IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 September 2023, accepted 1 October 2023, date of publication 10 October 2023, date of current version 19 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3323017

RESEARCH ARTICLE

A Hybrid Multi-Population Reinitialization Strategy to Tackle Dynamic Optimization Problems

S. RAGHUL AND G. JEYAKUMAR
Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Coimbatore 641112, India
Corresponding author: G. Jeyakumar (g_jeyakumar@cb.amrita.edu)

ABSTRACT In conventional optimization problems, it is assumed that all relevant parametric constraints remain stationary. In contrast, optimization problems encountered in practical applications are dynamic and supervened by uncertainties. The research community has evinced a keen interest in multi-population approaches combined with nature-inspired algorithms to manage dynamic optimization problems efficiently. Applying multi-population approaches to solve dynamic optimization problems engenders specific vital issues, such as reproducing sub-populations in new environments influenced by archival information. Moreover, over-partitioning the population may lead to aberrant utilization of computational resources among the sub-populations. These impediments are addressed using the proposed hybrid multi-population reinitialization strategy, which is a combination of distributed differential evolution algorithmic framework and re-initialization strategy. This scheme relies on simple reinitialization to surmount the dynamism. This framework was assessed on different instances in a moving peak benchmark problem, a proven benchmarking function in the domain of dynamic optimization. Furthermore, this study encompasses a comparative and statistical analysis to validate the effectiveness of the proposed approach in comparison to cutting-edge algorithms in solving dynamic optimization problems efficiently. The experimental results consistently show that the *hybrid multi-population reinitialization strategy* outperforms conventional Differential Evolution algorithms across various parameter configurations. This hybrid multi-population reinitialization strategy showcases its effectiveness in the successful handling of increased shift lengths and number of peaks, which are pivotal parameters in solving moving peak benchmark function.

INDEX TERMS Dynamic optimization problem, evolutionary algorithms, differential evolution, moving peaks benchmarking problems, multi population strategies.

I. INTRODUCTION

Evolutionary algorithms (*EAs*), including the highly effective differential evolution (*DE*) algorithm introduced by Storn and Price in 1995 for solving real-space optimization problems. *DE* offer an efficient approach for addressing global optimization challenges without the need for gradient information. *DE* is renowned for its minimal control parameters and strong convergence capabilities, establishing itself as one of the most powerful *EAs*. However, classical *DE* algorithms face challenges related to parameter configurations, evolution strategies, prolonged computational time, and the high volume of function evaluations. Unlike static optimization problems, many real-world optimization tasks are inherently dynamic, subject to changing conditions over time. For instance, in applications such as object detection the performance of sensitive components is influenced by immediate surroundings. Hence scheduling decisions in dynamic resource environments are continually affected by changing resources and incoming workpieces. Furthermore, financial trade models experience fluctuations due to shifts in market conditions. These challenges belong to the realm of dynamic

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

optimization problems (*DOPs*). The complexity, uncertainty, and diversity of these real-world problems are increasing. Consequently, employing a single evolutionary strategy with traditional serial processing methods faces significant challenges when adapting to these dynamic conditions.

Most of the studies apropos in the Evolutionary Computation (*EC*) based optimization are typified by static data appended by static problem constraints throughout the optimization process. However, exceptionally over two decades, various real-world optimization problems around us are seen to be non-stationary in nature. This stimulates the researchers to explore on proposing algorithms to solve the optimization problems whose characteristics are changing dynamically. The data, constraints, and solutions to these problems are naturally dynamic and inconsistent. The solution space is obscure until unstable events are set in motion. These optimization problems, whose characteristics change while solving them, are known as (*DOPs*) [1], [2], [3], [4].

A *DOP* with the maximization objective can be formulated as shown in equation (1).

$$\max f\ (X, e) = f(x_1, x_2, \ldots\ldots, x_D, e) \tag{1}$$

where f is the objective function to be optimized, X is the decision vector with dimension D, x_i is a decision variable whose values are in the range of $[x_{\text{i.min}}, x_{\text{i.max}}]$, and e denotes a dynamically changing environment. Various approaches have been reported in the literature for efficiently obtaining the solutions to *DOPs*.

The *EAs* are a pool of optimization algorithms under the domain of *EC*. Though the classical *EAs* can efficiently solve the given static optimization problems, due to their natural behavior and intrinsic attributes, they render inefficient solutions for *DOPs*.Loss of population diversity is one of the major factors restraining *EAs'*efficacy in solving *DOPs*. During the search process, diversity loss ensues from the reduced variation among the population members. The entire population converges into promising regions that increase the similarity among the individual in the population. When any dynamic event is triggered, the classical *EAs* cannot quickly diverge to another promising area in the search space.

Several diversity-based approaches have been introduced and categorized in the literature to overcome the mentioned confines. Cobb et al. [5] proposed a hyper-mutation strategy concomitant with the diversity-based approaches. In this strategy, the mutation rate is increased in response to the detection of change. It is one of the initial attempts at this category. Similar approaches entail adjustment of crossover rates and scaling factors triggered by encounters with dynamic events [6], [7]. Krishnakumar et al. proposed a simple re-initialization technique where the algorithm resets to its initial configuration on the occurrence of a dynamic event [8]. Such routines dynamically address changing environments as a separate static optimization problem. This superior diversity technique is accompanied by a loss of historically acquired information. This article serves as a fundamental concept for the proposed approach. To overcome the limitation, the method utilized here utilizes historical information obtained during the process and navigates the search process in the subsequent generations. Samples of re-initializing techniques were addressed in [9] and [10], where a designated subset of the population alone gets re-initialized.

In preference to incorporating techniques that increase a population's diversity, the diversity level can be restricted to fall after a specific pre-defined threshold value. Self-adaptive algorithms aiming to improve diversity in the population fall into this category. Incorporation of an adaptive mechanism into *EAs* alleviates manual tunning, besides increasing the search behavior of the algorithm [11], [12]. Grefenstette et al. [13] recommended an immigrant approach, where randomly generated individuals are periodically introduced into the population. Likewise, Branke and Blackwell [14] revealed a similar approach for managing *DOPs* using multi-swarm-based particle swarm optimization techniques.

Highly preferred approaches to solve *DOPs* are memory-based and prediction-based methods. Memory-based strategies target the maintenance of diversity. Whereas, prediction-based methods employ an algorithm to learn patterns from past generations, for predicting future changes. These types of approaches comprise simple archives where data of previous generations are stored, and it acts as a crucial factor in navigating the search process. These archives are extensively utilized for predicting future change when a change in environment has occurred [15]. A detailed survey on this topic is bestowed in [4], and some memory-based methodologies are presented in [16] and [17].

