A Multi-stage Parallel Differential Evolution for Global Optimization

Mohamed Saad [1], Hegazy Zaher [2], Naglaa Ragaa [1] and Heba Sayed [1]

[1] Department of Operations Research and Management, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza, Egypt.

[2] Department of Mathematical Statistics, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza, Egypt.

Abstract- Parallel optimization offers faster and more efficient problem-solving by reducing computational resource usage and execution time. By integrating multiple techniques such as evolutionary algorithms and swarm-based optimization, it enables more effective exploration of the search space and facilitates the attainment of optimal solutions within shorter time frames. Differential Evolution (DE) is a powerful and relatively recent evolutionary algorithm. However, its performance is often limited because most applications rely on a single mutation operator with fixed parameter values. To address this limitation, this study proposes a parallel framework that executes three DE algorithms simultaneously, with dynamic selection among three mutation strategies. Each algorithm runs independently on separate computational units, and the best solution identified is shared across units to accelerate convergence and improve efficiency. In the proposed approach, a mutation pool consisting of three mutation operators and a parameter pool with three predefined values are established. During evolution, mutation operators and parameter values are randomly selected from these pools to generate trial vectors, allowing the algorithm to exploit the complementary strengths of different strategies. The effectiveness of the proposed algorithm was evaluated on 24 widely used benchmark functions. Experimental results demonstrate significant improvements in both convergence speed and solution quality compared with traditional DE and non-DE algorithms. These findings indicate that the proposed parallel DE framework is highly competitive and provides a promising direction for solving complex optimization problems.

Keywords: optimization; global optimization; differential evolution; evolutionary algorithm; mutation operator; Parameter pool; parallel techniques; termination rules.

I. Introduction

The problem of finding the global minimum of a multidimensional function occurs in various scientific areas and has wide applications. In this context, the programmer seeks the absolute minimum of a function subject to some assumptions or constraints. The objective function is defined as:

$f : S \rightarrow R, S \subset R^n$ is expressed as:

$$x^* = \arg \min_{x \epsilon S} f(x) \quad (1)$$

Where the set S is defined as:

$$S = [a_1, b_1] \otimes [a_2, b_2] \otimes \dots [a_n, b_n]$$

A series of methods have been proposed in the recent literature to handle problems described by Equation (1). That usually divided into two categories: deterministic and stochastic methods. In the first category, the most common method is the interval method [1, 2], where the set S is divided through a series of steps into sub-regions and some sub-regions that do not contain the global solution can be removed using some pre-defined criteria. The second category contains stochastic techniques that do not guarantee finding the total minimum, and they are easier to program since they do not rely on some assumptions about the objective function. In addition, they also constitute the vast majority of global optimization methods. Among them, there are methods based on considerations derived from Physics, such as the Simulated Annealing method [3], the Henry's Gas Solubility Optimization (HGSO) [4], the Gravitational Search Algorithm (GSA) [5], the Small World Optimization Algorithm (SWOA) [6], etc. Also, Global optimization problems arise in diverse scientific and engineering domains and have inspired the development of numerous evolutionary-based techniques. Representative examples include Genetic Algorithms (GA) [7], Differential Evolution (DE) [8, 9], Particle Swarm Optimization

(PSO) [10, 11], Ant Colony Optimization (ACO) [12], Bat Algorithm (BA) [13], Whale Optimization Algorithm (WOA) [14], and Grasshopper Optimization Algorithm (GOA) [15]. These methods have demonstrated effectiveness in a wide range of applications. For instance, in physics, genetic algorithms have been successfully applied to locating particle positions in magnetic plasmas and developing optimization tools [16]. Furthermore, the combined application of different optimization methods has been shown to improve both stability and performance in complex problem settings [17]. Despite their effectiveness, most evolutionary algorithms demand substantial computational resources and execution time, which can hinder their practical applicability to large-scale or real-time problems. To overcome these limitations, parallel optimization has emerged as a powerful strategy. By distributing computations across multiple processing units, parallel optimization accelerates convergence and enhances solution quality. Applications of parallel optimization span a variety of domains, such as machine learning parameter tuning, control design, and engineering optimization [18]. For example, the parallel implementation of a Genetic Algorithm with a Fair Competitive Strategy (HFCGA) has been employed to design an optimized cascade controller for ball-and-beam systems, achieving better performance than traditional methods while avoiding premature convergence [19]. The advantages of parallel optimization extend beyond speed. By enabling multiple algorithms to run simultaneously and exchange information, parallel frameworks promote better exploration of the search space and improved robustness against local optima. They also facilitate effective error handling and allow the use of more complex models through increased computational capacity [20]. However, realizing these benefits requires careful design of workload distribution strategies and efficient mechanisms for result collection and evaluation. Among the various evolutionary algorithms, Differential Evolution (DE) has gained prominence for its simplicity and effectiveness. Introduced by Storn and Price [21], DE has proven particularly successful in addressing global and continuous optimization problems. The algorithm begins by initializing a population of candidate solutions in the search space, which are then evaluated using a fitness function. Offspring are generated through mutation and crossover operations, and a greedy selection mechanism ensures that superior solutions replace inferior ones [22, 23]. The success of DE, however, heavily depends on the choice of mutation strategies and parameter control methods, which can limit its performance in challenging optimization tasks. To address these challenges, this paper proposes a new optimization framework based on the parallel execution of multiple DE algorithms across different computational units. Each algorithm operates independently, and the best solutions discovered are periodically shared among units using novel propagation techniques. To further improve efficiency, intelligent termination criteria based on stochastic observations are incorporated and adapted to the parallel computing environment. In addition, the framework draws on insights from parallel computing strategies commonly applied in computational fluid dynamics (CFD), leveraging tools such as OpenMP, MPI, and CUDA [24] to reduce execution time and enhance scalability. The following sections are organized as follows: In Section 2, the method description and the parallel DE algorithm are briefly introduced. Section 3 provides the test functions. In Section 4, the experiments and performance evaluation are implemented. Finally, in Section 5, the conclusions from the current work are discussed.

