Improving Differential Evolution through Bayesian Hyperparameter Optimization

Subhodip Biswas[†], Debanjan Saha[‡], Shuvodeep De[§], Adam D Cobb[¶], Swagatam Das[**] and Brian Jalaian[‖]

[†]Virginia Tech, Arlington, VA, USA — subhodip@cs.vt.edu
[‡]Cognizant Technology Solutions, Kolkata, WB, India — debanjansh@gmail.com
[§]University of Alabama, Tuscaloosa, AL, USA — sde@eng.ua.edu
[¶]SRI International, Arlington, VA USA — adam.cobb@sri.com
[‖]Indian Statistical Institute, Kolkata, WB, India — swagatam.das@isical.ac.in
[**]US Army Research Laboratory, Adelphi, MD, US — brian.a.jalaian.civ@mail.mil

***Abstract*—We propose a novel Evolutionary Algorithm (EA) based on the Differential Evolution algorithm for solving global numerical optimization problem in real-valued continuous parameter space. The proposed MadDE algorithm leverages the power of the multiple adaptation strategy with respect to the control parameters and search mechanisms, and is tested on the benchmark functions taken from the CEC 2021 special session & competition on single-objective bound-constrained optimization. Experimental results indicate that MadDE is able to achieve superior performance on global numerical optimization problems when compared against state-of-the-art real-parameter optimizers. We also provide a hyperparameter optimization algorithm SUBHO for improving the search performance of any EA by finding an optimal set of control parameters, and demonstrate its efficacy in enhancing MadDE's performance on the same benchmark. The source code of our implementation is publicly available at https://github.com/subhodipbiswas/MadDE.**

I. INTRODUCTION

Evolutionary Algorithms (EAs) constitute a class of stochastic, derivative-free, search techniques that make use of randomness for solving difficult mathematical optimization problems in practicable time limits [1]. EAs act as blackbox optimization techniques that are mostly used for solving real-world problems, whose objective functions can be non-continuous, non-differentiable, non-linear, noisy, flat, multi-dimensional, multi-modal or have many local constraints, objectives or noise [2]. Differential Evolution (DE) algorithm is a powerful yet simple EA that was initially proposed by Storn and Price in mid-1990s for optimizing real-valued multi-modal functions [3]–[5]. Over the years DE has gained widespread adoption as a practical optimizer owing to its simple structure, powerful performance and easy implementation [6].

A classical DE algorithm starts by initializing a population of trial solutions to an optimization problem, and then improves the solutions through a cycle of three stages: mutation, recombination/crossover and fitness-based selection. It makes use of three control parameters: population size (NP), scaling factor (F), crossover rate (Cr). The effect of these control parameters on DE's performance has been well-studied and new ways to update them is an active area of research [7], [8]. This has led to a number of improved DE variants being proposed in the last two decades. Some notable examples include SaDE [9], jDE [10], JADE [11], SHADE [12], LSHADE [13], jSO [14] and so on. Over the years, these variants have resulted in state-of-the-art performance on single-objective real-valued continuous global optimization problems proposed by the Congress on Evolutionary Computation (CEC) competition.

In this article, we devise a variant of the DE algorithm that uses a **m**ultiple **ad**aptation (Mad) strategy for simultaneously adapting the control parameters and search mechanisms, and hence the name MadDE. Our approach takes inspiration from the self-adaptive DE variants like JADE/(L)SHADE [11]–[13], whose basic framework has been the foundation behind recent state-of-the-art DE algorithms [15]. However, it differs from the above approaches in employing multiple strategies for mutation and crossover to generate trial vectors. We employ multiple search strategies as they are likely to give an overall consistent performance across a variety of objective function (landscape) with varying properties.

The MadDE algorithm has the following characteristics. *Firstly*, it improvises on existing mutation strategies for carrying out the search and selects them in a probabilistic manner. The probability of selecting a mutation strategy depends on its historical rate of generating successful solutions. *Secondly*, it uses a probabilistic crossover technique that selects between a binomial crossover and its greedy variant called q-best binomial crossover. And *thirdly*, it adapts the DE control parameters—population size NP, crossover rate Cr and scaling factor F—following the LSHADE algorithm [13].

Additionally, motivated by our recent work [16], we propose a hyperparameter optimization (HPO) algorithm for tuning the control parameters of MadDE and, as such, any EA, in general. We call this technique SUBHO which stands for Surrogate-based Bayesian Hyperparameter Optimizer. SUBHO finds good values of control parameters of an EA by maximizing a scoring function that indicates the performance metric of the EA corresponding to a given set of control parameters[1]. It relies on Bayesian Optimization (BO) – a sequential design strategy for global optimization of black-

[1]Kindly note that the terms *control parameter* and *hyperparameter* are used interchangeably here. While the former term is used frequently in the EA community, the latter is unanimously adopted in the ML community.

box functions that do not have any analytical form [17]. BO techniques are generally used to optimize expensive-to-evaluate functions [18] and have found recent success in hyperparameter tuning of ML models [19].

The remainder of the paper is organized as follows. The proposed MadDE algorithm is outlined in Section II followed by experimental investigations on the CEC2021 benchmark problems in Section III. Section IV details about our SUBHO strategy for enhancing performance of any EA in general. Finally, Section V concludes the paper and provides some directions on future research.

II. Proposed Algorithm: MadDE

The MadDE algorithm is designed to solve real-parameter continuous global optimization problem. A trial solution to the problem can be represented as a column vector, $\vec{X} = [x_1, x_2, \ldots, x_D]^T$, where D is the problem dimensionality and each solution component, $x_i, i = 1, 2, \ldots, D$, is a real number. Henceforth, the terms *solution* and *vector* shall be used interchangeably. Usually, an objective (or fitness) function $f(\vec{X})$, which measures either the performance (or cost) of a system, is made available. We are tasked with searching for a solution (or parameter vector) $\vec{X}^*$ that extremizes[2] the objective function, i.e., $f(\vec{X}^*) \leq f(\vec{X}) \; \forall \vec{X} \in \Omega \subseteq \mathfrak{R}^D$, where Ω is a finite set representing the search domain. Here we assume that we are given a maximum evaluation budget of $FEs_{\max}$ to solve the optimization problem. In describing the steps of our algorithm up next, we use terminology and the algorithmic notations consistent with the standard DE literature [7].

A. Initialization

The search for an optimal solution begins by initializing a population $\mathbf{P}$ of NP trial solutions to the optimization problem in D-dimensional parameter space $\mathfrak{R}^D$. The population of solutions is updated in an iterative manner and each iteration is referred to as a generation, $G = 0, 1, \ldots, G_{\max}$. We represent the i^{th} solution of the population at the G^{th} iteration as

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}], \quad i = 1, 2, \ldots, NP. \quad (1)$$

Usually, each solution parameter is restricted to lie within some bounds: $\vec{X}_{\min} = [x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}]$ and $\vec{X}_{\max} = [x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}]$. Given the bounds, we try to ensure that the initial population is well-spread across the real-parameter space by initializing them uniformly randomly:

$$x_{j,i,0} = x_{j,\min} + rand_{i,j}[0,1] \cdot (x_{j,\max} - x_{j,\min}), \quad (2)$$

where i is the index of a solution, j is the solution component, and $rand_{i,j}[0,1]$ is an uniform random number lying in the range $[0, 1]$ and instantiated independently for every i, j pair.