The multi-population approaches increase diversity by reorganizing the initial population into numerous sub-populations, spreading them across a search space. This approach effectively aids the algorithm in detecting the change, and each sub-population is liable for diversifying or intensifying targeted search space. Mendes et al. [18] proposed Differential Evolution (*DE*) algorithm dealing with multi-populations to solve *DOPs*. Each sub-population in the experiment is assigned to different regions in the search space. *DE* algorithm provided better results in solving the Moving Peak Benchmark (*MPB*) problem. Yang and Li [19] presented a fast-multi-swarm algorithm in which the population was categorized into children and parents. The parent swarm explored the complete search space looking for the global optima, whereas the child swarm explore the best individual identified by the parent swarm. Application of this algorithm to the *MPB* problem produced comparative results [20]. The same investigators proposed a swarm-based clustering algorithm where the swarm was classified by clustering methods, which easily enabled the algorithm to track and locate the optima. The following factors drove the endorsement of the multi-population approaches as being effective and preferred for solving *DOPs* [12].

i. The decomposition of the initial population into sub-populations ensures population diversity.
ii. A concurrent search of different areas in the search space enables efficient tracking of optima.

iii. Any algorithm dealing with a single population can be incorporated into a multi-population approach.

Equipping natural *EAs* to solve *DOPs* is a common interest to the research society. Among *EAs*, the *DE*algorithm is preferred, spurred by its stochastic and meta-heuristic characteristics. Brest et al. recommended an adaptive *DE* (*jDE*) to solve *DOPs* [21]. An enhanced version of the *jDE*algorithm [22] had a noticeable performance in solving Congress on evolutionary computation 2009's *DOPs*. A multi-population-based *DE*, proffered by Mendes et al. in their work [18], does not require any strategy for parameter control to solve *DOPs*. Similarly, Halder et al. [23] suggested a cluster-based *DE* with an external archive, in which the population is decomposed using a clustering strategy. Zhan et al. [24] presented a cloud-based *DE,*one of the most prominent strategies to solve *DOPs* efficiently. In this experiment, an individual migration strategy is employed among the populations. The cloud acts as a master node, and each population member is treated as a computing resource. The computational resources are programmed by the master node, which makes the algorithm's time complexity small. Zhan's team propounded the Cloudde algorithm, an extension of previous work that showed promising performance in solving *DOPs*. Besides the *DE* algorithm, researchers used other swarm algorithms to solve *DOPs* effectively [20], [25]. Following the stance of incorporating *EA*s to solve *DOPs*, many attempts have been initiated to solve *DOPs* in real-time [26], [27], [28] using *DE* as a base algorithm.

A significant limitation observed in the surveyed approaches that effectively address *DOPs* is the increase in error metrics as the complexity of the problem escalates. For instance, when dealing with problems like *MPB* function, these approaches tend to exhibit a notable rise in error rates when variables such as change frequency, shift length, and the number of peaks are increased.

To overcome these limitations, this work proposes a simple hybrid multi-population reinitialization strategy (*HMRS*) following the multi-population approaches. This work utilizes a distributed *DE* (*dDE*), a native extension of the classical *DE* algorithm, to solve *DOPs*. The *dDE* has proven to be fault-tolerant [29] and has been used to solve various real-world optimization problems [30], [31]. Fixed region-based sub-population is used to generate individuals of a subpopulation. The individual within a subpopulation is randomly generated within a restricted bound, which is pre-defined. The success rate of each sub-population is calculated after every successful generation. The sub-population re-initialization takes place based on the success rate and change strength. The population re-initialization mechanism is incorporated with the proposed approach to avoid stagnation at local optima. The proposed *HMRS* is employed on the *MPB* problem to evaluate the effectiveness of the strategy. Compared to the State-of-the-Art methods, the proposed method *HMRS* generates estimable results in several instances. The key contributions of the article are stated below.

1. **Hybrid Multi-Population Reinitialization Strategy (*HMRS*):** The *HMRS* is a simple yet effective hybrid multi-population reinitialization strategy and is a novel approach that follows multi-population methodologies. This strategy aims to enhance the performance and effectiveness of evolutionary algorithms in addressing *DOPs*.
2. **Utilization of Distributed Differential Evolution (*dDE*):** The work leverages *dDE*, an extension of the classical *DE* algorithm, to tackle *DOPs*. The use of *dDE* is particularly noteworthy due to its fault-tolerant properties, as demonstrated in previous research [29]. This approach has been successfully applied to a diverse set of real-world optimization problems, highlighting its applicability and robustness.
3. **Fixed Region-Based Sub-Population and Reinitialization Mechanism:** The work introduces a fixed region-based sub-population generation method, where individuals within a subpopulation are generated randomly within predefined bounds. Additionally, a mechanism for reinitializing sub-populations based on success rates and change strength is incorporated. This technique is crucial for preventing stagnation at local optima and promoting exploration in the optimization process.

These contributions collectively contribute to the proposed *HMRS's* effectiveness, as demonstrated in the evaluation on the *MPB* problem, showcasing superior performance compared to state-of-the-art methods.

The rest of the paper is organized as follows. An overview of *DE* and *dDE* (distributed *DE*) is presented in Sections II and III. The proposed solution is detailed in Section IV, followed by the design of experiments in Section V. Section VI details the results and discussions, and the closing remarks are noted in the conclusions presented in Section VII.

II. THE CLASSICAL DE

Storn and Price [32] introduced the *DE* to the society of evolutionary computing by solving the Chebyshev polynomial problem. In solving numerous benchmarking and real-world problems, *DE* has been evidenced as an efficient method. Highly stochastic and meta-heuristic characteristics aid *DE* in easily addressing various optimization problems. *DE* is a population-based algorithm that utilizes crossover and mutation techniques like other *EAs*. Nevertheless, *DE* possesses a unique *differential mutation* operator, making it stand out from all other *EAs*. The differential mutation operator sums up the scaled difference between two solution vectors in the population to generate the mutant vector. Further, *DE* generates a target vector by implementing a recombination mechanism between the mutant vector and the current vector. A selection mechanism is employed to choose between the target and current vectors; the extant vector can pass for upcoming generations.

Algorithm 1 ***DE*** algorithm

1 Begin
2 Generate initial population
3 Do
4 For every individual (i) in the population
5 Generate three random integers (r_1, r_2, r_3) ∈ (1, *NP*)
6 Generate a random integer j_{rand} ∈ (1, *D*)
7 For every parameter j:
8 $$x_{j,i} = \begin{cases} x_{j,r_3} + F * \left(x_{j,r_1} - x_{j,r_2}\right) & if\ rand\ (0,1) < CR\ or\ j = j_{rand} \\ x_{j,i} & otherwise \end{cases}$$
CR represents the crossover rate, F represents mutation scale factor
9 End For loop;
10 Replace (U_i) trail vector with (X_i) target vector, if target vector is better.
11 End For loop;
12 Repeat Until (Termination criteria is satisfied)
13 End

FIGURE 1. The structure of the DE algorithm.

DE is a search-based algorithm that efficiently explores the search space using random sampling techniques. The initial population is created by manipulating the pre-defined *NP*(Population size) vectors with *D*-Dimensions; each individual member in the population is a candidate solution. The candidate solutions are represented as $X_{i,G} = \left\{x^1_{i,G}, ..., x^D_{i,G}\right\}\ i = i, \ldots, NP$. Where $X_{i,G}$ is the target vector. Furthermore, uniform random techniques are used to initialize the candidates, ensuring the entire search space is covered by the population. Once the initial population is fixed, an iterative process is initiated. For every succeeding generation, a new population is produced by the *DE* algorithm.