II. METHOD DESCRIPTION

The Differential Evolution Method

The DE method relies on differential operators and is particularly effective in optimization problems that involve searching through a continuous search space. By employing differential operators, DE generates new solutions that are then evaluated and adjusted until the optimal solution is achieved [25]. The method was used in a series of problems, such as electromagnetics [26], energy consumption problems [27], job shop scheduling [28], image segmentation [29], etc. The Differential Evolution operates through the following steps: Initially, initialization takes place with a random population of solutions in the search space. Then, each solution is evaluated based on the objective function. Subsequently, a process is iterated involving the generation of new solutions through modifications, evaluation of these new solutions, and selection of the best ones for the next generation. The algorithm terminates when a termination criterion is met, such as achieving sufficient improvement in performance or exhausting the number of iterations.

Mutation Strategies

In stage of mutation, the vectors of the population solutions are mutated using mutation strategies at each iteration, the mutation operator is employed for each target vector to yield corresponding mutant vector $v_{i,g}$, the most common mutation strategies are found in Deng et al.[30] showed as follows:

(1) DE/rand/1

$$v_{i,g} = x_{r1,g} + F.\left(x_{r2,g} - x_{r3,g}\right) \tag{2}$$

(2) DE/best/1

$$v_{i,g} = x_{best,g} + F.\left(x_{r1,g} - x_{r2,g}\right) \tag{3}$$

(3) DE/current-to-best/1

$$v_{i,g} = x_{i,g} + F.\left(x_{best,g} - x_{i,g}\right) + F.\left(x_{r1,g} - x_{r2,g}\right) \tag{4}$$

(4) DE/rand/2

$$v_{i,g} = x_{r1,g} + F.\left(x_{r2,g} - x_{r3,g}\right) + F.\left(x_{r4,g} - x_{r5,g}\right) \tag{5}$$

(5) DE/best/2

$$v_{i,g} = x_{best,g} + F.\left(x_{r1,g} - x_{r2,g}\right) + F.\left(x_{r3,g} - x_{r4,g}\right) \tag{6}$$

(6) DE/rand-to-best/1

$$v_{i,g} = x_{3,g} + K.\left(x_{best,g} - x_{3,g}\right) + F.\left(x_{r1,g} - x_{r2,g}\right) \tag{7}$$

Where the indices r_1, r_2, r_3, r_4 *and* $r_5 \in \{1, 2, \cdot\cdot\cdot, \text{NP}\}$ are mutually exclusive integers that randomly chosen and should keep different from each other, which means that $r_1 \neq r_2 \neq r_3 \neq r_i$. The vector $x_{best,g}$ represents the optimal individual in the population at the *gth* generation, the vector $x_{i,g}$ represents target vector.

Propagation Mechanism

In the proposed PDEmp method, the algorithm uses a (1-to-1) propagation mechanism. Generally, a random island will send its best value to another randomly selected island. i.e., the best values of the islands are spread to the rest by replacing their worst values, To investigate the potential of this in various combinations in many islands have been tested on a set of 24 benchmark problems, this technique is shown in Fig. 1.

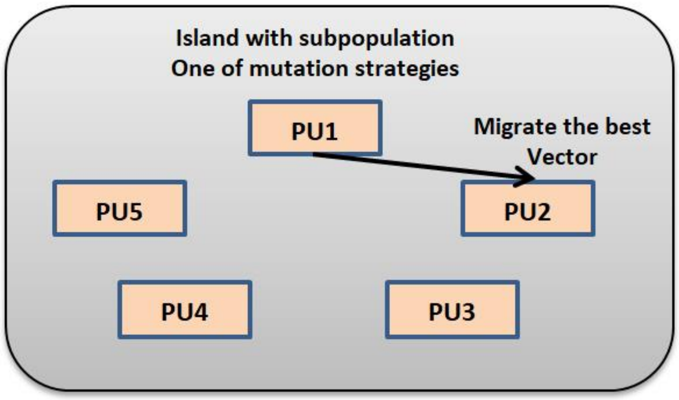

Fig. 1. (1-to-1) Propagation technique.

Mutation Pool of the Proposed Algorithm

Mutation operators and parameter settings significantly influence DE's performance. To exploit their complementary strengths, several researchers have pointed out that different mutation operators can result in better performances [31, 32]. The proposed PDEmp framework incorporates a mutation pool comprising three categories: the first is Best-guided strategy: DE/rand-to-best/1, which uses the current best solution. The second is Random strategy: DE/rand/2, which promotes global exploration. The third is Intermediate strategy: DE/current-to-rand/1, which provides robustness for rotated problems [33]. During evolution, one operator is randomly selected from the pool for application in a given DE instance. The control parameters F (mutation scaling factor) and CR (crossover probability) are also drawn from predefined sets. Larger values of F promote exploration across the search space, whereas smaller values accelerate convergence by focusing on local neighborhoods. Similarly, higher values of CR increase population diversity, while lower values are better suited for separable problems [32]. By combining different strategies and parameter values, PDEmp achieves a balance between exploration and exploitation [31, 32].