[2]We can restrict our attention to minimization problems without loss of generality since $\max\{f(\vec{X})\} = -\min\{-f(\vec{X})\}$.

B. Mutation

EAs employ mutation operation to randomly perturb/move the solutions in the parameter space in expectation of finding better solutions. In DE, mutation is performed by adding scaled difference vectors to a *target* vector, which is a parent vector selected from the present generation, in order to generate a *mutant* vector, also known as *donor* vector. A simple mutation strategy called DE/rand/1 is enacted as

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}), \quad (3)$$

where $\vec{X}_{r_1^i,G}, \vec{X}_{r_2^i,G}$, and $\vec{X}_{r_3^i,G}$ are three parameter vectors sampled randomly from the population, $r_1^i, r_2^i, r_3^i, i \in [1, NP]$ such that $r_1^i \neq r_2^i \neq r_3^i \neq i$, and F is a scaling factor that usually lies in the range $[0.4, 1.0]$. The role of F is to scale the difference vector, $(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G})$, which gets added to the target vector $\vec{X}_{r_1^i,G}$ to generate the donor vector $\vec{V}_{i,G}$, which represents a new position in the parameter space. For details about different types of mutation, refer to [7].

MadDE uses an ensemble of three mutation strategies as detailed next. The first strategy is the DE/current-to-*p*best/1 mutation with archive based on JADE [11]. Similarly, the next strategy is a variant of the DE/current-to-rand/1 mutation [20] and utilizes an archive. This external archive $\mathbf{A}$ is maintained besides the population $\mathbf{P}$ to store the solutions that fail the selection test. These inferior solutions help to preserve the population diversity by participating in the mutation operation.

- **DE/current-to-$p$best/1 + archive**

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}_{p\text{best},G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_3^i,G}), \quad (4)$$

- **DE/current-to-rand/1 + archive**

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_3^i,G}). \quad (5)$$

$\vec{X}_{p\text{best},G}$ is a solution picked randomly from the top $p\%$ ($p \in [0, 1]$) of the population based on fitness value. $\vec{X}_{r_1^i,G}$ and $\vec{X}_{r_2^i,G}$ are random solutions picked up from the population $\mathbf{P}$. $\vec{X}_{r_3^i,G}$ is selected from the combined population $\mathbf{A} \cup \mathbf{P}$.

The third strategy is a modification of the earlier the DE/current-to-*p*best/1 mutation as shown below.

- **DE/weighted-rand-to-$q$best/1**

$$\vec{V}_{i,G} = F_i \cdot \vec{X}_{r_1^i,G} + F_i \cdot F_a \cdot (\vec{X}_{q\text{best},G} - \vec{X}_{r_2^i,G}), \quad (6)$$

where $\vec{X}_{q\text{best},G}$ is randomly selected solution from the top $q\%$ of the population with q being varied as

$$q = 2p - p \cdot \frac{FEs}{FEs_{\max}}, \quad (7)$$

and F_a is an attraction factor that is varied as

$$F_a = 0.5 + 0.5 \cdot \frac{FEs}{FEs_{\max}}. \quad (8)$$

This strategy is designed to slowly enhance elitism by increasing the attraction towards the top solutions. The greediness of the move is controlled by using randomly selected vectors, i.e., $\vec{X}_{r_1^i,G}$ and $\vec{X}_{r_2^i,G}$, from the population.

The mutant vectors, generated via mutation operation, need to be checked to ensure that they lie within the bound-constrained parameter space. This is performed as follows:

$$\vec{v}_{j,i,G} = \begin{cases} \frac{1}{2}\left(x_{j,i,G} + x_{j,\min}\right) & \textbf{if } v_{j,i,G} < x_{j,\min} \\ \frac{1}{2}\left(x_{j,i,G} + x_{j,\max}\right) & \textbf{if } x_{j,\max} < v_{j,i,G} \end{cases}. \quad (9)$$

C. Crossover

During crossover, the donor vector $\vec{V}_{i,G}$ generated by the mutation operation exchanges its components with the target vector $\vec{X}_{i,G}$ to form the *trial* vector $\vec{U}_{i,G}$. Crossover acts as a diversity preserving mechanism by restricting the modifications in the trial vector. There are two types of crossover: *exponential* and *binomial*. Here, we focus on the latter and use two crossover strategies as follows.

- **Binomial crossover** (BX): Each of the D parameters of the solution participate in crossover whenever a randomly generated number between 0 and 1 is less than or equal to probability of crossover, Cr. In this case, the number of parameters inherited from the donor approximately has a binomial distribution. It is implemented as

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \textbf{if } (rand_{i,j}[0,1] \leq Cr \text{ or } j = j_{rand}) \\ x_{j,i,G} & \textbf{otherwise} \end{cases}, \quad (10)$$

- **q-best binomial crossover** (qBX): This is a greedy variant of the binomial crossover inspired by the DE/current-to-$p$best/1 mutation scheme. In qBX, we replace the target vector in Eq. (10) by a solution randomly picked up from the top $q\%$ of the solutions from the combined population of $\mathbf{P} \cup \mathbf{A}$. q is calculated as per Eq. (7).

We instantiate these crossover strategies in a probabilistic manner based on the parameter p_{qBX} as follows

$$\begin{array}{c} \textbf{if}(rand_i[0,1] \leq p_{\text{qBX}}) \\ \text{Perform qBX} \\ \textbf{otherwise} \\ \text{Perform BX} \end{array} \quad (11)$$

This probabilistic crossover strategy uses the parameter p_{qBX} to control the degree of greediness. It resembles a binomial crossover when p_{qBX} is set to 0, and the greedy qBX when p_{qBX} is set to 1. It is recommended to set p_{qBX} in the range $(0, 0.5]$ for balancing between exploration and exploitation.

