-RCA	13.73
CoDE	7.25
SaDE	9.68
jDE	9.91
DE-DPS	12.68
JADE	9.80
JADE-sort	6.80
SHADE	6.55
EPSDE	9.14
IDE	6.82
jSO	4.93

It can be concluded that the proposed MARLwCMA is superior to the competing algorithms.

From the Wilcoxon test, as it is clear from Table 26, the proposed MARLwCMA is statistically superior to all the competing algorithms. While there is no statistical difference between the proposed algorithm MARLWCMA and ADDE for significance level=0.05, however, there is a significance difference when the significance level=0.10.

As a further analysis, the Friedman test is conducted in order to rank all the competing algorithms, with the results depicted in Table 27. From this table, the proposed MARLwCMA is ranked first, followed by ADDE. While DE-DPS performs the worst.

I. ALGORITHM COMPLEXITY

In this section, the complexity of the proposed MARLwCMA algorithm was computed following the competitions guideline in [18]–[20]. Table 28 presents the time complexity of the proposed MARLwCMA calculated for 10D, 30D, 50D and 100D in CEC2013 [18], CEC2014 [19], CEC2015 [20] and CEC2017. As defined in [18]–[20], T_0 is the time calculated when running the code in Algorithm 5.

TABLE 28. Algorithm complexity.

Benchmark	D	T_0	T_1	$\hat{T_2}$	$\frac{(\hat{T_2}-T_1)}{T_0}$
CEC2013	$D=10$	0.14379	0.41211	1.43070	7.08398
	$D=30$		1.32658	3.33580	13.97350
	$D=50$		2.23885	5.07180	19.70228
	$D=100$		4.38432	9.24170	33.78151
CEC2014	$D=10$	0.14379	0.13433	2.86090	18.96246
	$D=30$		0.42763	3.74530	23.07332
	$D=50$		0.99038	5.15310	28.95041
	$D=100$		3.71774	9.39430	39.47868
CEC2015	$D=10$	0.14379	0.22904	0.25491	0.17995
	$D=30$		0.95557	1.47280	3.59716
	$D=50$		1.93863	3.62640	11.73793
	$D=100$		5.81775	8.96860	21.91320

Algorithm 5 Algorithm Complexity

```
1: for $i = 1 : 1000000$ do
2:     $x = x + x; x = \sqrt(x);$
3:     $x = log(x);$
4:     $x = \exp(x);$
5:     $x = x/(x + 2);$
6: end for
```

T_1 is the time of executing 200,000 evaluations of the benchmark functions $F14$, $F18$, $F1$ and $F18$ for CEC2013, CEC2014, CEC2015 and CEC2017, respectively. T_2 is the time to run the proposed MARLwCMA for 200000 function evaluations for the selected functions in D dimensions, $\hat{T_2}$ is the average of T_2 values of 5 runs. It can be concluded from Table 28 that, the computational time is reasonably small and linearly increases as the problem dimensions increase.

V. CONCLUSION AND FUTURE WORK

While several EAs have been developed to solve optimization problems, the literature reveals that no single search operator and/or algorithm can successfully solve all forms of optimization problems. As a consequence, several multi-operator and/or multi-method-based algorithms have been introduced. Their structures were largely based on trial and error techniques and, furthermore, their efficiency could be statistically surpassed by single-operator algorithms. As a result, we proposed a new multi-method framework including a multi-operator DE and CMA-ES, with the SQP local search algorithm applied in the later generations of the optimization process. In the multi-operator DE, a RL technique was used to select the most suitable mutation operator based on the diversity and quality of solutions. The performance of the proposed algorithm was judged by solving 73 bound-constrained optimization problems with 10, 30 and 50 variables. It also used to solve optimization problems with higher dimensions (100D) taken from CEC2014 and CEC2017. The performance of the proposed algorithm is further tested by solving real-world application problems taken from CEC2011 benchmark problems.

The computational results and statistical analysis revealed that MARLwCMA was statistically superior to its constituent algorithms for the 10D, 30D and 50D problems considered. It was also compared with state-of-the-art algorithms and demonstrated its superiority.