The Framework of the Proposed Algorithm

Parallel differential evolution algorithm with mutation pool (PDEmp) is presented, which implements the algorithms with advanced features, such as an enhanced stopping rule and advanced mutation schemes. The software allows for coding the objective function in C++ and has been tested on well-known benchmark functions, showing promising results. This work compares the performance of the proposed method with other parallel optimization methods. The pseudocode of the PDEmp is presented in Algorithm 1. The mutation strategies divide into three categories according to their features. DE/rand-to-best/1 belongs to the first category that has the best individual found heretofore; DE/rand/2 is the second category that perform random search, and the DE/current-to-rand/1 belongs to the last category. It is very difficult to decide which mutation strategy is better in each category. Based on the observations and motivation, a mutation pool is designed. In the evolution, one mutation strategy is randomly chosen for each algorithm, and three associated parameters are chosen simultaneously. Based on the mutation pool, a variant DE is designed, in which three mutation strategies and parameters are used in a single iteration. So, the proposed algorithm is defined as PDEmp. The performance of the PDEmp is estimated on a set of benchmark functions. The flowchart of the proposed algorithm is shown in Fig. 2, a parallel Differential Evolution technique is proposed that utilizes the mutation pool in order to random choosing of operators, each algorithm with its own independent termination criteria as follows:

The Termination Rule

In the base Differential Evolution algorithm, termination occurs after reaching a predefined number of iterations, which can sometimes result in premature halting before the global minimum is identified. In contrast, the difference in the proposed method is calculated as follows:

$$\delta_i^{(k)} = \left| f_{i,\min}^{(k)} - f_{i,\min}^{(k-1)} \right|, \qquad (8)$$

In this equation, $f_{i,\min}^{(k)}$ represents the best function value found for island i iteration k. If $\delta_i^{(k)} \leq \epsilon$ for a specified number of iterations, then the population evolution for the island is terminated. Furthermore, in the proposed PDEmp method, if this condition holds for more than one island, the entire algorithm will terminate.

In the proposed Parallel Differential Evolution with Mutation Pool (PDEmp) algorithm, the population is divided into N independent subgroups, referred to as islands (Fig. 2). Each island performs the differential evolution process independently. For example, if ten agents are distributed across two islands, agents 1–5 are assigned to Island 1, while agents 6–10 belong to Island 2. This island-based model enhances parallelism and allows efficient utilization of available computational resources. The PDEmp algorithm integrates both the mutation pool strategy and the island model with a dedicated termination mechanism. The termination mechanism serves two main purposes: to accelerate convergence by monitoring the progress of each island, and to avoid unnecessary computations once no significant improvement is observed. At the beginning of the process, three

DE algorithms are distributed algorithms across N_I computational threads, where $N_I > 3$. Each thread independently executes a DE algorithm. During each iteration, every computational unit identifies its best solution and communicates this to other units, replacing their worst-performing solutions. This exchange ensures a balanced distribution of search efforts and promotes faster convergence. When the number of threads is equal to, or an integer multiple of, the number of DE variants, the distribution is fully equivalent across units. In cases where equivalence is not exact, the degree of balance improves as the number of threads increases. The termination criterion is designed to adapt to continuous optimization tasks. Specifically, the algorithm evaluates the progress of the search at each iteration. If predefined stopping conditions are satisfied, such as stagnation in fitness improvement or reaching a maximum number of iterations, the corresponding thread halts execution. This adaptive termination rule minimizes redundant computations while ensuring that computational resources are efficiently allocated to active search processes. By combining parallel execution, inter-island communication, and adaptive termination, PDEmp achieves improved convergence speed and higher solution quality compared with traditional DE methods. The termination rule thus plays a central role in balancing efficiency with robustness, ensuring that the algorithm achieves competitive performance while avoiding unnecessary resource consumption.

Algorithm 1 The proposed overall algorithm

```
1.  Set as $N_I$ the total number of parallel processing units.
2.  Set as $N_k$ the total number of allowed iterations.
3.  Set $k = 0$ the iteration number.
4.  For $j = 1, . . . , N_I$ do in parallel
    (a) Execute an iteration of differential evolution algorithm as follow:
```

```
1. INPUT:
    (a) The population size $N_d \geq 4$. The members of this population are
        also called agents.
    (b) The crossover probability CR $\in [0, 1]$.
    (c) The differential weight $F \in [0, 2]$.
2. OUTPUT:
    (a) The agent $x_{\text{best}}$ with the lowest function value $f(x_{\text{best}})$.
3. Initialize all agents in $S$.
4. While termination criteria are not met do
    (a) Randomly select a mutation operator from DE/rand-to-best/1, DE/rand/2
        and DE/current-to- rand /1.
    (b) For $i = 1 \ldots N_d$ do
    i. Select as $x$ the agent $i$.
    ii. Select randomly three agents $a, b, c$ with the property $a \neq b,\ b \neq c,\ c \neq a$.
    iii. Select a random position $R \in \{1, \ldots, n\}$
    iv. Create the vector $y = [y_1, y_2, \ldots, y_n]$ with the following procedure
    v. For $j = 1, \ldots, n$ do
        A. Set $r_i \in [0, 1]$ a random number.
        B. If $r_j <$ CR or $j = R$ then $y_j = a_j + F \times (b_j - c_j)$ else $y_j = x_j$.
    vi. If $y \in S$ AND $f(y) \leq f(x)$ then $x = y$.
    vii. EndFor
    (c) EndFor
5. End While
```

(b) **Find** the best element from all optimization methods and **propagate** it to the rest of processing units.

5. **End For**
6. **Update** $k = k + 1$
7. **Check** the proposed termination rule. If the termination rule is valid, then go to step 7a else go to step 4.

(a) **Terminate** and report the best value from all processing units.