During every generation, for every vector in the population, a new mutant vector, $V_{i,G} = \left\{v^1_{i,G}, ..., v^D_{i,G}\right\}$ is generated by applying the differential mutation operation. The crossover/recombination mechanism is initiated on termination of the mutation operation. The recombination of the target vector $X_{i,G}$ and the mutant vector $V_{i,G}$ produces a new trail vector, $U_{i,G} = \left\{u^1_{i,G}, ..., u^D_{i,G}\right\}$.

Succeeding, the selection mechanism compares the target vector $X_{i,G}$ with the trail vector $U_{i,G,}$, allowing the survivor for the next generation. This iterative process continues until the termination criteria are satisfied. In addition, parameter tuning plays a vital role in enhancing the performance of the classical *DE* algorithm. The detailed working of the *DE* algorithm is shown as Algorithm 1 in Figure 1.

Numerous self-adaptive parameter approaches are proposed in the literature [33], [34], [35], [36] to solve different optimization problems. Algorithmic simplicity, ease of implementation, and strong convergence have inspired many researchers to employ *DE* to solve various optimization problems. Active research on *DE* for the past two decades has yielded numerous *DE* variants summarized in [37]. An empirical study on the performance of *DE* variants was studied by

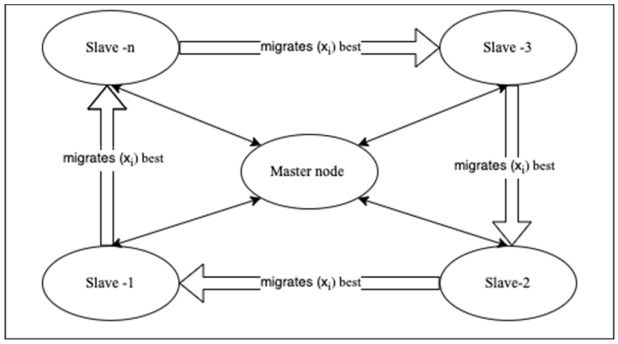

FIGURE 2. The general architecture of the dDE framework.

Praveen et al. [38]. Mixing of *DE* variants affords promising options to solve optimization problems, yielding vitreous results [39], [40], [41], [42]. In [43] and [44], Shunmuga velayutham et al. has proposed various meta-evolutionary selections of constituents in ensemble *DE*, through which the algorithm parameters can be configured automatically based on the targeted optimization problem. They also proposed a generic assembly source code-based framework that facilitates any *EA* to provide antigens in malware scanners [45]. The *dDE* is a native extension of *DE* in a distributed framework, and is detailed in the next section.

III. THE DISTRIBUTED DE (dDE) FRAMEWORK

The exponential growth of technology and the heightened levels of parallelism for solutions to complex optimization problems pose collateral concerns for the research community. To attain a maximum level of parallelism, this study adopted the *dDE* framework, which reduces the computational time. The *dDE* framework acts as a platform for the proposed *HMRS* strategy, through which a multi-population approach is achieved.

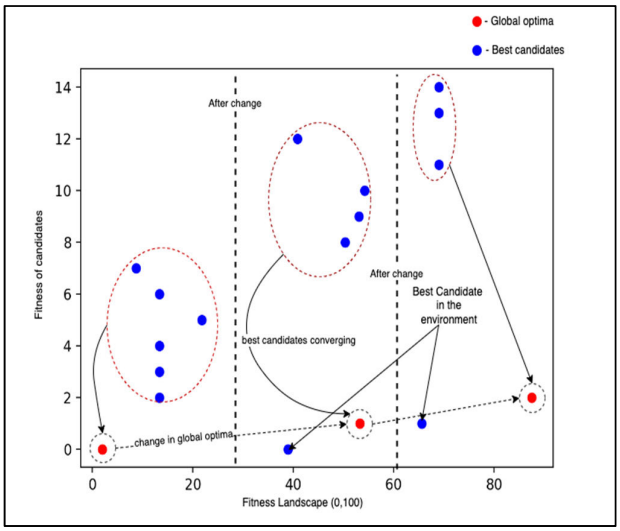

FIGURE 3. An illustration of the workflow of the proposed strategy.

The *dDE* framework follows the principles of the Island Model (*IM*). The concept of natural evolution is the base for *IMs,* which distribute the initial population throughout the search space. The population is decomposed and distributed among the islands to support the principle of simultaneous exploration. The *dDE* framework in this experiment customizes the islands to have one master node and a set of slave nodes. The master node generates an initial population and distributes it to all the slave. Every slave node, incorporated with the *DE* algorithm, works in parallel to ensure synchronization. During each generation, the objective of the slave node is to find an optimal solution for the given sub-population. Adhering to a strict migration topology, after certain number of generations (called migration frequency [46]), the slave nodes can exchange their best candidates with their neighboring nodes. A ring topology was adopted to provide a cooperative evolutionary environment. The architecture representation of the *dDE* framework is shown in Figure 2.

IV. THE PROPOSED HMRS STRATEGY

The proposed hybrid strategy is a commix of re-initialization techniques through which dynamism is managed efficiently. It combines the memory-based approach with the population reinitialization-based approach. The random test point method is used to detect the change in the environment. If the change is detected, its severity is calculated which is crucial in the proposed strategy. This strategy utilizes a population aging mechanism and success rate-based reinitialization technique to reinitialize each sub-population. It requires a cooperative evolutionary environment to synchronize master and slave nodes.

The master node detects the entire search space, and based on the number of slave nodes, population bounds are produced, the search space and bounds are equally segregated. Random individuals are generated within the given bounds

TABLE 1. Parameter setting for HMRS.

Parameters	Description	Value
$hist_{pro}$	Specifies the generation at which the optimization algorithm begins on the master node	1000 generations
DE_{run}	Specifies the count of generations for which the *DE* algorithm on the master node should run.	100 generations
CS_{thresh}	change strength threshold	0.03
$population_{age}$	Signifies the activation of the population aging mechanism	45 generations

(user-specified) until every sub-population reaches a predefined *NP*. Through this fixed region-based sub-population generating mechanism, the framework can cover the entire search space and explore promising areas during the optimization process. Further, the sub-population is distributed by the master node to all the slave nodes in the framework. Every slave node will concentrate on a specific search-space region and initiate the optimization process with the dispensed sub-population. A partial illustration displaying the evolution of the proposed algorithm to solve the *DOP* is presented in Figure 3.