D. Selection

The newly generated trial vector $\vec{U}_{i,G}$ enters a one-to-one comparison with its corresponding target vector $\vec{X}_{i,G}$, and the one with higher fitness (or lower objective functional value) survives the process. This fitness-based selection helps to keep the population size constant while preserving the fitter individuals in the process. It is implemented as follows

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \textbf{if } f(\vec{U}_{i,G}) < f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \textbf{otherwise} \end{cases}. \quad (12)$$

E. Multiple adaptation strategy

The Mad strategy is motivated from the success history-based parameter adaptation of (L)SHADE [12], [13]. It maintains a memory archive of the successful DE control parameters−scaling factor F and crossover rate Cr. The memory archives $\mathbf{M}_F$ and $\mathbf{M}_{Cr}$ are initialized to values F_0 and Cr_0, respectively, i.e., $M_{F,i} = F_0, i = 1, \ldots, H$, and $M_{Cr,i} = Cr_0$, $i = 1, \ldots, H$. The memory archives are used to generate the control parameters that take part in the mutation and crossover operation:

$$F_i \sim randc_i\left(M_{F,r_i}, 0.1\right) \quad (13)$$

$$Cr_i \sim randn_i\left(M_{Cr,r_i}, 0.1\right). \quad (14)$$

The scaling factor F_i is sampled from a Cauchy distribution to ensure a better spread of values. F_i is truncated to 1 in case $F_i > 1$, and is regenerated if $F_i < 0$ till a valid value of $F_i \in (0, 1]$ is produced. Similarly, the crossover rate Cr is sampled from the normal distribution and truncated to lie in the range $[0, 1]$. The index r_i is an uniformly sampled random integer lying in the range $[1, H]$.

Following the fitness-based selection in Eq. (12), there is success history-based update of the memory archives [13]. The F_i and Cr_i of trial vectors that replace their target vectors, i.e., $f(\vec{U}_{i,G}) < f(\vec{X}_{i,G})$, are recorded in success-based archives $\mathbf{S}_F$ and $\mathbf{S}_{Cr}$, respectively, and are used to update the memory archives $\mathbf{M}_F$ and $\mathbf{M}_{Cr}$, respectively. If successful selection takes place, i.e., $|\mathbf{S}_{Cr}| > 0$ and $|\mathbf{S}_F| > 0$, then the memory update takes place as

$$M_{F,k,G+1} = \text{mean}_{WL}(\mathbf{S}_{Cr}), \quad (15)$$

$$M_{Cr,k,G+1} = \begin{cases} \perp & \textbf{if } M_{Cr,k,G+1} = \perp \\ & \text{or } \max(\mathbf{S}_{Cr}) = 0 \\ \text{mean}_{WL}(\mathbf{S}_{Cr}) & \textbf{otherwise} \end{cases}, \quad (16)$$

where $\perp$ represents a terminal value of -1. Note that the weighted Lehmer mean mean_{WL} is used in accordance to [21]. It is calculated as

$$\text{mean}_{WL}(\mathbf{S}) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k}, \quad (17)$$

where $w_k = \Delta f_k / \sum_{k'=1}^{|S|} \Delta f_{k'}$ is the weighting factor, $\Delta f_k = |f(\vec{U}_{k,G}) - f(\vec{X}_{k,G})|$ is the amount of fitness improvement is used in order to influence the parameter adaptation, and $\mathbf{S}$ refers to either $\mathbf{S}_{Cr}$ or $\mathbf{S}_F$. This memory update is followed by the index update as $k = \max((k+1) \bmod H, 1)$.

If no successful trial vectors are generated, the index is not updated and the memory archive is reset as

$$\begin{array}{c} M_{F,k,G+1} = 0.5 \\ M_{Cr,k,G+1} = 0.5 \end{array}. \quad (18)$$

Additionally, we perform population size reduction which has been shown to be highly effective in improving the performance of EAs [13], [22], [23]. If NP_0 is the initial population number at generation 0, $NP_{\min}$ is the minimum

population size, then the population size at generation G, $G = 1, 2, \ldots$ is updated as

$$NP_G = \left\lfloor NP_{\max} - (NP_{\max} - NP_{\min}) \frac{FEs}{FEs_{\max}} \right\rceil, \quad (19)$$

where FEs is the number of function evaluations at the end of the given generation, and $\lfloor . \rceil$ is the nearest integer function. Whenever the condition $NP_G < NP_{G-1}$ occurs, the worst $NP_{G-1} - NP_G$ individuals are removed from the population.

Next, we update the probability of applying each mutation strategy based on their past performance as follows

$$p_m = \frac{\Delta F_m}{\sum_{m'=1}^{M} \Delta F_{m'}}, \; m = 1, \ldots, M, \quad (20)$$

where,

$$\Delta F_m = \frac{\sum_{i=1}^{NP} \mathbb{I}_m(\vec{V}_{i,G}) \max(0, f(\vec{X}_{i,G}) - f(\vec{V}_{i,G}))/f(\vec{X}_{i,G})}{\sum_{i=1}^{NP} \mathbb{I}_m(\vec{V}_{i,G})} \quad (21)$$

M is the number of mutation strategies used and $\mathbb{I}_m(\vec{V}_{i,G})$ is an indicator function that evaluates to 1 if trial vector has been generated by the m-th mutation strategy. We ensure that probability p_m of selecting a mutation strategy is truncated to lie in the range $[0.1, 0.9]$, so that each mutation strategy has a fair chance of being applied during a given generation.

Lastly, the solution archive $\mathbf{A}$ is maintained separately and its size is kept at $\lfloor A_{rate} \cdot NP_G \rceil$. Since the population size NP_G decreases with time, so does the archive size.

F. Putting it all together

We provide the pseudocode of MadDE in Algortihm 1. Besides the information about the objective function to be optimized, MadDE requires the following hyperparameters:

- Probability of qBX crossover: p_{qBX}
- Percentage of population in p-best mutation: p
- Archive size multiplier: A_{rate}
- Memory size multiplier: H_m
- Initial population size multiplier: NP_m
- Initial value for memory archive $\mathbf{M}_F$: F_0
- Initial value for memory archive $\mathbf{M}_{Cr}$: Cr_0

Based on the above, we set the following:

- Maximum population size ($NP_{\max}$): $\lfloor NP_m \cdot D^2 \rceil$,
- Minimum population size ($NP_{\min}$): 4,
- Size of archive $\mathbf{A}$: $\lfloor A_{rate} \cdot NP_G \rceil$
- Size of memory archives $\mathbf{M}_F$ and $\mathbf{M}_{Cr}$: $\lfloor H_m \cdot D \rceil$,