It is concluded that the proposed MARLwCMA was able to produce better results than other algorithms, as it was able to effectively place emphasis on the well-performing operator during the optimization process. This effective selection of operators (refer to Figure 2) was due to the proper utilization of reinforcement learning. Further, the right selection of operators and algorithms to be involved in the framework provided complementary search capabilities. This, in turn, increased MARLwCMA's ability to obtain better solutions for different problem types.

On the other hand, one limitation of the proposed approach was the number of parameters that required tuning. This, in fact, opens a new future research direction; that is, how to propose an adaptive mechanism that adapts these parameters during the optimization process. Other possible future research directions could be: (1) adapting RL to select the best-performing algorithm among several under a single-algorithm framework; (2) adopting the algorithm to solve other types of test problems, i.e., constrained and/or integer; and (3) evaluation the scalability of the proposed approach.

REFERENCES

[1] S. Elsayed, R. Sarker, and C. A. Coello Coello, "Fuzzy rule-based design of evolutionary algorithm for optimization," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 301–314, Jan. 2019.

[2] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Improved united multi-operator algorithm for solving optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.

[3] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[4] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-based adaptive operator selection mechanism for differential evolution," *Inf. Sci.*, vols. 418–419, pp. 383–404, Dec. 2017.

[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 Comput.*, vol. 22, no. 19, pp. 6595–6616, Oct. 2018.

[6] N. Hansen, "Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed," in *Proc. 11th Annu. Conf. Companion Genetic Evol. Comput. Conf. GECCO*, 2009, pp. 2389–2396.

[7] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Two-phase differential evolution framework for solving optimization problems," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.

[8] X. Yu, C. Li, and J. Zhou, "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios," *Knowl.-Based Syst.*, vol. 204, Sep. 2020, Art. no. 106209.

[9] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-method based orthogonal experimental design algorithm for solving CEC2017 competition problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1350–1357.

[10] Y. C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *J. Optim. Theory Appl.*, vol. 115, no. 3, pp. 549–570, Dec. 2002.

[11] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 3, pp. 385–398, Mar. 2015.

[12] Y. Hu, Y. Yao, and W. S. Lee, "A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs," *Knowl.-Based Syst.*, vol. 204, Sep. 2020, Art. no. 106244.

[13] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 2, no. 4. Cambridge, MA, USA: MIT Press, 1998.

[14] E. Emary, H. M. Zawbaa, and C. Grosan, "Experienced gray wolf optimization through reinforcement learning and neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 681–694, Mar. 2018.

[15] A. K. Sadhu, A. Konar, T. Bhattacharjee, and S. Das, "Synergism of firefly algorithm and Q-Learning for robot arm path planning," *Swarm Evol. Comput.*, vol. 43, pp. 50–68, Dec. 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 Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 825–837, Apr. 2019.

[17] A. Rost, I. Petrova, and A. Buzdalova, "Adaptive parameter selection in evolutionary algorithms by reinforcement learning with dynamic discretization of parameter range," 2016, *arXiv:1603.06788*. [Online]. Available: http://arxiv.org/abs/1603.06788

[18] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China and Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, 2013, pp. 281–295, no. 34.

[19] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou China, Nanyang Technol. Univ., Singapore, Tech. Rep., 635, 2013.

[20] 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," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou China, Nanyang Technol. Univ., Singapore, Tech. Rep. 201411A, 2014, pp. 625–640, vol. 29.

[21] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, Jun. 2005.

[22] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[23] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1952–1959.

[24] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.

[25] J. Brest, M. S. Maucec, and B. Boskovic, "IL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1188–1195.

[26] J. Brest, M. S. Maucec, and B. Boskovic, "Single objective real-parameter optimization: Algorithm jSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1311–1318.

[27] 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 *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 372–379.

[28] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction," *Swarm Evol. Comput.*, vol. 39, pp. 141–156, Apr. 2018.

[29] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 145–152.

[30] V. Stanovov, S. Akhmedova, and E. Semenkin, "LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.

[31] Z. Meng, J.-S. Pan, and K.-K. Tseng, "PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization," *Knowl.-Based Syst.*, vol. 168, pp. 80–99, Mar. 2019.

[32] Z. Meng, Y. Chen, and X. Li, "Enhancing differential evolution with novel parameter control," *IEEE Access*, vol. 8, pp. 51145–51167, 2020.

