Available online at www.HighTechJournal.org

HighTech and Innovation Journal

ISSN: 2723-9535

Vol. 5, No. 2, June, 2024

An Adaptive Differential Evolution with Multiple Crossover Strategies for Optimization Problems

Irfan Farda [1], Arit Thammano [1*]

[1] *Computational Intelligence Laboratory, School of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand.*

Received 10 January 2024; Revised 07 May 2024; Accepted 11 May 2024; Published 01 June 2024

Abstract

The efficiency of a Differential Evolution (DE) algorithm largely depends on the control parameters of the mutation strategy. However, fixed-value control parameters are not effective for all types of optimization problems. Furthermore, DE search capability is often restricted, leading to limited exploration and poor exploitation when relying on a single strategy. These limitations cause DE algorithms to potentially miss promising regions, converge slowly, and stagnate in local optima. To address these drawbacks, we proposed a new Adaptive Differential Evolution Algorithm with Multiple Crossover Strategy Scheme (ADEMCS). We introduced an adaptive mutation strategy that enabled DE to adapt to specific optimization problems. Additionally, we augmented DE with a powerful local search ability: a hunting coordination operator from the reptile search algorithm for faster convergence. To validate ADEMCS effectiveness, we ran extensive experiments using 32 benchmark functions from CEC2015 and CEC2016. Our new algorithm outperformed nine state-of-the-art DE variants in terms of solution quality. The integration of the adaptive mutation strategy and the hunting coordination operator significantly enhanced DE's global and local search capabilities. Overall, ADEMCS represented a promising approach for optimization, offering adaptability and improved performance over existing variants.

Keywords: Metaheuristic Algorithm; Differential Evolution Algorithm; Multiple Strategies; Reptile Search Algorithm.

1. Introduction

Optimization problems and their solutions remain important for practical applications, such as enhancing wireless communication systems [1], finding the best route for traveling salesman problems [2], optimizing manufacturing processes [3], improving parameter extraction in photovoltaic models [4, 5], and others. Optimization involves identifying the most favorable solution or optimal values for parameters in order to attain a specific outcome or objective. This goal can be to minimize a cost or loss function or to maximize an objective or reward function, depending on the problem to be solved [6]. Historically, numerous mathematical programming methods, such as Linear Programming, Dynamic Programming, and Newton's Methods were traditionally employed for solving optimization problems. However, as the complexity of these problems has grown, those conventional methods are often not sufficient [7]. New methods have been provided by researchers to solve optimization problems, collectively referred to as metaheuristics [8].

Metaheuristics, a term first coined by Glover in 1986, refers to algorithms that use heuristics within a more general framework, allowing them to be applied to a wider range of problems. The term "metaheuristics" is derived from the

* Corresponding author: arit@it.kmitl.ac.th

http://dx.doi.org/10.28991/HIJ-2024-05-02-02

Greek words "meta," meaning beyond or at an elevated level, and "heuristics," meaning the act of finding [9]. Metaheuristic methods have become commonly used for solving optimization problems compared to other methods due to their simplicity and reliability in producing good results in a wide range of fields, including engineering, business, transportation, and social sciences [10].

Nature-inspired algorithms are a well-established type of metaheuristic approach that have been successfully applied to solving complex optimization problems and real-world issues that cannot be addressed by traditional gradient-based or approximation optimization techniques [11]. Some well-known algorithms include the Genetic Algorithm (GA) introduced by Holland [12], based on the concepts of genetic inheritance and natural selection found in evolution. Differential Evolution (DE), proposed by Storn & Price [13], was inspired by natural selection and mutation in biology. Other biologically inspired algorithms include Kennedy and Eberhart's Particle Swarm Optimization (PSO), which mimics the behavior of flocking birds and their movement patterns while foraging [14]. Ant Colony Optimization (ACO), developed by Dorigo et al. [15], emulates the cooperation and communication of ants as they forage and uses this principle to tackle optimization problems and find the optimal solution. The Artificial Bee Colony (ABC) Algorithm, developed by Karoboga & Basturk [16], simulates the foraging of honeybees. Yang [17] created the Bat Algorithm (BA), based on the behavior of bats in locating prey, which adapted to changing conditions. Yang added a Firefly Algorithm (FA), inspired by the flashing of fireflies [18], to the biology-inspired repertoire. The Cuckoo Search Algorithm (CSA) of Yang & Deb [19] modeled the egg-laying strategy and aggressive brood parasitism of cuckoos. The Bees Algorithm of Pham et al. [20] was also based on foraging—honeybees in their case. Grey Wolf Optimization (GWO), proposed by Mirjalili et al. [21], replicated the hierarchical leadership structure and hunting behavior observed in grey wolf packs. All of these nature-inspired algorithms are computational models based on the behaviors observed in nature for solving optimizations.