III. Experiments and Results

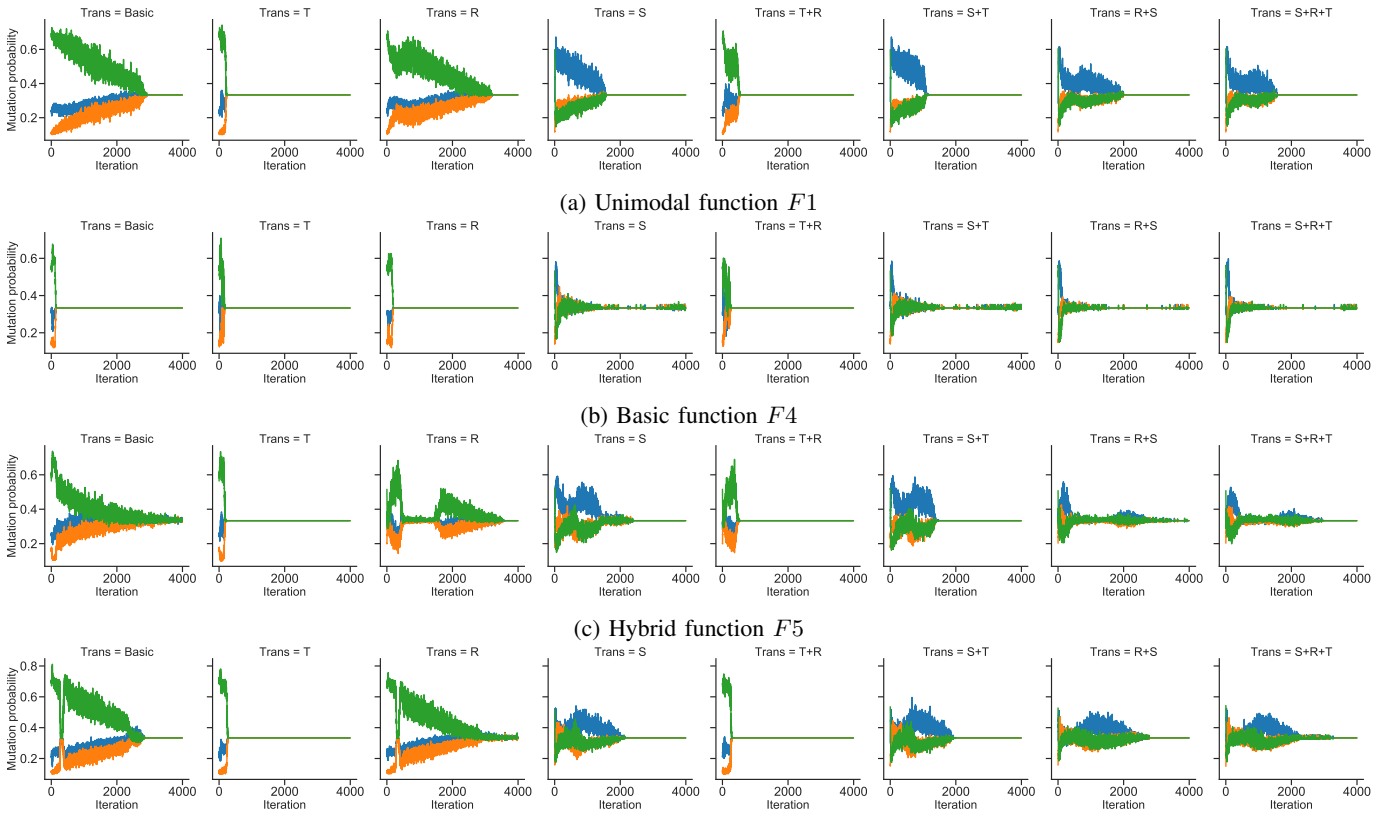
We test MadDE on the CEC2021 Special Session on Real-Parameter Single Objective Optimization benchmark suite. For more details, kindly refer to the technical report in [24]. It consists of 10 functions– $F1$ is an unimodal function, $F2$-$F4$ are basic functions, $F5$-$F7$ are hybrid functions, and $F8$-$F10$ are composition functions. These are bound-constrained single-objective optimization problems in $10D$ and $20D$ real-parameter space, and are subjected to multiple transformations, inlcuding translation (**T**), rotation (**R**), shift (**S**)– and combinations thereof.

Algorithm 1: MadDE algorithm

Data: Objective function:$\mathcal{F}$, Problem size:D, Search bounds: $\vec{X}_{\min}, \vec{X}_{\max}$, Evaluation Budget:$FEs_{\max}$, MadDE hyperparameters: p_{qBX}, p, A_{rate}, H_m, NP_m, F_0, Cr_0
Result: Best possible solution to $\mathcal{F}$
begin
  Set $NP_{\max} \leftarrow \lfloor NP_m \cdot D^2 \rceil$, $NP_{min} \leftarrow 4$.
  $G \leftarrow 1$, $NP_G \leftarrow NP_{\max}$
  Archive $\mathbf{A} \leftarrow \phi$.
  Set all values in $\mathbf{M}_F$ and $\mathbf{M}_{Cr}$ to F_0 and Cr_0, respectively.
  Initialize a population of NP_G solutions using Eq.(2).
  Evaluate the solutions to determine their fitness value
  $FEs \leftarrow FEs + NP$.
  `// Main loop`
  while $FEs < FEs_{\max}$ **do**
    $\mathbf{S}_F \leftarrow \phi$, $\mathbf{S}_{Cr} \leftarrow \phi$
    Determine q and F_a as per Eq. (7) and (8).
    `// Generate trial solutions`
    for $i = 1, 2, \ldots, NP_G$ **do**
      Pick a random value $k \in \{1, \ldots, H\}$
      if $M_{Cr,k} == \perp$ **then**
        $Cr_{i,G} \leftarrow 0$
      else
        $Cr_{i,G} \sim randn_i\left(M_{Cr,k}, 0.1\right)$
      $F_{i,G} \sim randc_i\left(M_{F,k}, 0.1\right)$
      Generate donor vector $\vec{V}_{i,G}$ using mutation strategy m, $m \in \{1, 2, \ldots, M\}$ based on probability p_m
      Check bound-constraints of $\vec{V}_{i,G}$ using Eq. (9)
      Generate trial vector $\vec{U}_{i,G}$ from $\vec{X}_{i,G}$ and $\vec{V}_{i,G}$ using multi-crossover strategy outlined in Eq. (11)
      Evaluate the fitness of the solution $f(\vec{U}_{i,G})$
      $FEs \leftarrow FEs + 1$
    `// Perform fitness-based selection`
    for $i = 1, 2, \ldots, NP_G$ **do**
      if $\mathcal{F}(\vec{U}_{i,G}) < \mathcal{F}(\vec{X}_{i,G})$ **then**
        $\mathbf{S}_F \leftarrow \mathbf{S}_F \bigcup F_{i,G}$
        $\mathbf{S}_{Cr} \leftarrow \mathbf{S}_{Cr} \bigcup Cr_{i,G}$
      if $\mathcal{F}(\vec{U}_{i,G}) \leq \mathcal{F}(\vec{X}_{i,G})$ **then**
        $\vec{X}_{i,G+1} \leftarrow \vec{U}_{i,G}$
      else
        $\vec{X}_{i,G+1} \leftarrow \vec{X}_{i,G}$
        $\mathbf{A} \leftarrow \mathbf{A} \bigcup \vec{X}_{i,G}$
    `// Update the control parameters`
    Update the memory archives $\mathbf{M}_F$ and $\mathbf{M}_{Cr}$ based on the success-based archives, i.e., $\mathbf{S}_F$ and $\mathbf{S}_{Cr}$
    Update the mutation probabilities p_m as per Eq. (20).
    Perform linear population size reduction as per Eq. (18). If $NP_{G+1} < NP_G$, remove the worst $NP_G - NP_{G+1}$ solutions from the population.
    Similarly prune the archive $\mathbf{A}$ if $|\mathbf{A}| > \lfloor A_{rate} \cdot NP_{G+1} \rceil$.
    $G \leftarrow G + 1$
  Determine the best solution $\vec{X}^*$ in the population.
return: $\vec{X}^*$

A. Parametric Setting

For both $10D$ and $20D$ problems, we evaluate MadDE with the given hyperparameter values: $p_{qBX} = 0.01$, $p = 0.18$, $A_{rate} = 2.30$, $NP_m = 2$, $H_m = 10$, $F_0 = 0.20$, and $Cr_0 = 0.20$. We arrived at this hyperparameter setup by running our hyperparameter optimizer SUBHO followed by manual tuning. We shall revisit the steps of hyperparameter tuning in Section IV. The low value for p_{qBX} seeks to minimize the chances premature convergence specially when the functional landscape is multimodal, i.e, has many global/local optima. A high value of H_m allows MadDE to try out more varied range of scaling factor F and crossover rate Cr for generating trial solutions.