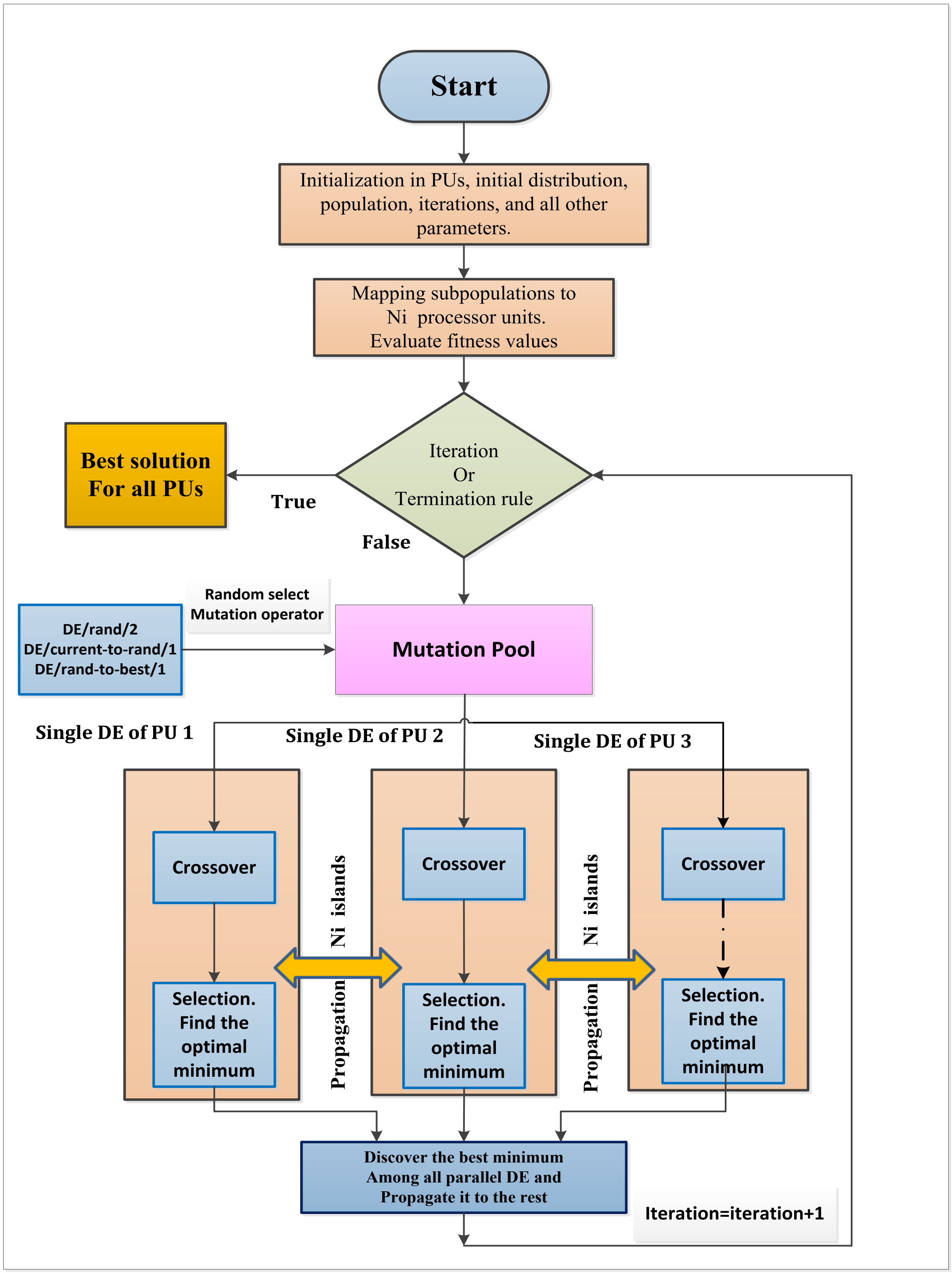
Start
Initialization in PUs, initial distribution, population, iterations, and all other parameters.
Mapping subpopulations to Ni processor units. Evaluate fitness values
Iteration Or Termination rule
True
False
Best solution For all PUs
Random select Mutation operator
DE/rand/2
DE/current-to-rand/1
DE/rand-to-best/1
Mutation Pool
Single DE of PU 1
Single DE of PU 2
Single DE of PU 3
Crossover
Selection. Find the optimal minimum
Ni islands
Propagation
Discover the best minimum Among all parallel DE and Propagate it to the rest
Iteration=iteration+1

Fig. 2. Flowchart of the overall process, The DE acronym stands for the Differential Evolution method, the PU acronym represents the Parallel Unit that executes the algorithm.

III. TEST FUNCTIONS

The benchmark functions used in the experiments have a fairly complex structure, and some of them have a large number of dimensions that make them perfect for studying, testing and comparing the results with the modified differential evolution method of Tsoulos, I.G. [34].

Benchmark functions To evaluate the effectiveness of the suggested parallel PDEmp algorithm in locating the global minimum of functions, a set of test functions that are used here is selected [35, 36]. The selected functions are as follows:

- **BF1** function (Bohachevsky1) is defined as follows:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

  With $x \in [-\ 100, 100]^2$

- **BF2** function (Bohachevsky1) is defined as follows:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\ \cos(4\pi x_2) + \frac{3}{10}$$

  With $x \in [-\ 50, 50]^2$

- **BRANIN** function is given by:

$$f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

  At range $-5 \le x_1 \le 10$, $0 \le x_2 \le 15.$

- **CM** (Cosine Mixture) function

$$f(x) = 0.1 * \sum_{i=1}^{n}\cos(5\pi x_i) - \sum_{i=1}^{n} x_i^2$$

  With $x \in [-\ 1, 1]^n$

- **Camel back** function. Six Hump The function is given by:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

  At range $-5 \le x_i \le 5.$

- **Easom** function is given by:

$$f(x) = -cos(x_1)\, cos(x_2)\, e^{(-((x_2-\pi)^2+(x_1-\pi)^2))}$$

  With $x \in [-\ 100, 100]^2$

- **EXPONENTIAL** function. The function is given by the following:

$$f(x) = -\ exp\left(-0.5\sum_{i=1}^{n} x_i^2\right),$$

  At range $-1 \le x_i \le 1$. in the experimentsthe function used with $n = 4$. And the

  Corresponding function is denoted by EXP16.