Once the slave node initiates the optimization process, it directs the best candidates to the master node for every succeeding generation. The master node creates an archive (*Archive_1*) to store the best candidates received from all the slave nodes. Periodically, after certain generations ($hist_{pro}$), the master node starts processing the best candidates aggregated in the archive equipped with the *DE* algorithm. The number of runs the master node needs to run the *DE* algorithm is customized as (DE_{run}). The processed result is stored in *Archive_2*, the best candidate up to the current generation is held here, and the archive is cleared. The *Archive_2* is updated after every successive generation; it compares the results produced by the master node and all slave nodes. Similarly, after every generation, the master node monitors the success rate (*sr*) of each slave node in the framework to find the non-contributing sub-populations. The *sr* values are calculated using equation (2).

$$sr = \frac{number\ of\ improved\ individuals}{total\ number\ of\ individuals} \tag{2}$$

Detection of dynamic events plays a crucial role in solving *DOPs*; this work utilized a commonly pursued procedure to detect the change, the random test point method [47]. In this technique, individual candidates are randomly chosen from the search space. At each generation, the candidate's fitness is evaluated. The dynamic event will be detected if there is a change in the fitness of the chosen individual and the algorithm of the framework responds to it. Further, if a change

Algorithm 2 Population-aging mechanism

```
Begin
Generate a random integer m_rand ∈ (0.01, 0.10)
If (current_sub-population contains global best)
        Do not use aging mechanism.
Else if (age of current_sub-population > population_age & m_rand > 0.5)
        Re-initialize the current sub-population based on corresponding bounds.
End
```

FIGURE 4. Algorithm for the population-aging mechanism.

Algorithm 3 Working of the Master Node

```
Begin
Divide search space based upon the number of slave nodes to produce bounds
Bounds = bound_1{0, a1}, bound_2{a1, a2} ….. bound_n {an, an+1}
Generate sub-population until population size is met, with corresponding bounds
Send (sub-population to respective nodes)
Do while (Termination condition is not satisfied)
    Collect sr from all slave node
    Collect the best candidate from all slave nodes and store it in the archive
    If (hist_pro == 1000)
        Run (DE) for the data present in the archive for DE_run times
        Store result in Archive_2 and clear archive
    Archive_2 = compare (best candidate from slave nodes, result produced by running DE in master node)
# random_test_point_method returns true or false based on detection of change in the environment
    RTP = random_test_point_method( )
    If (RTP == True)
        If (C_s > CS_thresh):
            Reinitialize (non-contributing sub-population)
            Send (new sub-population to the corresponding node)
 End loop
```

FIGURE 5. The algorithm for working of the master node.

is detected, the framework calculates the change strength (C_s) as given in equation (3).

$$C_s = f\left(best_{before}\right) - f\left(best_{after}\right) \tag{3}$$

where $f\left(best_{before}\right)$ and $f\left(best_{after}\right)$ are the fitness of the best candidates before and after the change is detected. A pre-defined change strength threshold (CS_{thresh}) value is fixed. When the C_s value increases beyond the pre-defined CS_{thresh} value, the non-contributing sub-population is re-initialized with random individuals based on the corresponding bounds. The optimization process is re-initiated in the respective slave node.

In a dynamic environment, if the similarity between individuals in a population is low, the optimization process will likely stagnate at local optima. To address this limitation, at each sub-population level, the proposed strategy incorporates a mere population re-initialization strategy, which is the population-aging mechanism. A threshold value ($population_{age}$) is set in advance to ensure that a population aging mechanism should be initiated after a certain number of generations. A counter variable is customized to count the age of the sub-population in each slave node separately, and the variable is reinitialized once the population-aging mechanism is successful. The working of the population-aging mechanism is presented in Algorithm 2 (in Figure 4). The Algorithm 3 (in Figure 5) and Algorithm 4 (in Figure 6) depict the working of master and slave nodes, respectively. The parameter configuration for deploying *HMRS* is displayed in Table 1.

In the implementation part, the proposed *HMRS* was deployed using *mpi4py* library. *Mpi4py* serves as a python module that simplifies parallel programming by adhering

Algorithm 4 Working of the Slave Nodes

```
Begin
Population = Receive (sub-population from the master node, bounds)
Do while (Population.Length != 0)
    New Population = Check for population update        //sent from master node
    If (New Population is not Empty)
        Population = New Population
    Do while (Termination condition is not satisfied)
        Initiate the optimization process
            age+=1    #counter variable to count population age
            For (every generation)
                Send (best candidate to master node)
                Calculate (sr)
                Send (sr) to master node
                If (age of current_sub-population > population_age)
                    Check (respective conditions in Algorithm 2)
                        If true (reinitialize the current population)
                        Age=0
                End If
            End for loop;
    End For loop;
End
```

FIGURE 6. The algorithm for working of slave node.

to the Message Passing Interface (*MPI*) standard, allowing for efficient interaction and coordination among dispersed processes. This tool is particularly valuable for scientific and data-intensive tasks, where parallelization can significantly accelerate computational and analytical processes. In the realm of parallel computing, *mpi4py* establishes synchronization among nodes through the widely embraced *MPI* standard, which serves as a prevalent protocol for communication and coordination in distributed systems. Beyond its primary focus on message passing, *MPI* extends its capabilities to shared-memory functionality via one-sided communication and shared memory windows. These functionalities empower processes on separate nodes to synchronize their actions effectively by locking and accessing shared data structures. Here the core allocation has been done manually, while launching *MPI* job using command-line options.

Furthermore, *mpi4py* furnishes synchronization primitives such as mutexes and condition variables, affording precise control over synchronization within nodes. These primitives enable the orderly coordination of thread execution within a node, ensuring synchronization as required. This enables *HMRS* to harness the power of parallelism and efficiently address complex *DOPs*leveraging the strengths of both *mpi4py* and the *HMRS* strategy for improved performance and effectiveness in various computing environments.

Incorporating *HMRS* in *dDE* framework for optimization yields numerous benefits. Its parallel computing capabilities distribute the optimization process among multiple nodes, significantly expediting convergence and rendering it well-suited for extensive problem domains. The fault tolerance inherent in *dDE* ensures its continuous operation even in the event of node failures, enhancing robustness in distributed settings. Concurrent exploration of multiple sub-populations by *dDE* fosters diverse search trajectories, heightening the likelihood of discovering global optima. This adaptability extends *DOPs*, allowing *dDE* to tailor its strategies for evolving conditions. In addition, *dDE* effectively leverages distributed computing resources, rendering it suitable for tasks demanding substantial computational prowess. In essence, *dDE* streamlines computation time, positioning it as a valuable tool for intricate, large-scale, and time-critical optimization endeavours across diverse fields.

V. DESIGN OF EXPERIMENTS

This section discusses benchmarking suites, evaluation metrics used in the experiment, the parameter settings for the proposed algorithm, and the comparative analysis with State-of-the-Art methods.

A. BENCHMARKING SUITE

The Moving Peak Benchmarking (*MPB*) is a maximization problem proposed in [2] and [4] that was first proposed by Branke et al. is a dynamic and continuous optimization problem; by nature. The *MPB* problem was used to evaluate the

TABLE 2. Parameter setting for MPB.