(a) Unimodal function $F1$

(b) Basic function $F4$

(c) Hybrid function $F5$

(d) Composite function $F9$

Fig. 1: Plots highlighting the iterative change in the mutation probabilities for different transformations of (a) unimodal function $F1$, (b) basic function $F4$, (c) hybrid function $F5$, and (d) composite function $F9$. The mutations strategies are color-coded: **DE/current-to-$p$best/1 + archive**, **DE/current-to-rand/1+archive** and **DE/weighted-rand-to-$q$best/1**. (**Zoom on the PDF**)

B. Results

As per the competition format, we report the error values obtained by MadDE for the different transformations of each of the functions in both $10D$ and $20D$ in Tables I and II, respectively. Considering the best case, MadDE is able to find the global optima for all the functions with basic transformation. The same observation holds when the functions undergo the translation (**T**), rotation (**R**), or any combinations of it, i.e., rotated and translated (**R**+**T**). When the functions undergo shift (**S**) transformation or any combinations thereof, i.e., shifted and translated (**S**+**T**), and shifted and rotated (**S**+**R**), and shifted, rotated and translated (**S**+**R**+**T**), the performance varies across the functions. However, in such instances, MadDE shows consistent performance as is evident from low to moderate values of standard deviation. Besides, the performance of MadDE is robust to the increase in the problem dimensionality, i.e., $10D$ to $20D$.

The use of multiple mutation strategies enable MadDE to operate efficiently on different functional landscapes. For further insights into the effectiveness of the different mutation strategies for each function (and transformation), we compute the iteration-wise change in the mutation probabilities from Eq. (20) and plot them for the different types of functions in Figure 1. We observe that the DE/weighted-rand-to-$q$best/1 is activated in higher proportion for functions undergoing basic, **T**, **R** and **T**+**R** transformations. For all the shifted functions, i.e., with some combination of the **S** transformation, DE/current-to-$p$best/1 mutation has higher success probability. Overall, the choice of mutation strategies is important in ensuring improved performance across a wide array of functions.

C. Computational Complexity

All the experiments were executed on the following system:

- OS: Windows 10
- CPU: AMD Ryzen 7 3700X (3.59 GHz)
- RAM: 16 GB
- Language: MATLAB 2018a
- Compiler: MinGW-w64 C/C++ Compiler

The algorithm complexity of MadDE was computed according to the detailed instructions in [24]. We measure $T0$, which is the time take to run a test program. $T1$ is the time taken to execute 200,000 evaluations of benchmark function $F1$ with the **S**+**R**+**T** transformation, while $T2$ is the mean of the time taken by MadDE to optimize the same function. Table III shows the complexity results of MadDE on $10D$ and $20D$ problems.

TABLE I: Results of MadDE on $10D$ problems.

Basic	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
T	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
R	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	2.6183E-04	0.0000E+00	8.7276E-06	4.7000E-05
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
S	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	1.0874E+01	1.0874E+01	1.0874E+01	1.0874E+01	1.7764E-15
$F4$	3.3382E-02	2.7450E-01	1.9164E-01	1.8780E-01	4.8012E-02
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	4.3441E-04	3.2769E-02	2.0295E-02	1.6174E-02	1.0641E-02
$F7$	7.2455E-06	6.0856E-03	8.8810E-04	1.4161E-03	1.3996E-03
$F8$	0.0000E+00	1.0000E+02	1.0000E+02	8.7932E+01	3.0962E+01
$F9$	0.0000E+00	1.0000E+02	1.0000E+02	9.3333E+01	2.4944E+01
$F10$	4.0000E+02	4.0000E+02	4.0000E+02	4.0000E+02	0.0000E+00
T+R	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	9.4383E-04	0.0000E+00	3.1461E-05	1.6942E-04
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
S+T	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	6.2454E-02	0.0000E+00	8.3273E-03	2.1230E-02
$F3$	1.0874E+01	1.0874E+01	1.0874E+01	1.0874E+01	1.7764E-15
$F4$	9.8992E-02	2.9787E-01	1.8787E-01	1.9278E-01	5.0664E-02
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	9.6713E-07	4.2611E-02	1.6981E-02	1.4467E-02	1.1843E-02
$F7$	5.6707E-08	7.7190E-03	9.1396E-04	1.5458E-03	1.9031E-03
$F8$	0.0000E+00	1.0000E+02	1.0000E+02	9.3966E+01	2.2711E+01
$F9$	0.0000E+00	1.0000E+02	1.0000E+02	9.0000E+01	3.0000E+01
$F10$	4.0000E+02	4.0000E+02	4.0000E+02	4.0000E+02	0.0000E+00
R+S	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	1.2491E-01	4.0051E+01	1.1037E+01	1.2270E+01	1.0260E+01
$F3$	1.1173E+01	1.5516E+01	1.3633E+01	1.3570E+01	1.1905E+00
$F4$	5.7047E-02	5.6503E-01	3.2506E-01	3.5097E-01	1.1804E-01
$F5$	0.0000E+00	2.6143E+00	1.2031E+00	1.0955E+00	7.9477E-01
$F6$	2.3794E-02	6.2046E-01	3.4063E-01	3.2546E-01	1.6743E-01
$F7$	1.0532E-03	8.7233E-01	1.3514E-01	2.3317E-01	2.4001E-01
$F8$	3.1176E+01	1.0000E+02	1.0000E+02	9.5976E+01	1.5312E+01
$F9$	0.0000E+00	1.0000E+02	1.0000E+02	9.0000E+01	3.0000E+01
$F10$	3.9774E+02	3.9774E+02	3.9774E+02	3.9774E+02	0.0000E+00
S+R+T	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	3.7473E-01	1.2907E+02	1.0245E+01	2.1641E+01	3.2061E+01
$F3$	1.1257E+01	1.5553E+01	1.4154E+01	1.3995E+01	1.1344E+00
$F4$	1.1362E-01	5.8012E-01	3.8265E-01	3.7230E-01	1.0844E-01
$F5$	0.0000E+00	3.4012E+00	4.1629E-01	9.6317E-01	8.4976E-01
$F6$	5.0021E-02	5.5750E-01	3.0428E-01	3.0207E-01	1.2438E-01
$F7$	5.6698E-03	8.7114E-01	6.4364E-02	1.7148E-01	2.1359E-01
$F8$	0.0000E+00	1.0029E+02	1.0000E+02	9.1339E+01	2.6300E+01
$F9$	0.0000E+00	1.0000E+02	1.0000E+02	9.0000E+01	3.0000E+01
$F10$	3.9774E+02	3.9774E+02	3.9774E+02	3.9774E+02	0.0000E+00

TABLE II: Results of MadDE on $20D$ problems.