Differential Evolution (DE) algorithms have been widely used to solve optimization problems [22, 23]. Although it was proposed several decades ago, work continues to improve its performance and extend its capabilities. Several improved DE algorithms are reviewed and discussed here. Li et al. [24] introduced a differential evolution algorithm using Leader-Adjoint Populations (LADE), designed to balance global and local ability by integrating four mutation strategies across various stages of the evolution. This helped to prevent negative effects, such as premature convergence or failure to converge, resulting from improper mutation strategy settings. LADE was tested on various optimization functions and was found to be competitive, outperforming other DE variants and two well-known metaheuristics. Subsequently, Li et al. [25] presented a Dual Mutation Strategy Collaboration algorithm that blended an elite guidance mechanism with a dual mutation strategy collaboration to achieve a balance between global search and local optimization, which was also tested and found effective on unimodal, step, quartic, and multimodal functions. Meng et al. [26] studied established DE variants and identified two weaknesses: ineffective control parameter adaptation and inadequate mutation strategy, which could lead to slow convergence and less than optimum performance. They proposed their PaDE, which included a new control parameter adaptation method, population reduction techniques, and an improved timestamp-based mutation strategy, to address these issues. PaDE was found to produce outcomes competitive to LPALMDE, JADE, iLSHADE, LSHADE, SHADE, and jSO.

Deng et al. [27] addressed the challenges in setting appropriate control parameters and selecting a reasonable mutation strategy in solving engineering optimization problems; they proposed a novel and enhanced DE algorithm, WMSDE, which integrated the complementary benefits of five mutation strategies and used a wavelet basis function. Their WMSDE had a balance of local and global search ability, fast convergence, improved population diversity, and enhanced search quality when compared to five established DE variants. They validated its efficiency by applying it to a real-world airport gate assignment problem and achieved an impressive airport gate assignment rate of 97.6% of 100%. Mohamed & Mohamed [28] presented a new Adaptive Guided Differential Evolution (AGDE) for solving continuous space numerical optimization problems: AGDE balanced global exploration and local exploitation through a new mutation rule that selected vectors from different regions of the population. Moreover, AGDE had a flexible adaptation scheme that adjusted the crossover rate without additional parameters or knowledge of the problem. In testing against CMAES, COOA, MDE-pBX, CCPSO2, and SMADE, AGDE performed better in terms of robustness, stability, and solution quality. Finally, Deng et al. [29] introduced a Hybrid Mutation-based Cooperative Framework Quantum Differential Evolution (HMCFQDE) to overcome low solution efficiency, insufficient diversity, a low convergence rate, and a high search stagnation possibility. HMCFQDE combined the quantum computing characteristics of the quantum evolutionary algorithm and the divide-and-conquer concept of the cooperative coevolution evolutionary algorithm to enhance performance. It used a hybrid mutation strategy based on local neighborhood mutation and SaNSDE, quantum chromosome encoding to increase population diversity, and a cooperative coevolution framework to divide the problem into low-dimensional sub-problems. HMCFQDE was tested on six functions and found to have higher convergence accuracy and stability, particularly for high-dimensional complex functions.

Gao et al. [30] presented a version of the Adaptive Differential Evolution with Optional External Archive algorithm named CJADE, which integrated chaotic local search (CLS) to avoid premature convergence and sticking at a local optimum. Four schemes incorporating CLS into JADE were studied, including individual, random, parallel, and

memory-selective incorporation. They showed that the memory-based CJADE-M performed the best among JADE, other CJADE variants, and other optimization algorithms. Nadimi-Shahraki et al. [31] introduced a new optimization technique: multi-trial vector-based differential evolution (MTDE). Its main feature was its adaptive step size, which was determined through a multi-trial vector approach (MTV). It combined various exploration strategies through trial vector producers (TVPs) and applied them to specific subpopulations. MTDE used three TVPs: representative-based, local random-based, and global best-history-based TVPs. Evaluation of MTDE on the CEC 2018 benchmark suite and four complex engineering design problems showed that the MTV approach significantly enhanced the performance of the MTDE algorithm and outperformed other algorithms. Sun et al. [32] introduced a new variation, CSDE, which aimed to address challenges in the mutation operator scaling factor and crossover rate parameters. CSDE used two mutation operators, one effective in global exploration and another for local exploitation. To balance these two mutation operators, a coordination mechanism that adjusted their historical success rate was used. Additionally, the scaling factor and crossover rate were adaptively determined by introducing a periodic function, an individual-independent macro-control function, and a function based on fitness value information that was dependent on the individuals. Experiments on 30 benchmark functions and 4 real-world problems showed that CSDE has good performance.