Parameters	Description	Value
Max_FE_s	Maximum functional evaluation	$5e^5$
p	Number of peaks	(5, 10, 20)
Peak shape	Peak shape	Cone
cf	Change periodicity	5000
height_severity	Height severity	7.0
width_severity	Width severity	1.5
S	Shift length	1 and 5
D	Dimension	5
sr	Search range	[0, 100]
H	Height range	[30.0, 70.0]
W	Width range	[1.0, 12.0]
λ	Correlation Coefficient	0 and 1

performance of *HMRS*. Researchers around the globe widely use *MPB* because of its ease of implementation and high configurability. *MPB* comprises sets of peaks that change over a period of time (after specific functional evaluations). The return value of the benchmark represents the quality of the solution. The mathematical representation of the fitness function of *MPB* is expressed in equation (4).

$$F(x,t) = \max_{i=1,\ldots,p} \left(\frac{H_{i(t)}}{1 + w_i(t) \sum_{j=1}^{D} (x_j(t) - X_{ij}(t))^2} \right) \quad (4)$$

where, $F(x,t)$ – solution (x) quality at time t, p- number of peaks, D- search-space dimension, $H_{i(t)}$ defines the peak height, and $w_i(t)$ representing the peak width, describes the peak location of the j^{th} element. The height and width parameters follow a random Gaussian distribution (σ). The peak change performed dynamically by adding height and width severity, is mathematically expressed in equations (5) and (6):

$$H_i(t) = H_i(t-1) + height\_severity * \sigma \quad (5)$$

$$W_i(t) = W_i(t-1) + Width\_severity * \sigma \quad (6)$$

Based on problem severity, height and width severity are calculated. The positional change by velocity vector (v_i) is expressed in equation (7).

$$X_i(t) = X_i(t-1) + v_i(t) \quad (7)$$

The change frequency (cf) happens every 5000 functional evaluations (FE_s) [48]. The parameter settings of *MPB*used for the experiment are presented in Table 2.

B. EVALUATION METRICS

Two popularly used evaluation metrics in the *DOP* community were chosen to analyze the proposed strategy's performance.

(1) Offline error (OFF_e) and
(2) Best before change error (BB_{ce}) [48].

TABLE 3. Optimization algorithm settings, for DE/rand/1/bin.

Algorithm Settings
NP = 10
Scaling factor (F) = 0.2
Crossover rate (Cr) = 0.5
$hist_{pro}$ = 1000 generations
DE_{run} =100 generations
CS_{thresh} = 0.03
$population_{age}$ = 45 generations
Slave_nodes = 4

The Offline error is formulated in equation (8).

$$OFF_e = \frac{1}{C * N} \sum_{i=1}^{C} \sum_{j=1}^{N} E_{ij} \quad (8)$$

where C denotes the environmental change, Nrepresents Max_FE_s spent in each environment, and E_{ij} indicates the error of the j^{th} functional evaluation under the i^{th} environment. *Error*is the difference between the fitness of the best candidate identified by the strategy and the actual optimum solution in current environment. The OFF_e indicates the ability of the algorithm to respond when environmental change occurs.

The BB_{ce} indicates ability of the algorithm to perform the global search. It is mathematically expressed in equation (9):

$$BB_{ce} = \frac{1}{C} \sum_{i=1}^{C} E_{ij} \quad (9)$$

The combination of these two-performance metrics (OFF_e, BB_{ce}) is widely used to provide an evident performance measure for the algorithms that can solve *DOPs* efficiently.

C. EXPERIMENTAL SETTINGS

This work is a simulation of distributed system in a laptop with multiple processors. The experiments were conducted on a laptop (MacBook Pro- 2019) whose features included the following:

- graphics card (Intel Iris Plus Graphics 645 1536 MB)
- memory (8 GB 2133 MHz LPDDR3), and
- processor (1.4 GHz Core Intel Core i5 9^{th} gen)

As the proposed strategy relies upon a fixed region-based sub-population mechanism, the process of migration is not incorporated as the core. The parameter used for DE/rand/1/bin are listed in Table 3.

The *HMRS* was evaluated on different parameter combinations using the *MPB* test suite. Each parametric combination arises with varying environmental characteristics. To minimize the error that occurred from randomness, each instance was run 20 times. The environment change happened every 5000 FE_s; the termination condition was set to $5e^5$ FE_s. Therefore, there were 100 changes in the environment for every single instance. Standard deviation and the average

TABLE 4. Experimental results of HMRS against other methods with p = 5.

P	λ	S	Error	jDE	CESO	CPSO	Cloudde	HIDE	HMRS
5	0	1	OFF_e	37.48 (7.97)+	8.47 (3.02)+	3.86 (0.85)+	29.44 (5.49)+	2.48 (0.52)+	**1.84 (0.67)**
			BB_{ce}	9.34 (2.58)+	6.33 (3.22)+	1.38 (0.41)+	8.35 (2.22)+	0.33 (0.38)-	**0.68 (1.37)**
		5	OFF_e	37.42 (8.04)+	15.72 (4.72)+	9.16 (1.94)+	29.48 (5.60)+	5.63 (0.97)+	**1.68 (0.57)**
			BB_{ce}	9.26 (2.62)+	8.83 (3.54)+	2.04 (0.79)+	8.33 (2.32)+	0.50 (0.42)+	**0.34 (0.63)**
	1	1	OFF_e	37.14 (7.37)+	8.47 (2.94)+	3.93 (0.90)+	29.33 (5.82)+	2.48 (0.65)+	**1.52 (0.53)**
			BB_{ce}	9.34 (2.50)+	6.13 (3.13)+	1.38 (0.58)+	8.24 (2.36)+	0.34 (0.69)=	**0.33 (0.39)**
		5	OFF_e	37.26 (7.55)+	16.35 (4.25)+	9.28 (2.12)+	29.57 (6.36)+	5.68 (1.05)+	**1.72 (0.65)**
			BB_{ce}	9.28 (2.50)+	8.91 (3.13)+	2.32 (1.07)+	8.32 (2.65)+	0.56 (0.67)=	**0.55 (0.96)**

TABLE 5. Experimental results of HMRS against other methods with P = 10.

P	λ	S	Error	jDE	CESO	CPSO	Cloudde	HIDE	HMRS
10	0	1	OFF_e	34.30 (6.42)+	7.97 (2.53)+	4.17 (0.67)+	29.23 (5.65)+	2.68 (0.38)+	**1.76 (0.54)**
			BB_{ce}	10.86 (2.81)+	5.92 (2.39)+	1.90 (0.38)+	9.74 (2.41)+	0.74 (0.50)+	**0.34 (0.54)**
		5	OFF_e	34.16 (6.32)+	14.89 (3.42)+	8.37 (1.30)+	29.12 (5.44)+	5.69 (0.83)	**2.08 (0.84)**
			BB_{ce}	10.73 (2.58)+	9.79 (2.81)+	2.61 (0.68)+	9.75 (2.38)+	1.39 (0.68)+	**0.35 (0.46)**
	1	1	OFF_e	34.36 (6.34)+	8.83 (2.82)+	4.19 (0.73)+	28.92 (5.08)+	2.81 (0.69)+	**1.73 (1.10)**
			BB_{ce}	10.83 (2.64)+	6.75 (2.73)+	1.96 (0.61)+	9.66 (2.17)+	0.87 (0.80)+	**0.70 (1.06)**
		5	OFF_e	34.79 (6.21)+	15.31 (3.86)+	8.58 (1.54)+	29.70 (5.53)+	5.66 (0.95)+	**1.91 (0.62)**
			BB_{ce}	11.13 (2.56)+	9.89 (3.31)+	2.72 (0.75)+	9.99 (2.45)+	1.37 (0.90)+	**0.28 (0.36)**

TABLE 6. Experimental results of HMRS against other methods with P = 20.