Basic	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
T	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
R	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
S	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	2.0162E+01	2.0162E+01	2.0162E+01	2.0162E+01	3.5527E-15
$F4$	2.2806E-01	5.8175E-01	4.6759E-01	4.6399E-01	7.5797E-02
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	1.8326E-02	1.1040E-01	5.8624E-02	5.7363E-02	2.1117E-02
$F7$	2.0772E-03	4.6196E-02	2.7698E-02	2.5467E-02	1.1382E-02
$F8$	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	0.0000E+00
$F9$	1.0000E+02	3.0000E+02	3.0000E+02	2.4000E+02	9.1652E+01
$F10$	4.0000E+02	4.0000E+02	4.0000E+02	4.0000E+02	0.0000E+00
T+R	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F4$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F7$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F8$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F9$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F10$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
S+T	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F3$	2.0162E+01	2.0162E+01	2.0162E+01	2.0162E+01	3.5527E-15
$F4$	3.3194E-01	5.6581E-01	4.5050E-01	4.4907E-01	5.5554E-02
$F5$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F6$	3.0258E-03	1.1645E-01	6.1641E-02	6.3045E-02	2.6708E-02
$F7$	2.0623E-03	5.1286E-02	2.6752E-02	2.5604E-02	1.3122E-02
$F8$	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	0.0000E+00
$F9$	1.0000E+02	3.0000E+02	3.0000E+02	2.4000E+02	9.1652E+01
$F10$	4.0000E+02	4.0000E+02	4.0000E+02	4.0000E+02	0.0000E+00
R+S	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	6.2457E-02	7.5913E+00	2.6706E+00	2.7168E+00	2.1527E+00
$F3$	2.0387E+01	2.1893E+01	2.1094E+01	2.1106E+01	4.4049E-01
$F4$	4.3803E-01	7.4917E-01	6.4235E-01	6.1932E-01	7.9815E-02
$F5$	3.4012E+00	7.8429E+01	1.4357E+01	1.9638E+01	1.5278E+01
$F6$	1.5206E-01	8.4822E-01	3.5414E-01	3.9797E-01	1.7004E-01
$F7$	5.1775E-01	9.0110E+00	1.4176E+00	1.7620E+00	1.7463E+00
$F8$	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	0.0000E+00
$F9$	1.0000E+02	4.1424E+02	4.0838E+02	3.2602E+02	1.3638E+02
$F10$	4.1366E+02	4.1366E+02	4.1366E+02	4.1366E+02	5.6843E-14
S+R+T	Best	Worst	Median	Mean	Std
$F1$	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
$F2$	6.2457E-02	9.2676E+00	1.8481E+00	2.5582E+00	1.9448E+00
$F3$	2.0387E+01	2.1780E+01	2.0920E+01	2.0947E+01	4.1756E-01
$F4$	4.8881E-01	7.8483E-01	6.1560E-01	6.2103E-01	6.7549E-02
$F5$	3.4571E+00	3.6431E+01	1.6948E+01	1.7631E+01	8.8026E+00
$F6$	1.0752E-01	6.4624E-01	3.3492E-01	3.6563E-01	1.3207E-01
$F7$	1.0893E-01	9.2745E+00	1.2653E+00	1.9785E+00	2.4884E+00
$F8$	1.0000E+02	1.0000E+02	1.0000E+02	1.0000E+02	0.0000E+00
$F9$	1.0000E+02	4.1326E+02	4.0970E+02	3.2727E+02	1.3708E+02
$F10$	4.1366E+02	4.1366E+02	4.1366E+02	4.1366E+02	5.6843E-14

TABLE III: Algorithm complexity analysis

D	$T0$	$T1$	$T2$	$(T2-T1)/T0$
10	0.0115	0.4029	2.1753	153.9182
20	0.0280	1.8721	12.2638	371.4375

D. Comparative performance

For comparative analysis, we evaluate MadDE with recent state-of-the-art algorithms. The baseline methods include AGSK [25], LSHADE [13], LSHADE_cnEpSin [26], j2020 [27] and IMODE [28]. IMODE was the winner of last year's competition followed by j2020. To evaluate the strength of these algorithms, we didn't activate any local search technique that some of these methods use during the later stages of a trial run. For summarizing the results, we provide the scores obtained by the baseline methods in Table IV. The details of the scoring function is available in the technical report [24].

TABLE IV: Scores achieved by the baseline algorithms.

Scores \ Algo	AGSK	LSHADE	LSHADE_cnEpSin	IMODE	j2020	MadDE
Score 1	33.93	18.16	16.94	28.55	28.16	**50.00**
Score 2	42.70	34.14	30.22	41.44	33.68	**50.00**
Score	76.62	52.30	47.17	69.98	61.84	**100.00**

We observe that MadDE and IMODE are the leading algorithms when functions either don't go any transformation or undergo the **R**, **T** and **R+T** transformations. The **S** transformation poses challenge to the search methods, yet MadDE, j2020 and IMODE are the better performing ones. MadDE suffers slightly for functions undergoing **S** transformation. This may be improved by promoting better population diversity.

For statistically comparing the algorithms, we perform the Friedman test at 5% significance level [29]. The average rank achieved by the baselines based on the mean and median performance are reported in Tables V and VI, respectively. We observe that MadDE is able to achieve the best rank among the state-of-the-art methods in both the cases.

TABLE V: Average rank of all the algorithms based on mean performance as determined by Friedman test.

D \ Algo	AGSK	LSHADE	LSHADE_cnEpSin	IMODE	j2020	MadDE
10	3.10	3.78	4.47	2.94	3.98	**2.73**
20	3.06	3.93	4.23	3.41	3.83	**2.54**

TABLE VI: Average rank of all the algorithms based on median performance as determined by Friedman test.

D \ Algo	AGSK	LSHADE	LSHADE_cnEpSin	IMODE	j2020	MadDE
10	3.08	3.71	4.28	3.22	3.95	**2.76**
20	3.17	3.84	3.86	3.57	3.87	**2.69**

IV. Hyperparameter Optimization

The empirical performance of state-of-the-art optimization algorithms for solving computationally difficult problems can be improved by modifying the set of parameters used by these algorithms. This set of parameters are often referred to as *hyperparameters*, specially in the Machine Learning (ML) domain. Finding a good value of these hyperparameters, often requires tedious manual exploration of the resulting combinatorial space of parameter settings and may lead to unmanageable outcomes. This is also known as the general *algorithm configuration* problem [30].