- **GKLS** function. The function is given by the following[37]:

  F (x) = GKLS (*x, n, w*), *n* = 2, *w* = 50,

  With $x \in [-1, 1]^2$, n is a positive integer between 2 and 100.

- **GRIEWANK** function. The function is given by the following:

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}},$$

  With $x \in [-100, 100]^2$

- **Hansen** function. The function is given by the following:

$$f(x) = \left(\sum\nolimits_{i=1}^{5}(i\,cos((i+1)x_1 + i)\right) * \left(\sum\nolimits_{j=1}^{5}(j\ \ cos((j+1)x_2 + j)\right)$$

  With $x \in [-10, 10]^2$

- **HARTMAN3** function. The function is given by the following:

$$f(x) = -\sum_{i=1}^{4} c_i\ exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right),$$

$$\text{Where a} = a_{ij} = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},\ c = c_i = \begin{bmatrix} 1.0 \\ 1.2 \\ 3.0 \\ 3.2 \end{bmatrix},\ P = p_{ij} = \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$$

  With $x \in [0, 1]^3$

- **HARTMAN6** function. The function is given by the following:

$$f(x) = -\sum_{i=1}^{4} c_i\ exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right),$$

$$\text{Where a} = a_{ij} = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},\ c = c_i = \begin{bmatrix} 1.0 \\ 1.2 \\ 3.0 \\ 3.2 \end{bmatrix},$$

$$P = p_{ij} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8723 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

  With $x \in [0, 1]^6$

- **POTENTIAL** function. As a test case, the molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard–Jones potential [38] is utilized. The function to be minimized is defined as follows:

$$V_{LJ}(r) = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}],$$

  In the current experiments, two different cases were studied: N = 3, 5.

- **RASTRIGIN** function. The function is given by the following:

$$f(x) = x_1^2 + x_2^2 - cos(18x_1) - cos(18x_2),$$

With $x \in$ [-1, 1]^2

- **SHEKEL5** function:

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x - a_i)\ (x - a_i)^T + c_i}$$

$$\text{Where a} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{bmatrix}, \text{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{bmatrix},$$

With $x \in$ [0, 10]^4

- **SHEKEL7** function:

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x - a_i)\ (x - a_i)^T + c_i}$$

$$\text{Where a} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{bmatrix}, \text{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{bmatrix},$$

With $x \in$ [0, 10]^4

- **SHEKEL10** function:

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x - a_i)\ (x - a_i)^T + c_i}$$

$$\text{Where a} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}, \quad \text{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{bmatrix},$$

With $x \in$[0, 10]^4

- **ROSENBROCK** function. The function is given by the following:

$$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2),$$

At range $-30 \leq x_i \leq 30$, in the experiments using function with n = 16.

- **SINUSOIDAL** function

$$f(x) = -(2.5 \prod_{i=1}^{n} \sin(x_i - \mathrm{z}) + \prod_{i=1}^{n} \sin(5(x_i - \mathrm{z}))), \qquad 0 \ \leq x_i \leq \pi.$$

The experiments use n = 4, and $z = \frac{\pi}{6}$ and the corresponding function denoted by The label SINU4.

- **Test2N** function. This function is given by the equation:

$$f(x) = \frac{1}{2} \sum_{i=1}^{4} x_i^4 - 16x_i^2 + 5x_i\,, \qquad x_i \ \in [-5,5]$$

The function has 2^n in the specified range and in the experiments n = 4, 9.

- **Test30N** function. This function is given by:

$$f(x) = \frac{1}{10} sin^2(3\pi x_1) * \sum_{i=2}^{n-1} (\left(x_i - 1)^2 \left(1 + sin^2(3\pi x_{i+1})\right)\right) + (x_n - 1)^2 \left(1 + sin^2(2\pi x_n)\right)$$

$x_i \ \in [-10,10]$, The function has 30^n local minima in the search space and in the experiments n = 3, 4.

IV. RESULTS AND DISCUSSION

The parallel implementation of differential evolution was employed in comparative experiments, the performance of the proposed PDEmp algorithm was assessed through a series of experiments utilizing 3 parallel units. The existing OpenMP library [39] facilitated the parallelization, and the entire technique was implemented using the C++ programming language. All experiments were conducted on a system equipped with an Intel (R) Core (TM) i7-8650U multi-core processor and 16 GB of RAM, using (the Windows 11 Pro) operating system, and the experimental settings used in the proposed PDEmp algorithm are presented in Table 1.

Table 1: The following settings were initially used to conduct the experiments.

Parameter	Value	Explanation
N_d	120	Total elements for DE
N_k	200	number of iterations
F	0.8	Differential weight for DE
CR	0.9	Crossover Probability for DE
NP	10	Number of populations (agents)
propagation	1-to-1	scenario
N_R	5	iterations
N_I	2	islands
M	30	Number of iterations
ϵ	10^{-4}	Small positive number

In table 2, the columns display the average number of function calls for each problem, The first three columns labeled (Multi-start Method – PSO method – GWO method – PDE_t) refer to a different optimization methods that implement in parallel manner (Multi-start algorithm – Particle swarm optimization algorithm – Grey wolf optimization algorithm – differential evolution algorithm) respectively, The final column refers to the proposed PDEmp corresponds to the proposed Parallel Differential Evolution algorithm that utilizes a new mutation pool strategy. To ensure the reliability and validity of the research, experiments were conducted 30 times and concerned Table 2. The total function calls of benchmark tested functions using different parallel optimization methods and the proposed (PDEmp) algorithm, this comparison showed in Fig. 3.