P	λ	S	Error	jDE	CESO	CPSO	Cloudde	HIDE	HMRS
20	0	1	OFF_e	31.75 (2.95)+	9.32 (2.92)+	4.19 (0.54)+	27.57 (2.84)+	3.81 (1.01)+	**1.95 (1.03)**
			BB_{ce}	10.31 (1.35)+	7.52 (2.91)+	2.17 (0.41)+	9.41 (1.27)+	2.07 (0.92)+	**0.88 (1.52)**
		5	OFF_e	31.63 (3.13)+	14.07 (2.20)+	7.91 (0.85)+	28.06 (2.65)+	5.93 (0.70)+	**2.11 (0.97)**
			BB_{ce}	10.33 (1.45)+	9.87 (2.04)+	2.90 (0.62)+	9.59 (1.35)+	2.05 (0.55)+	**0.69 (1.66)**
	1	1	OFF_e	31.85 (2.80)+	9.83 (2.04)+	4.27 (0.54)+	27.84 (2.81)	3.81 (0.77)+	**2.15 (0.90)**
			BB_{ce}	10.36 (1.24)+	7.84 (1.85)+	2.23 (0.46)+	9.43 (1.30)+	1.92 (0.74)=	**1.36 (3.16)**
		5	OFF_e	32.10 (2.99)+	15.58 (3.01)+	7.96 (0.90)+	28.45 (2.97)+	5.84 (0.73)+	**2.05 (0.64)**
			BB_{ce}	10.55 (1.31)+	11.04 (2.75)+	2.88 (0.65)+	9.67 (1.41)+	2.00 (0.63)+	**0.41 (0.88)**

value of the error were taken as the comparison factors for comparison with other State-of-the-Art algorithms.

VI. RESULTS AND DISCUSSIONS

As additional measures for verification of the solution were put forth, the proposed approach was compared with the following State-of-the-Art methods - *jDE* [22], *CESO* [49], *CPSO* [20], and *Cloudde* [50]. The performance of the latter algorithms was inferred from recent work that has efficiently solved *DOP*, the *HIDE* [51] algorithm. To provide a fair empirical analysis, the proposed strategy parameters were set based on the parameter configuration of the *HIDE* algorithm. Tables 4, 5 and 6 depicts the proposed strategy's performance with other algorithms, where the number of peaks

TABLE 7. Statistical results – One sample t-test.

P	λ	S	Error	*HIDE*	*HMRS*	*t*-value	*SDE*	Significance
5	0	1	E_o	2.48 (0.52)	1.84 (0.67)	4.27	0.15	+
			E_b	0.33 (0.38)	0.68 (1.37)	1.142	0.30	=
		5	E_o	5.63 (0.97)	1.68 (0.57)	30.99	0.12	+
			E_b	0.50 (0.42)	0.34 (0.63)	1.13	0.14	=
	1	1	E_o	2.48 (0.65)	1.52 (0.53)	8.10	0.11	+
			E_b	0.34 (0.69)	0.33 (0.39)	0.11	0.08	=
		5	E_o	5.68 (1.05)	1.72 (0.65)	27.24	0.14	+
			E_b	0.56 (0.67)	0.55 (0.96)	0.04	0.21	=
10	0	1	E_o	2.68 (0.38)	1.76 (0.54)	7.61	0.12	+
			E_b	0.74 (0.50)	0.34 (0.54)	3.31	0.12	+
		5	E_o	5.69 (0.83)	2.08 (0.84)	19.21	0.18	+
			E_b	1.39 (0.68)	0.35 (0.46)	10.11	0.10	+
	1	1	E_o	2.81 (0.69)	1.73 (1.10)	4.39	0.24	+
			E_b	0.87 (0.80)	0.70 (1.06)	0.71	0.23	=
		5	E_o	5.66 (0.95)	1.91 (0.62)	27.04	0.13	+
			E_b	1.37 (0.90)	0.28 (0.36)	13.54	0.08	+
20	0	1	E_o	3.81 (1.01)	1.95 (1.03)	8.07	0.23	+
			E_b	2.07 (0.92)	0.88 (1.52)	3.50	0.34	+
		5	E_o	5.93 (0.70)	2.11 (0.97)	17.61	0.21	+
			E_b	2.05 (0.55)	0.69 (1.66)	3.66	0.37	+
	1	1	E_o	3.81 (0.77)	2.15 (0.90)	8.24	0.20	+
			E_b	1.92 (0.74)	1.36 (3.16)	0.79	0.70	=
		5	E_o	5.84 (0.73)	2.05 (0.64)	26.48	0.14	+
			E_b	2.00 (0.63)	0.41 (0.88)	8.08	0.19	+

P (5, 10, 20) changes with corresponding tables. E_o and E_b in the Table 7 represents offline errors and best before change error.

The experimental results in Table 4, 5 and 6 summarize *HMRS* performance in terms of offline and best before change errors. The value inside the parenthesis indicates the standard deviation of 20 runs. The best results of the corresponding instance are highlighted in bold font. "+," "−," and "=" symbols in Tables 4, 5 and 6 represent the *HMRS*performance in comparison to the State-of-the-Art methods as better, worse, and equals to, respectively. These results affirm that the *dDE* framework outfitted with

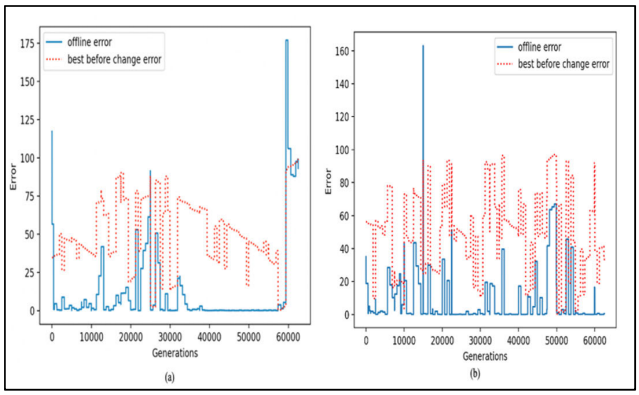

FIGURE 7. Exemplification of offline error and best before change error with instances chosen randomly. a) Peaks (P) = 5 and b) Peaks (P) = 20.

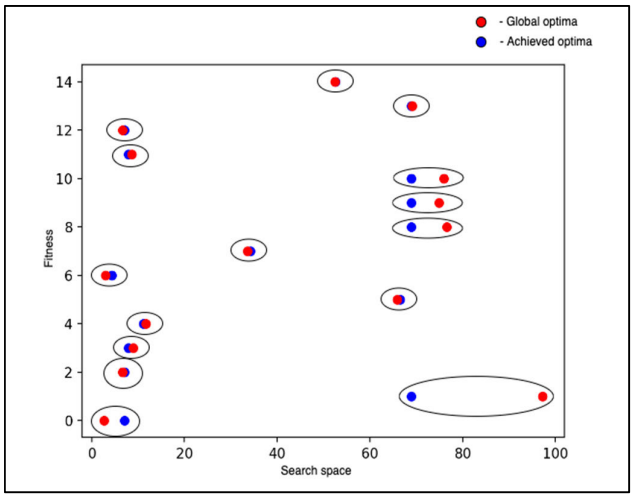

FIGURE 8. Illustrating the searchability of the HMRS after the change is detected.