Our recent work [16] demonstrated the advantage of using Bayesian Optimization (BO) techniques in hyperparameter optimization (HPO) of ML algorithms. Motivated by this, we devise the Surrogate-based Bayesian Hyperparameter Optimizer (SUBHO) for finding a desirable hyperparameters in EAs. The end goal is to improve the empirical performance of stochastic optimization algorithms like MadDE, and any EA, in general.

SUBHO leverages the power of BO in enabling interpolation of performance between observed hyperparameter settings and subsequent extrapolation to previously unseen regions of hyperparameter space. BO is a class of ML-based optimization methods for solving problems of the form [18]

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\arg\max} \quad f(\mathbf{x}), \tag{22}$$

where $\mathcal{X}$ is some $d-$dimensional search space of interest, i.e., $\mathcal{X} \subseteq \mathfrak{R}^d$, such that d is not too large ($d \leq 20$). The objective function usually has the following properties:

- f is "expensive to evaluate"$-$ usually each evaluation takes a substantial amount of time (in order of minutes or hours), and may be performed in limited capacity.
- f is a black-box function, i.e., it lacks known special structure like concavity or linearity that would make it easy to optimize using techniques that leverage such structure to improve efficiency.
- f has no derivative information, which prevents the application of first- and second-order optimization methods. However, we should be able to evaluate it for any $\mathbf{x} \in \mathcal{X}$.
- f may be obscured by stochastic noise. In our work, we assume it to be noise-free.

A BO algorithm has two main components: a (Bayesian) statistical model for modeling f, typically a Gaussian process (GP) regression; and an acquisition function for deciding where to sample next set of observations, which is often expected improvement (EI) [17]. The BO process starts with an initial space-filling experimental design, which consists of n_0 trial solutions chosen uniformly at random and evaluated using f. A typical iteration of BO consists of GP regression and EI acquisition function. The statistical model, i.e., the GP, estimates a (Bayesian) posterior probability distribution that describes potential values for $f(\mathbf{x})$ at a candidate point $\mathbf{x}$. Hence, each time the GP observes a new point $\{\mathbf{x}, f(\mathbf{x})\}$, the posterior distribution is updated. The acquisition function $\alpha(\mathbf{x})$ predicts the objective functional value at a new (unseen) point $\tilde{\mathbf{x}}$ based on the current posterior distribution over f.

Usually the GP regression produces a posterior probability distribution on each $f(\mathbf{x})$ that is normally distributed with $\mathcal{N}\left(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x})\right)$. The mean $\mu_n(\mathbf{x})$ can be interpreted as a point estimate of $f(\mathbf{x})$. The credible interval $\mu_n(\mathbf{x}) \pm 1.96 \times \sigma_n(\mathbf{x})$ is equivalent to confidence interval in frequentist statistics, and is likely to contain $f(\mathbf{x})$ with 95% probability according to the posterior distribution. The mean interpolates the previously evaluated points. The width of the credible interval is 0 at these previously evaluated points, and grows wider as we move away from them indicating uncertainty.

The acquisition function $\alpha(\mathbf{x})$ corresponds to the posterior distribution and takes value 0 at points where $\{\mathbf{x}, f(\mathbf{x})\}$ is already known. This is a reasonable assumption when the f is noise-free because evaluating $\mathbf{x}$ at these points provides no additional information for solving (22). Note that $\alpha(\mathbf{x})$ has higher value for points with larger credible intervals, because observing a point where we are more uncertain about f is expected to be more useful in finding good approximate global optima. Also, $\alpha(\mathbf{x})$ tends to have high value near points with high posterior means, because such points may be proximal to approximately good global optima.

At every iteration t, a new point $\mathbf{x}_t$ is generated by optimizing $\alpha(\mathbf{x})$ and evaluated $f(\tilde{\mathbf{x}})$, following which the GP updates the posterior distribution based on $\{\mathbf{x}_t, f(\mathbf{x}_t)\}$. This iterative process takes place till the budget of N function evaluations is expended. The output is an approximately global point $\{\mathbf{x}^*, f(\mathbf{x}^*)\}$ determined by BO. In SUBHO, $\mathbf{x}_t$ represents the hyperparameters of any EA. Here, we consider the hyperparameters of MadDE, namely $p_{_{qBX}}$, p, A_{rate}, $H_{_m}$, $NP_{_m}$, F_0, and Cr_0, as inputs (refer to Section II-F). In Table VII, we define the design space of SUBHO using integer variables and relate them to the MadDE hyperparameters. This combinatorial design space enables us to control the hyperparameters with an accuracy of 2 decimal points.

TABLE VII: Defining the design space of SUBHO

Variable	Type	Range	Hyperparamter	Relation
q_cr_rate	Integer	[1, 5]	$p_{_{qBX}}$	$p_{_{qBX}} = 0.01\times$ q_cr_rate
p_best_rate	Integer	[5, 25]	p	$p = 0.01\times$ p_best_rate
arc_rate	Integer	[100, 300]	A_{rate}	$A_{rate} = 0.01\times$arc_rate
mem_mult	Integer	[100, 1000]	$H_{_m}$	$H_{_m} = 0.01\times$q_cr_rate
pop_mult	Integer	[100, 500]	$NP_{_m}$	$NP_{_m} = 0.01\times$q_cr_rate
sf_init	Integer	[10, 90]	F_0	$F_0 = 0.01\times$sf_init
cr_init	Integer	[10, 90]	Cr_0	$Cr_0 = 0.01\times$cr_init

In optimization algorithms, the search performance is measured using performance profiles [31], empirical distribution function [32], and so on. However, computing such functions might be complicated and time-consuming. Here, we consider a simple variant of the scoring function provided by the competition organizers [24]. Let $\mathbf{x}_0$ and $\mathbf{x}_t$ be the hyperparameters of MadDE found by manual tuning and by SUBHO at iteration t, respectively. While $\mathbf{x}_0$ is fixed, SUBHO suggests new hyperparameters $\mathbf{x}_t$ iteratively. We compare the hyperparameters $\mathbf{x}_t$ with $\mathbf{x}_0$ based on the performance function

$$P(\mathbf{x}_t) = score_{\mathbf{x}_0} + \left(100 - score_{\mathbf{x}_t}\right), \tag{23}$$

where $score_{\mathbf{x}_0}$ and $score_{\mathbf{x}_t}$ correspond to the manually-tuned and algorithmically-tuned MadDE, respectively. These two scores are computed as per page 20 of the technical report [24]. When SUBHO is able to find better hyperparameters $\mathbf{x}_t$, the value of $score_{\mathbf{x}_t}$ increases reaching a maximum of 100. As $score_{\mathbf{x}_t}$ increases, the value of $score_{\mathbf{x}_0}$ usually decreases since the scoring mechanism in [24] assesses the performance of one algorithm relative to the other. Consider two sets of values for $(score_{\mathbf{x}_0}, score_{\mathbf{x}_t})$: $(90, 100)$ and $(95, 100)$. The former one corresponds to a better hyperparameter configuration $\mathbf{x}_t$ as the difference between their score is more. Hence, the design of the performance function $P(\mathbf{x}_t)$ in Eq. (23).