[33] S. Elsayed, "Evolutionary approach for constrained optimization," School Eng. Inf. Technol., Univ. New South Wales, Canberra, ACT, Australia, 2012.

[34] 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 Syst. Appl.*, vol. 162, Dec. 2020, Art. no. 113033.
[35] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.
[36] A. W. Mohamed, A. K. Mohamed, E. Z. Elfeky, and M. Saleh, "Enhanced directed differential evolution algorithm for solving constrained engineering optimization problems," *Int. J. Appl. Metaheuristic Comput.*, vol. 10, no. 1, pp. 1–28, Jan. 2019.
[37] Q. Fan, X. Yan, and Y. Zhang, "Auto-selection mechanism of differential evolution algorithm variants and its application," *Eur. J. Oper. Res.*, vol. 270, no. 2, pp. 636–653, Oct. 2018.
[38] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, and P. N. Suganthan, "Ensemble of differential evolution variants," *Inf. Sci.*, vol. 423, pp. 172–186, Jan. 2018.
[39] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms—A survey," *Swarm Evol. Comput.*, vol. 44, pp. 695–711, Feb. 2019.
[40] 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 *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1650–1657.
[41] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.
[42] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, Mar. 2011.
[43] 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," *Inf. Sci.*, vol. 482, pp. 86–104, May 2019.
[44] M. Attia, M. Arafa, E. Sallam, and M. Fahmy, "An enhanced differential evolution algorithm with multi-mutation strategies and self-adapting control parameters," *Int. J. Intell. Syst. Appl.*, vol. 11, no. 4, p. 26, 2019.
[45] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Multi-operator differential evolution algorithm for solving real-world constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.
[46] K. M. Sallam, R. K. Chakrabortty, and M. J. Ryan, "A two-stage multi-operator differential evolution algorithm for solving resource constrained project scheduling problems," *Future Gener. Comput. Syst.*, vol. 108, pp. 432–444, Jul. 2020.
[47] A. Santiago, B. Dorronsoro, A. J. Nebro, J. J. Durillo, O. Castillo, and H. J. Fraire, "A novel multi-objective evolutionary algorithm with fuzzy logic based adaptive selection of operators: FAME," *Inf. Sci.*, vol. 471, pp. 233–251, Jan. 2019.
[48] Q. Lin, Y. Ma, J. Chen, Q. Zhu, C. A. C. Coello, K.-C. Wong, and F. Chen, "An adaptive immune-inspired multi-objective algorithm with multiple differential evolution strategies," *Inf. Sci.*, vols. 430–431, pp. 46–64, Mar. 2018.
[49] J. Chen, J. Chen, and H. Min, "A united framework with multi-operator evolutionary algorithms and interior point method for efficient single objective optimisation problem solving," *Int. J. High Perform. Comput. Netw.*, vol. 13, no. 3, pp. 340–353, 2019.
[50] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. Coello Coello, and F. Herrera, "Bio-inspired computation: Where we stand and what's next," *Swarm Evol. Comput.*, vol. 48, pp. 220–250, Aug. 2019.
[51] J. Grobler, A. P. Engelbrecht, G. Kendall, and V. S. S. Yadavalli, "Multi-method algorithms: Investigating the entity-to-algorithm allocation problem," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 570–577.
[52] S. A. G. van der Stockt and A. P. Engelbrecht, "Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization," *Swarm Evol. Comput.*, vol. 43, pp. 127–146, Dec. 2018.
[53] J. Grobler, A. P. Engelbrecht, G. Kendall, and V. S. S. Yadavalli, "Heuristic space diversity control for improved meta-hyper-heuristic performance," *Inf. Sci.*, vol. 300, pp. 49–62, Apr. 2015.