HMRS is more likely to find global optima under dynamic environments. The proposed strategy has outperformed other comparable algorithms in 12 *MPB* test cases, which testifies the efficacy of the proposed algorithm. The results unequivocally highlight *HMRS's* consistent superiority over the other five algorithms in various configurations when tackling *MPB* problems with different degrees of shift severity. As shift lengths increase, tracking peaks becomes progressively more challenging, causing the performance of all algorithms to naturally deteriorate. However, *HMRS* demonstrates minimal sensitivity to elongating shift lengths in terms of OFF_e compared to the other algorithms, showcasing its remarkable resilience in detecting and tracking multiple optimal points even in dynamically changing environments.

Furthermore, *HMRS* maintains its stability when faced with an increasing number of peaks, a situation that typically poses a greater challenge for algorithms to converge effectively to optimal solutions. This intriguing phenomenon can be attributed to the emergence of numerous local optima with heights resembling that of the global optimum as the number of peaks grows. Consequently, *HMRS* have a higher probability of identifying these relatively superior local optima, contributing to its consistent performance even as the number of peaks expands.

In direct comparisons, *HMRS* consistently achieves significantly lower OFF_e and BB_{ce} compared to all other algorithms, even when the number of peaks increases. The results of *HIDE*algorithms slightly lag behind *HMRS* as the number of peaks rises. Nevertheless, it still outperforms other state-of-the-art algorithms. Additionally, *HMRS* exhibits substantial performance gains over *HIDE* when the change frequency is increased.

Results of a Statistical study focusing on *HIDE* versus *HMRS* are presented in Table 7. The t-value represents the ratio of the difference between the mean of two sample sets, and *SDE* represents the standard error of the difference. The "+" and "=" in Table 7 represents whether the proposed *HMRS* is statistically significant or insignificant. The proposed strategy outperforms the *HIDE* algorithm in many instances. This detailed empirical and statistical analysis concludes that the *HMRS* solves *DOPs* efficiently.

Figures 7. a and 7. b present the convergence graph of the proposed strategy with a change in the number of peaks P (5, 20). From Figure 7, it is evident that *HMRS* has demonstrated high convergence capability. Due to random population initialization, the proposed strategy has lost track of global optima periodically, but it is capable of recovering of the global optima in the promising region. Further to underscore the significance of the proposed strategy statistically, a sample t-test was also performed. Among the State-of-the-Art algorithms, the *HIDE*performed significantly better than other algorithms.

An illustration representing the searchability of the proposed algorithm, concerning 15 changes in single dimension search space with random parameter settings, is presented in Figure 8. And it is clear that during every environmental change, the proposed *HMRS*finds the global optima efficiently.

VII. CONCLUSION

This study incorporated a *HMRS* within the *dDE* framework to tackle multimodal dynamic optimization problems. The proposed strategy consistently achieved high population diversity through population re-initialization mechanisms. Incorporation of the proffered scheme into the *dDE* framework further balances the local and global search routines in a dynamic environment. Manipulation of historical information that emanates during the optimization process aids the strategy to achieve effective results. The experiments were conducted in the *MPB* problem, a commonly used benchmark suite. The empirical and statistical analysis proves that the *HMRS* outperforms the State-of-the-Art methods under different parameter settings.

HMRS, serves practical purposes across diverse domains. For instance, it can enhance aircraft design by optimizing wing profiles and engine configurations, resulting in improved fuel efficiency and overall performance within the aviation sector. In realm of healthcare, *HMRS* proves

valuable by optimizing resource allocation, encompassing staff scheduling and equipment utilization. This ensures cost-effective patient care and operational efficiency. Additionally, *HMRS* can enhance renewable energy grids by optimizing the integration of sources such as wind and solar power, maximizing energy output and grid stability, and ultimately diminishing reliance on fossil fuels. These examples emphasize the adaptable and impactful role of *HMRS* in practical applications.

Despite *HMRS* offers several advantages, it also presents limitations that require careful consideration. These include the computational overhead associated with managing multiple populations and periodic reinitialization. In addition based on the targeted optimization task, adaptive parameter tuning is necessary to achieve comparable results. And the algorithm fails to detect change when exposed to noisy environment. Addressing these limitations is essential to harness *HMRS* effectively for diverse optimization tasks.

The future research direction focuses on incorporating multi-threaded architecture and adaptive migration topology. Evaluation of the framework's performance with real-world dynamic optimization problems is one of the significant directions.

REFERENCES

[1] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*, vol. 178. Berlin, Germany: Springer, 2004.

[2] J. Branke, *Evolutionary Optimization in Dynamic Environments*, vol. 3. Berlin, Germany: Springer, 2012.

[3] K. Weicker, *Evolutionary Algorithms and Dynamic Optimization Problems*. Berlin, Germany: Der Andere Verlag, 2003.

[4] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3, 1999, pp. 1875–1882.

[5] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," Nav. Res. Lab., Washington, DC, USA, Tech. Rep. 6760, 1990.

[6] 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.

[7] J. Brest, V. Zumer, and M. S. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 215–222.

[8] K. Krishnakumar, "Micro-genetic algorithms for stationary and nonstationary function optimization," *Proc. SPIE*, vol. 1196, Feb. 1990, Art. no. 969927.

[9] A. Baykasoğlu and F. B. Ozsoydan, "An improved firefly algorithm for solving dynamic multidimensional knapsack problems," *Exp. Syst. Appl.*, vol. 41, no. 8, pp. 3712–3725, Jun. 2014.

[10] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, May 2002, pp. 1666–1670.

[11] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: A survey on problems, methods and measures," *Soft Comput.*, vol. 15, no. 7, pp. 1427–1448, Jul. 2011.

[12] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1–24, Oct. 2012.

[13] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: A genetic algorithm approach," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3, 1999, pp. 2031–2038.

[14] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Proc. Workshops Appl. Evol. Comput.* Berlin, Germany: Springer, 2004.

[15] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: A forward-looking approach," in *Proc. 8th Annu. Conf. Genetic Evol. Comput.*, 2006, pp. 1–8.

[16] S. Yang, "On the design of diploid genetic algorithms for problem optimization in dynamic environments," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 1362–1369.

[17] M. Daneshyari and G. G. Yen, "Dynamic optimization using culturally based PSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2011, pp. 509–516.

[18] R. Mendes and A. S. Mohais, "DynDE: A differential evolution for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, Sep. 2005, pp. 2808–2815.

[19] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," in *Proc. 4th Int. Conf. Natural Comput.*, vol. 7, 2008, pp. 624–628.

[20] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959–974, Dec. 2010.

[21] 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.

[22] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "Dynamic optimization using self-adaptive differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 415–422.

[23] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 881–897, Jun. 2013.