SUBHO minimizes $P(\mathbf{x}_t)$ using MATLAB's `bayesopt` library,[3] which is designed to find optimal hyperparameters of ML algorithms by minimizing their loss function. We specify a budget of 128 function evaluations to `bayesopt` with the EI acquisition function and noise-free[4] objective function $P(\mathbf{x}_t)$. Figure 2 depicts a sample output of SUBHO from MATLAB's terminal. We observe that at iteration 16 the best-so-far score of 91.686 is obtained, which equates to $score_{\mathbf{x}_0}$ = 91.686 and $score_{\mathbf{x}_t}$=100. The corresponding MadDE hyperparameters are $p_{_{qBX}}$ = 0.02, p = 0.15, A_{rate} = 2.02, $H_{_m}$ = 5.08, $NP_{_m}$ = 1.54, F_0 = 0.16, and Cr_0 = 0.13, based on Table VII. Thus, SUBHO helps to achieve desired hyperparameters $\mathbf{x}_t$ of an EA (like MadDE here) by comparing its performance with a manually-found hyperparameter setting of the same EA.

V. CONCLUSION

We propose a multi-adaptation based DE algorithm for the CEC 2021 competition. The usage of multiple strategies for mutation and crossover along with linear population size adaptation helps MadDE to obtain improved convergence across different class of objective functions. Exhaustive comparison with state-of-the-art algorithms highlight the superiority of our method in solving real-parameter optimization problems. Additionally, we provide a hyperparameter optimization algorithm called SUBHO that helps in finding a good range of hyperparameter values for MadDE, and can be very well be extended to any EA in general. A future research direction can be undertaken to apply SUBHO to improve the performance of constrained optimization algorithms [33].

ACKNOWLEDGMENT

Research reported in this paper was sponsored in part by the DEVCOM Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

[3] https://www.mathworks.com/help/stats/bayesopt.html

[4] This is due to the random number seeding of MadDE. Even then the performance should vary across different computing platforms.

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	q_cr_rate	p_best_rate	arc_rate	mem_mult	pop_mult	sf_init	cr_init
1	Best	95.799	3359.5	95.799	95.799	4	18	252	659	109	22	21
2	Accept	117.92	2102.1	95.799	95.799	3	19	126	447	490	16	77
3	Accept	118.54	2289.2	95.799	95.799	3	6	201	467	498	90	11
4	Accept	113.86	2342.2	95.799	95.799	5	20	211	222	358	81	82
5	Accept	115.33	3347.1	95.799	95.799	4	6	206	743	100	87	43
6	Accept	105.93	3485.1	95.799	95.799	3	19	299	163	101	31	59
7	Accept	104.05	3437.8	95.799	95.799	5	24	264	462	102	52	34
8	Accept	100.48	3064.6	95.799	95.799	2	22	235	494	135	10	65
9	Accept	96.082	3089.4	95.799	95.799	3	21	262	648	138	10	32
10	Accept	102.81	3200.8	95.799	95.799	5	19	256	900	122	11	64
11	Accept	95.982	3468.2	95.799	95.799	4	13	286	499	104	11	16
12	Best	92.99	2752.8	92.99	92.99	2	8	148	575	163	17	11
13	Accept	99.869	2766.7	92.99	92.99	4	10	277	605	205	13	10
14	Accept	104.76	3023.1	92.99	92.99	3	10	104	213	121	10	11
15	Accept	98.167	2965.1	92.99	92.99	3	16	114	608	129	12	10
16	Best	91.686	2899.4	91.686	91.686	2	15	202	508	154	16	13
17	Accept	97.923	2871	91.686	91.686	2	22	184	763	165	14	19

Fig. 2: MATLAB output of SUBHO while searching for an improved hyperparameter configuration of MadDE.

REFERENCES

[1] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2015.

[2] K. Fleetwood, "An introduction to Differential Evolution," in *Proceedings of Mathematics and Statistics of Complex Systems One Day Symposium, 26th November, Brisbane, Australia*, 2004, pp. 785–791.

[3] R. Storn and K. Price, "Differential Evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, University of California - Berkeley, Tech. Rep. TR-95-012, 1995.

[4] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by Differential Evolution," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 842–844.

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

[6] K. V. Price, *Differential Evolution*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 187–214.

[7] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

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

[9] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[10] 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 Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[11] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[12] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for Differential Evolution," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78.

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

[14] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jso," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1311–1318.

[15] "Step-by-step improvement of jade and shade-based algorithms: Success or failure?" *Swarm and Evolutionary Computation*, vol. 43, pp. 88–108, 2018.

[16] S. Biswas, A. D. Cobb, A. Sistrunk, N. Ramakrishnan, and B. Jalaian, "Better call Surrogates: A hybrid Evolutionary Algorithm for Hyperparameter optimization," *arXiv preprint arXiv:2012.06453*, 2020.

[17] J. Mockus, *Bayesian approach to global optimization: theory and applications*. Springer Science & Business Media, 2012, vol. 37.

[18] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[19] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019, pp. 3–33. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5_1

[20] K. V. Price, *An Introduction to Differential Evolution*. GBR: McGraw-Hill Ltd., UK, 1999, p. 79–108.

[21] Fei Peng, K. Tang, Guoliang Chen, and X. Yao, "Multi-start jade with knowledge transfer for numerical optimization," in *2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1889–1895.

[22] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[23] J. L. J. Laredo, C. Fernandes, J. J. Merelo, and C. Gagné, "Improving genetic algorithms performance via deterministic population shrinkage," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 819–826.

[24] A. Wagdy, A. A. Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization," Nanyang Technological University - Singapore, Tech. Rep., November 2020. [Online]. Available: https://github.com/P-N-Suganthan/2021-SO-BCO/

[25] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, and N. H. Awad, "Evaluating the performance of Adaptive Gaining Sharing Knowledge based algorithm on cec 2020 benchmark problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

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

[27] J. Brest, M. S. Maučec, and B. Bošković, "Differential evolution algorithm for single objective bound-constrained optimization: Algorithm j2020," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

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

[29] J. Carrasco, S. García, M. Rueda, S. Das, and F. Herrera, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review," *Swarm and Evolutionary Computation*, vol. 54, p. 100665, 2020.

[30] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[31] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

[32] N. Hansen, A. Auger, D. Brockhoff, D. Tušar, and T. Tušar, "Coco: performance assessment," *arXiv preprint arXiv:1605.03560*, 2016.

[33] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm and Evolutionary Computation*, vol. 56, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650219308946