[54] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 243–259, Apr. 2009.
[55] S. Elsayed, N. Hamza, and R. Sarker, "Testing united multi-operator evolutionary algorithms-II on single objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 2966–2973.
[56] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016.
[57] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, pp. 329–345, Feb. 2016.
[58] M. M. Drugan, "Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms," *Swarm Evol. Comput.*, vol. 44, pp. 228–246, Feb. 2019.
[59] P. Goyal, H. Malik, and R. Sharma, "Application of evolutionary reinforcement learning (ERL) approach in control domain: A review," in *Smart Innovations in Communication and Computational Sciences*. Springer, 2019, pp. 273–288.
[60] P. K. Das, H. S. Behera, and B. K. Panigrahi, "Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 1, pp. 651–669, Mar. 2016.
[61] C. Zhou, "Robot learning with GA-based fuzzy reinforcement learning agents," *Inf. Sci.*, vol. 145, nos. 1–2, pp. 45–68, Aug. 2002.
[62] F. Liu and G. Zeng, "Study of genetic algorithm with reinforcement learning to solve the TSP," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 6995–7001, Apr. 2009.
[63] M. M. Alipour, S. N. Razavi, M. R. F. Derakhshi, and M. A. Balafar, "A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem," *Neural Comput. Appl.*, vol. 30, no. 9, pp. 2935–2951, Nov. 2018.
[64] C.-F. Juang and T. B. Bui, "Reinforcement neural fuzzy surrogate-assisted multiobjective evolutionary fuzzy systems with robot learning control application," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 3, pp. 434–446, Mar. 2020.
[65] M. Sharma, A. Komninos, M. L. Ibanez, and D. Kazakov, "Deep reinforcement learning based parameter control in differential evolution," 2019, *arXiv:1905.08006*. [Online]. Available: http://arxiv.org/abs/1905.08006
[66] T. C. Bora, V. C. Mariani, and L. D. S. Coelho, "Multi-objective optimization of the environmental-economic dispatch with reinforcement learning based on non-dominated sorting genetic algorithm," *Appl. Thermal Eng.*, vol. 146, pp. 688–700, Jan. 2019.
[67] W. Ning, B. Guo, X. Guo, C. Li, and Y. Yan, "Reinforcement learning aided parameter control in multi-objective evolutionary algorithm based on decomposition," *Prog. Artif. Intell.*, vol. 7, no. 4, pp. 385–398, Dec. 2018.
[68] B. Cunha, A. M. Madureira, B. Fonseca, and D. Coelho, "Deep reinforcement learning as a job shop scheduling solver: A literature review," in *Proc. Int. Conf. Hybrid Intell. Syst.* Springer, 2018, pp. 350–359.
[69] A. G. Barto, "Reinforcement learning: Connections, surprises, and challenge," *AI Mag.*, vol. 40, no. 1, pp. 3–15, Mar. 2019.
[70] S. S. Choong, L.-P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Inf. Sci.*, vols. 436–437, pp. 89–107, Apr. 2018.
[71] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, Mar. 2003.
[72] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[73] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.
[74] A. Draa, S. Bouzoubia, and I. Boukhalfa, "A sinusoidal differential evolution algorithm for numerical optimisation," *Appl. Soft Comput.*, vol. 27, pp. 99–126, Feb. 2015.
[75] F. Caraffini, G. Iacca, F. Neri, L. Picinali, and E. Mininno, "A CMA-ES super-fit scheme for the re-sampled inheritance search," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1123–1130.
[76] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.
[77] S. M. Elsayed, R. A. Sarker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1932–1937.