[24] 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.

[25] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proc. Congr. Evol. Comput.*, vol. 1, 2004, pp. 98–103.

[26] H. Zhang, J. Zhao, H. Leung, and W. Wang, "Multi-stage dynamic optimization method for long-term planning of the concentrate ingredient in copper industry," *Inf. Sci.*, vol. 605, pp. 333–350, Aug. 2022.

[27] Z. Li, C. Wang, J. Zhang, M. Zeng, P. Xu, Z. Song, and N. Dong, "Two-stage dynamic optimization on station-to-door delivery with uncertain freight operation time in urban logistics," *J. Urban Planning Develop.*, vol. 148, no. 3, p. 853, Sep. 2022.

[28] L. Abualigah and K. H. Almotairi, "Dynamic evolutionary data and text document clustering approach using improved Aquila optimizer based arithmetic optimization algorithm and differential evolution," *Neural Comput. Appl.*, vol. 34, pp. 20939–20971, Aug. 2022.

[29] S. Raghul and G. Jeyakumar, "Investigations on distributed differential evolution framework with fault tolerance mechanisms," in *Differential Evolution: From Theory to Practice*. Singapore: Springer, 2022, pp. 175–196.

[30] S. Raghul and G. Jeyakumar, "Investigations on performance of multi-threaded distributed evolutionary algorithmic framework in image processing applications," *Int. J. Intell. Eng. Syst.*, vol. 16, no. 5, pp. 649–665, 2023.

[31] V. S. Sree and S. Thangavelu, "Performance analysis of differential evolution algorithm variants in solving image segmentation," in *Computational Vision and Bio-Inspired Computing*. New York, NY, USA: Springer, 2020.

[32] R. Storn, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep. 95-012, 1995.

[33] D. M. Dhanalakshmy, G. Jeyakumar, and C. S. Velayutham, "Empirical investigations on evolution strategies to self-adapt the mutation and crossover parameters of differential evolution algorithm," *Int. J. Intell. Syst. Technol. Appl.*, vol. 20, no. 2, pp. 103–125, 2021.

[34] D. M. Dhanalakshmy, P. Pranav, and G. Jeyakumar, "A survey on adaptation strategies for mutation and crossover rates of differential evolution algorithm," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 6, no. 5, pp. 613–623, 2016.

[35] L. Deng, C. Li, Y. Lan, G. Sun, and C. Shang, "Differential evolution with dynamic combination based mutation operator and two-level parameter adaptation strategy," *Exp. Syst. Appl.*, vol. 192, Apr. 2022, Art. no. 116298.

[36] M. Wang, Y. Ma, and P. Wang, "Parameter and strategy adaptive differential evolution algorithm based on accompanying evolution," *Inf. Sci.*, vol. 607, pp. 1136–1157, Aug. 2022.

[37] 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.
[38] Aswani, V. V. Praveen, and S. Thangavelu, "Performance analysis of variants of differential evolution on multi-objective optimization problems," *Indian J. Sci. Technol.*, vol. 8, no. 17, pp. 1–6, Aug. 2015.
[39] S. Thangavelu and C. S. Velayutham, "An investigation on mixing heterogeneous differential evolution variants in a distributed framework," *Int. J. Bio-Inspired Comput.*, vol. 7, no. 5, pp. 307–320, 2015.
[40] S. S. Thangavelu and C. S. Velayutham, "Combining different differential evolution variants in an island based distributed framework—An investigation," in *Advances in Intelligent Informatics*. Cham, Switzerland: Springer, 2015, pp. 593–606.
[41] G. Jeyakumar and C. S. Velayutham, "Distributed mixed variant differential evolution algorithms for unconstrained global optimization," *Memetic Comput.*, vol. 5, no. 4, pp. 275–293, Dec. 2013.
[42] G. Jeyakumar and C. S. Velayutham, "Distributed heterogeneous mixing of differential and dynamic differential evolution variants for unconstrained global optimization," *Soft Comput.*, vol. 18, no. 10, pp. 1949–1965, Oct. 2014.
[43] M. T. Indu and C. S. Velayutham, "A meta-evolutionary selection of constituents in ensemble differential evolution algorithm," *Exp. Syst. Appl.*, vol. 205, Nov. 2022, Art. no. 117667.
[44] M. T. Indu and C. S. Velayutham, "Towards grammatical evolution-based automated design of differential evolution algorithm," in *Proc. Congr. Intell. Syst.* Singapore: Springer, 2020, pp. 329–340.
[45] R. Murali and C. S. Velayutham, "Adapting novelty towards generating antigens for antivirus systems," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2022, pp. 1254–1262.
[46] M. Tomassini, *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time*. Berlin, Germany: Springer, 2006.
[47] K. Deb, "Single and multi-objective dynamic optimization: Two tales from an evolutionary perspective," Dept. Mech. Eng., Indian Inst. Technol., India, KanGAL Rep. no. 2011004, 2011.
[48] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*. Berlin, Germany: Springer, 2003.
[49] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 564–567.
[50] Y.-X. Li, Z.-H. Zhan, H. Jin, and J. Zhang, "Cloudde-based distributed differential evolution for solving dynamic optimization problems," in *Proc. 10th Int. Conf. Intell. Control Inf. Process. (ICICIP)*, Dec. 2019, pp. 94–99.
[51] S.-H. Wu, K.-J. Du, Z.-H. Zhan, H. Wang, and J. Zhang, "Historical information-based differential evolution for dynamic optimization problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 119–126.

S. RAGHUL received the B.Tech. and M.Tech. degrees in computer science and engineering from Amrita University, Coimbatore, in 2019 and 2021, respectively. He is currently a full-time Ph.D. Scholar with the Department of CSE, Amrita School of Computing, Amrita Vishwa Vidyapeetham University, Coimbatore. He has one year of experience as a teaching assistant and delivered lectures on data visualization for M.Sc. students with Amrita University. He has two international conference papers and a book chapter in his publication, which are indexed in Scopus. He is currently proposing a new framework that effectively solves dynamic optimization problems. His area of research interests include evolutionary algorithms and their applications, parallel and distributed computing, and machine learning. His M.Tech. thesis received the Outstanding Thesis for 2021 by the Department of CSE, Amrita School of Engineering, Coimbatore.

G. JEYAKUMAR received the B.Sc. degree in mathematics and the M.C.A degree (under the faculty of Engineering) from Bharathidasan University, in 1994 and 1998, respectively, and the Ph.D. degree in distributed differential evolution algorithm from Amrita Vishwa Vidyapeetham University, Tamil Nadu, India, in 2013.

He has been a Professor with the Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham University, Coimbatore Campus, since 2000. He has published numerous papers in reputed journals and conference proceedings, out of which majority of the publications are indexed in SCOPUS. He received the best paper awards for few of his publications. He has guided many student projects/thesis belong to the courses B.Tech., M.Tech. (CSE), M.Tech. (automotive engineering), M.C.A., and M.Phil.; and currently guiding Ph.D. scholars, UG, and PG students. His research interests include design and applications of evolutionary algorithms, soft computing approaches, and parallel/distributed computing.