Table 2: Statistical comparison of function calls across different optimization methods and The proposed (PDEmp) algorithm using a new strategy.

NO	PROBLEMS	Multi-start method	PSO method	GWO method	PDE_t	Proposed PDEmp
1	BF1 (Bohachevsky)	50762	3005	3080	6114	**300**
2	BF2 (Bohachevsky)	35711	2889	3083	7560	**300**
3	BRANIN	10,521	2417	2661	4289	**300**
4	CM4 (Cosine Mixture)	58,581	3144	3187	4079	**600**
5	Camel back	15,255	2547	2724	5940	**600**
6	Easom	5412	2232	2135	1721	**300**
7	EXP (16)	10,280	2676	4991	5339	**300**
8	GKLS	5908	2422	4840	2641	**300**
9	GRIEWANK	17,877	2820	3155	6915	**300**
10	Hansen	17,541	2587	2795	4263	**1500**
11	HARTMAN3	16,562	2550	2911	4566	**300**
12	HARTMAN6	21,015	2809	3408	5550	**600**
13	Potential 3	17,161	3170	3645	5256	**300**
14	Potential 5	30,249	4730	3380	7742	**300**
15	RASTRIGIN	23,015	2829	2806	5999	**300**
16	ROSENBROCK (16)	25,060	5170	8528	8307	**300**
17	SHEKEL5	16,972	2816	2778	5933	**1500**
18	SHEKEL7	17,127	2856	3324	5677	**2100**
19	SHEKEL10	17,135	2866	2866	5100	**2400**
20	SINUSOIDAL (4)	15647	2657	2231	4776	**1500**
21	Test function 2N4	15,806	2681	3603	5727	**1800**
22	Test function 2N9	20716	3067	4207	6288	**1800**
23	Test function 30N3	16,145	2762	2769	2869	**600**
24	Test function 30N4	15,907	2915	3144	2714	**600**
-	**TOTAL**	**464075**	**70617**	**82251**	**125365**	**19200**

The proposed PDEmp algorithm was evaluated on the GKLS test function across multiple versions with dimensionality ranging from 2 to 5. For each configuration, the number of function calls and the corresponding execution times were recorded to assess both computational efficiency and convergence performance. The experimental findings are summarized in Figures 4 and 5. Fig. 4 illustrates the number of function calls, while Fig. 5 presents the average execution times. The results clearly indicate that the proposed algorithm requires substantially fewer function evaluations compared with competing approaches. This reduction in function calls directly translates into shorter execution times, confirming the efficiency of the parallel framework.

These improvements can be attributed to two key features of the PDEmp algorithm: the incorporation of multiple mutation strategies through the mutation pool, which enhances search diversity and reduces premature convergence, and the propagation mechanism, which enables efficient sharing of superior solutions among parallel units. Together, these features accelerate convergence toward high-quality solutions while minimizing computational overhead.

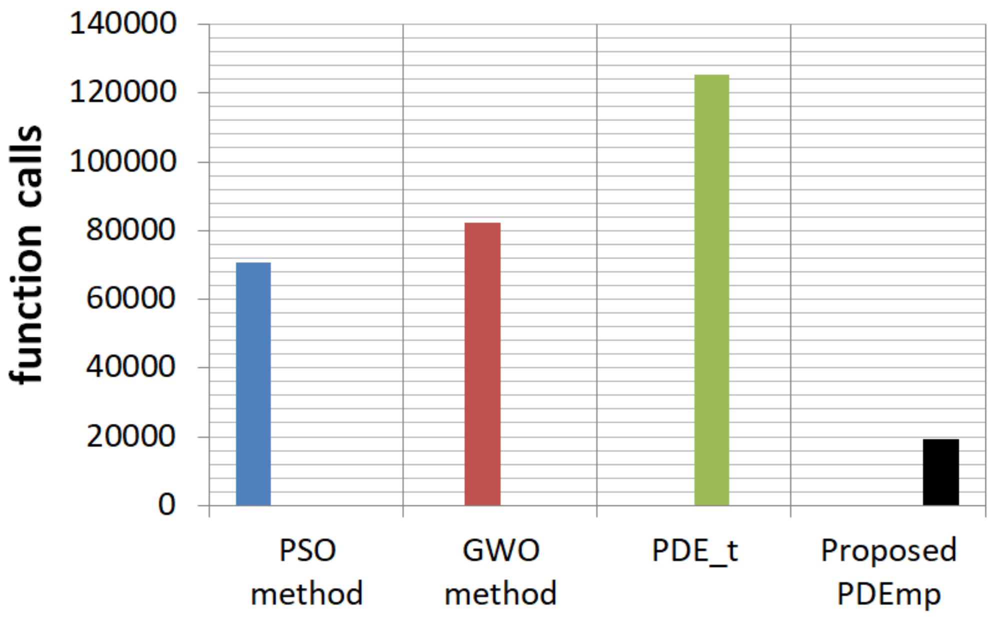

Fig. 3. Comparison of total function calls of benchmark tested functions using different parallel optimization methods and the proposed (PDEmp) algorithm