[78] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca, and E. Mininno, "Super-fit multicriteria adaptive differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1678–1685.

[79] 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," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.

[80] 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.

[81] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 302–315, Feb. 2015.

[82] D. Kovačević, N. Mladenović, B. Petrović, and P. Milošević, "DE-VNS: Self-adaptive differential evolution with crossover neighborhood search for continuous global optimization," *Comput. Oper. Res.*, vol. 52, pp. 157–169, Dec. 2014.

[83] L.-B. Deng, L.-L. Zhang, N. Fu, H.-L. Sun, and L.-Y. Qiao, "ERG-DE: An elites regeneration framework for differential evolution," *Inf. Sci.*, vol. 539, pp. 81–103, Oct. 2020.

[84] R. Polakova, J. Tvrdik, and P. Bujok, "Evaluating the performance of L-SHADE with competing strategies on CEC2014 single parameter-operator test suite," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1181–1187.

[85] A. Kumar, R. K. Misra, and D. Singh, "Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1835–1842.

[86] P. Bujok and J. Tvrdik, "Enhanced individual-dependent differential evolution with population size adaptation," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1358–1365.

[87] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, and A. Zamuda, "Distance based parameter adaptation for success-history based differential evolution," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100462.

[88] R. Biedrzycki, "A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1489–1494.

[89] Z. Meng, Y. Chen, X. Li, and F. Lin, "PaDE-NPC: Parameter adaptive differential evolution with novel parameter control for single-objective optimization," *IEEE Access*, vol. 8, pp. 139460–139478, 2020.

[90] A. Ghosh, S. Das, and A. K. Das, "A simple two-phase differential evolution for improved global numerical optimization," *Soft Comput.*, vol. 24, no. 8, pp. 6151–6167, Apr. 2020.

[91] Z.-H. Zhan, Z.-J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, early access, Oct. 21, 2019, doi: 10.1109/TCYB.2019.2944873.

[92] Z.-H. Zhan, X.-F. Liu, H. Zhang, Z. Yu, J. Weng, Y. Li, T. Gu, and J. Zhang, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[93] I. De Falco, U. Scafuri, E. Tarantino, and A. Della Cioppa, "An asynchronous adaptive multi-population model for distributed differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 5010–5017.

[94] Y.-F. Ge, W.-J. Yu, J.-J. Li, Z.-W. Yu, and J. Zhang, "Enhancing distributed differential evolution with a space-driven topology," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 4090–4095.

[95] J. Cheng, G. Zhang, and F. Neri, "Enhancing distributed differential evolution with multicultural migration for global numerical optimization," *Inf. Sci.*, vol. 247, pp. 72–93, Oct. 2013.

[96] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 689–707, Oct. 2014.

[97] Y.-Z. Zhou, W.-C. Yi, L. Gao, and X.-Y. Li, "Adaptive differential evolution with sorting crossover rate for continuous optimization problems," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2742–2753, Sep. 2017.

KARAM M. SALLAM received the Ph.D. degree in computer science from the University of New South Wales at Canberra, Australian Force Academy, Canberra, ACT, Australia, in 2018. He is currently a Lecturer with Zagazig University, Zagazig, Egypt. His current research interests include evolutionary algorithms and optimization, constrained-handling techniques for evolutionary algorithms, operation research, machine learning, deep learning, cybersecurity, and the IoT. He was the winner of IEEE-CEC2020 competition. He serves as an organizing committee member of different conferences in the evolutionary computation field and a reviewer for several international journals.

SABER M. ELSAYED (Member, IEEE) received the Ph.D. degree in computer science from the University of New South Wales at Canberra, Australian Defence Force Academy, Canberra, ACT, Australia, in 2012. He is currently a Senior Lecturer with the School of Engineering and Information Technology, University of New South Wales at Canberra, Australian Defence Force Academy. His research interests include evolutionary algorithms, constraint-handling techniques for evolutionary algorithms, scheduling, big data, and cybersecurity using computational intelligence. He was the winner of different IEEE-CEC competitions. He also serves as an organizing committee member of different conferences in the evolutionary computation field. Since 2019, he has been serving as the Chair of the IEEE Computational Intelligence Society (ACT Chapter).

RIPON K. CHAKRABORTTY (Member, IEEE) received the B.Sc. and M.Sc. degrees in industrial and production engineering from the Bangladesh University of Engineering and Technology, in 2009 and 2013, respectively, and the Ph.D. degree in computer science from the University of New South Wales (UNSW), Canberra, ACT, Australia, in 2017. He is currently a Lecturer of system engineering and project management with the School of Engineering and Information Technology, UNSW. He has written two book chapters and over 60 technical journal and conference papers. His research interests include wide range of topics in operations research, optimization problems, project management, supply chain management, and information systems management.

MICHAEL J. RYAN (Senior Member, IEEE) is currently the Director of the Capability Systems Center, University of New South Wales, Canberra. He lectures and regularly consults in a range of subjects, including communications systems, systems engineering, requirements engineering, and project management. He is the author or coauthor of 12 books, three book chapters, and over 300 technical articles and reports. He is also a Fellow of Engineers Australia, the International Council on Systems Engineering, and the Institute of Managers and Leaders. He is also a Co-Chair of the Requirements Working Group in the International Council on Systems Engineering (INCOSE).