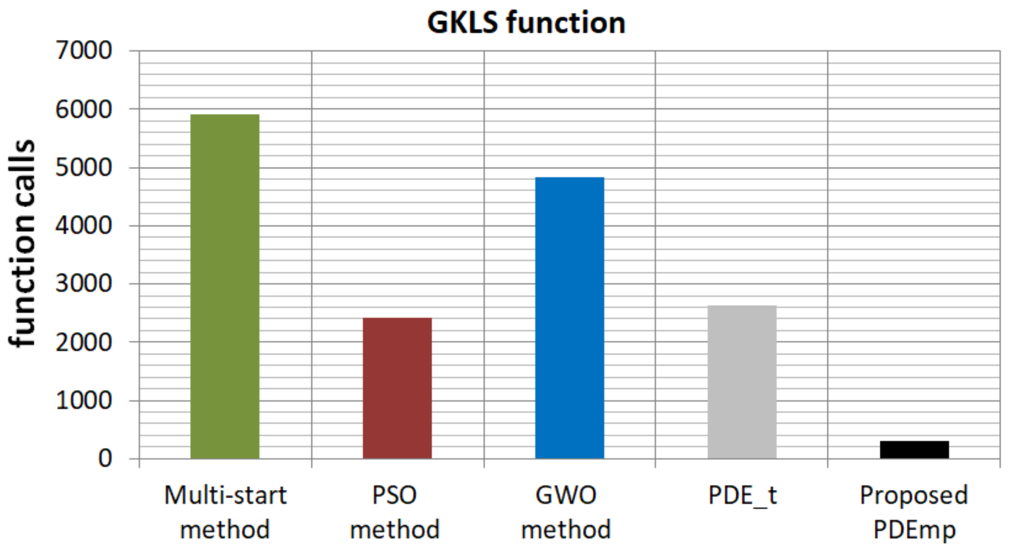

Fig. 4. The effect of the usage of the proposed (PDEmp) algorithm versus other parallel optimization methods applying to **GKLS** function.

Overall, the experiments on the benchmark functions demonstrate that PDEmp achieves significant gains in both convergence speed and computational efficiency. The reduction in function calls is particularly valuable for complex optimization problems where evaluating the objective function is computationally expensive. Table 3 provides further insights, presenting computational times, additionally; Time comparisons when applying the proposed PDEmp and different differential evolution optimization methods to GKLS test functions.

Table 3: Time comparisons (seconds) when applying parallel optimization methods and the proposed (PDEmp) algorithm to **GKLS** function.

NO	PROBLEMS	parallel optimization methods (Multi-start – PSO – DE)	Proposed PDEmp
1	GKLS (n=2)	0.225	**0.0593**
2	GKLS (n=3)	0.235	**0.0638**
3	GKLS (n=4)	0.25	**0.0725**
4	GKLS (n=5)	0.215	**0.0953**

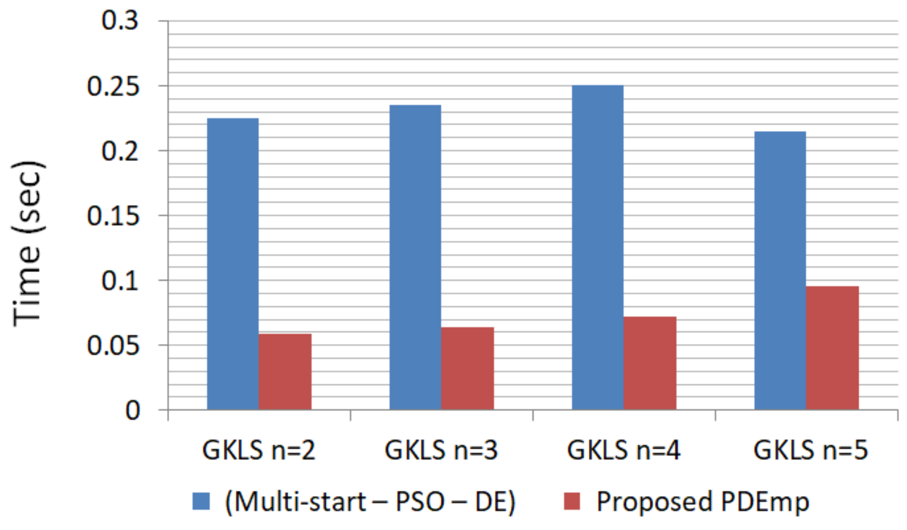

Fig. 5. Comparison of executing time applying parallel optimization methods and the proposed (PDEmp) algorithm to **GKLS** function.

V. CONCLUSION

Mutation operators and parameter settings play a critical role in enhancing the performance of Differential Evolution (DE). In this study, three mutation operators were incorporated into a mutation pool, and a parameter pool with three predefined values was established. During each generation, one mutation operator and corresponding parameter values were randomly selected to generate trial vectors. This design enables the algorithm to exploit the complementary strengths of different strategies, thereby improving its search capacity and convergence behavior. The proposed Parallel Differential Evolution with Mutation Pool (PDEmp) algorithm was evaluated on 24 benchmark functions and compared with both traditional DE and non-DE algorithms. The results demonstrate that PDEmp consistently outperforms the competing methods in terms of convergence speed and solution quality. The superior performance is primarily attributed to the random selection of multiple mutation strategies within a parallel execution framework. In addition to the mutation pool, the proposed method integrates an efficient propagation mechanism that disseminates the best solutions across parallel units, and a robust termination rule based on asymptotic criteria. These features ensure effective use of computational resources, reduced execution time, and broader applicability to different global optimization problems. The design of a mutation pool enhances solution quality and improves algorithmic performance and Comprehensive evaluation on benchmark functions, demonstrating that PDEmp is highly competitive compared with existing approaches. Overall, the study shows that combining parallel execution with adaptive mutation and termination strategies provides a promising direction for solving complex global optimization problems. Future research may extend this framework by exploring adaptive parameter control, hybridization with other evolutionary algorithms, and large-scale real-world applications.

REFERENCES

1. Wolfe, M.A. Interval methods for global optimization. Appl. Math. Comput. **1996**, 75, 179–206.
2. Csendes, T.; Ratz, D. Subdivision Direction Selection in Interval Methods for Global Optimization. SIAM J. Numer. Anal. **1997**, 34, 922–938.
3. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. Science **1983**, 220, 671–680.
4. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics based algorithm. Future Gener. Comput. Syst. **2019**, 101, 646–667.
5. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. Inf. Sci. **2009**, 179, 2232–2248.
6. Du, H.; Wu, X.; Zhuang, J. Small-World Optimization Algorithm for Function Optimization. In Proceedings of the International Conference on Natural Computation, Xi'an, China, 24–28 September **2006**; pp. 264–273.
7. Goldberg, D.E.; Holland, J.H. Genetic Algorithms and Machine Learning. Mach. Learn. **1988**, 3, 95–99.
8. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. IEEE Trans. Evol. Comput. **2011**, 15, 4–31.
9. Charilogis, V.; Tsoulos, I.G.; Tzallas, A.; Karvounis, E. Modifications for the Differential Evolution Algorithm. Symmetry **2022**, 14, 447.
10. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December **1995**; Volume 4, pp. 1942–1948.
11. Charilogis, V.; Tsoulos, I.G. Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions. Information **2022**, 13, 217.
12. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **1996**, 26, 29–41.
13. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. Eng. Comput. **2012**, 29, 464–483.
14. Mirjalili, S.; Lewis, A. The whale optimization algorithm. Adv. Eng. Softw. **2016**, 95, 51–67.
15. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. Adv. Eng. Softw. **2017**, 105, 30–47.
16. Honda, M. Application of genetic algorithms to modelings of fusion plasma physics. Comput. Phys. Commun. **2018**, 231, 94–106.
17. Luo, X.L.; Feng, J.; Zhang, H.H. A genetic algorithm for astroparticle physics studies. Comput. Phys. Commun. **2020**, 250, 106818.
18. Kim, H.S.; Tsai, L. Design Optimization of a Cartesian Parallel Manipulator. J. Mech. Des. **2003**, 125, 43–51.
19. Oh, S.; Jang, H.J.; Pedrycz, W. The design of a fuzzy cascade controller for ball and beam system: A

study in optimization with the use of parallel genetic algorithms. Science Direct Eng. Artif. Intell. **2009**, 22, 261–271.
20. Censor, Y.; Zenios, S. Parallel Optimization: Theory, Algorithms and Applications; Oxford University Press: Oxford, UK, **1998**; ISBN-13: 978-0195100624.
21. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. Glob. Optim. **1997**, 11, 341–359.
22. Bilal; Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential Evolution: A review of more than two decades of research. Eng. Appl. Artif. Intell. **2020**, 90, 103479.
23. Guo, S.-M.; Yang, C.-C.; Hsu, P.-H.; Tsai, J.S.H. Improving Differential Evolution with a Successful-Parent-Selecting Framework. IEEE Trans. Evol. Comput. **2015**, 19, 717–730.
24. Afzal, A.; Ansari, Z.; Faizabadi, A.R.; Ramis, M. Parallelization Strategies for Computational Fluid Dynamics Software: State of the Art Review. Arch. Comput. Methods Eng. **2017**, 24, 337–363.
25. Feoktistov, V. Differential Evolution. In Search of Solutions. Optimization and Its Applications; Springer: Berlin/Heidelberg, Germany, **2006**.
26. Rocca, P.; Oliveri, G.; Massa, A. Differential Evolution as Applied to Electromagnetics. IEEE Antennas Propag. Mag. **2011**, 53, 38–49.
27. Lee, W.S.; Chen, Y.T.; Kao, Y. Optimal chiller loading by differential evolution algorithm for reducing energy consumption. Energy Build. **2011**, 43, 599–604.
28. Yuan, Y.; Xu, H. Flexible job shop scheduling using hybrid differential evolution algorithms. Comput. Ind. Eng. **2013**, 65, 246–260.
29. Xu, L.; Jia, H.; Lang, C.; Peng, X.; Sun, K. A Novel Method for Multilevel Color Image Segmentation Based on Dragonfly Algorithm and Differential Evolution. IEEE Access **2019**, 7, 19502–19538.
30. W. Deng, J. Xu, Y. Song, H. Zhao, Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem, Appl. Soft Compute. 100 (**2021**).
31. Mallipeddi, R.; Suganthan, P.N.; Pan, Q.K.; Tasgetiren, M.F. Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl. Soft Comput. **2011**, 11, 1679–1696.
32. Wang, Y.; Cai, Z.; Zhang, Q. Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters. IEEE Trans. Evol. Comput. **2011**, 15, 55–66.
33. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. IEEE Trans. Evol. Comput. **2009**, 13, 398–417.
34. Charilogis, V.; Tsoulos, I.G.; Tzallas, A.; Karvounis, E. Modifications for the Differential Evolution Algorithm. Symmetry **2022**,14.
35. Ali, M.M. Charoenchai Khompatraporn, Zelda B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. J. Glob. Opt. **2005**, 31, 635–672.
36. Floudas, C.A.; Pardalos, P.M.; Adjiman, C.; Esposoto, W.; G¨um¨us, Z.; Harding, S.; Klepeis, J.; Meyer, C.; Schweiger, C. Handbook of Test Problems in Local and Global Optimization; Kluwer Academic Publishers: Dordrecht, The Netherlands, **1999**.
37. Gaviano, M.; Ksasov, D.E.; Lera, D.; Sergeyev, Y.D. Software for generation of classes of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. **2003**, 29, 469–480.
38. Lennard-Jones, J.E. On the Determination of Molecular Fields. Proc. R. Soc. Lond. A **1924**, 106, 463–477.
39. Chandra, R.; Dagum, L.; Kohr, D.; Maydan, D.; McDonald, J.; Menon, R. Parallel Programming in OpenMP; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, **2001**.