Zhan et al. [33] addressed the limitations faced by traditional DDE algorithms when it comes to strategy selection and parameter setting in their Adaptive Distributed Differential Evolution (ADDE) algorithm, which used a master-slave multi-population approach that included three populations—exploration, exploitation, and balance—which were dynamically adjusted in each iteration to enhance collaboration and improve global optimization. The populations dynamically selected their mutation strategy based on evolutionary state estimation, and the parameters (amplification factor, crossover rate, and population size) were adaptively updated based on past successful experiences and best solution improvement. Tests conducted on benchmark functions and real-world problems showed that ADDE was better than DDE-SD, IBDDE, DDEM-RCA, AsAMP-dDE, and Cloudde. Sun et al. [34] introduced a new method, ARSA, a dynamic space adjustment-based adaptive regeneration framework, that generated new individuals when a member of the population did not improve for a certain number of generations. It used two strategies for producing substitute individuals, one emphasizing global and the other local exploitation. ARSA parameters were dynamically tuned using a macroparameter and an individual-based microparameter, enabling a balance between exploitation and exploration. ARSA was tested on IEEE CEC 2017 and three real-world problems. ARSA improved the performance of DE with slight modifications and without slowing it down. Yu et al. [35] proposed a global optimum-based search (GoS) method, which aimed to balance between local and global search capabilities, enhancing search efficiency. This method was triggered when the global optimum did not change over a specific number of generations, and it involved a local refinement based on feedback from the global optimum. DEGoS was adopted in DE and showed significant improvements in search efficiency and solution quality.

Viktorin et al. [36] presented a modification to the Success-History-Based Adaptive Differential Evolution (SHADE) algorithm aimed at avoiding premature convergence in higher-dimensional search spaces. The modification involved using a distance-based adaptation mechanism rather than a change in objective function value to control the scaling factor and crossover rate; it was shown to be effective in experiments on the CEC2015 and CEC2017 benchmark sets, and it resulted in better optimization than the original SHADE, L-SHADE, and jSO algorithms. Wang et al. [37] proposed a self-adaptive mutation differential evolution algorithm, DEPSO. The DEPSO algorithm enhanced the performance by integrating the DE/rand/1 mutation strategy with the mutation strategy from Particle Swarm Optimization (PSO) and was tested on 30-dimensional and 100-dimensional test functions, where it outperformed jDE, CoDE, DEMPSO, aDE, EPSDE, and IMSaDE. In addition, DEPSO was applied to solve arrival flight scheduling and effectively decreased the delay time. Meng & Pan [38] presented a new differential evolution algorithm, HARD-DE, which had two enhancements. First, the mutation strategy was based on a hierarchical archive and included information about the depth of evolution to better understand the objective function's landscape. Second, it had novel adaptation schemes for the three control parameters. It was tested using two test suites containing 58 benchmark functions for real-parameter numerical optimization and 2 benchmarks for real-world optimization, where it secured an overall better performance, although it needed more time to run.

Tian & Gao [39] proposed a novel differential evolution (NDE) algorithm by combining a neighborhood-based mutation (NM) strategy and a neighborhood-based adaptive evolution (NAE) technique. The NM strategy adjusted the search performance of each individual adaptively by developing two novel mutation operators and an individual-based selection probability. The NAE mechanism identified and alleviated the evolutionary dilemmas of the neighborhood by tracking its performance and diversity and using a dynamic neighborhood model and two exchange operations. A simple reduction method was also used to adjust the population size adaptively. Experiments on 30 benchmark functions and a real-world application showed that NDE was reliable and performed well. Civicioglu & Besdok [40] introduced a Bernstain-Search Differential Evolution (BSD) algorithm, which was a universal differential evolution algorithm, implying that it was easier to control than previous algorithms and did not need to go through trial-and-error tasks to select the genetic operators. BSD used a unique bijective mutation strategy and a more efficient crossover operator controlled randomly by using Bernstein polynomials, which made it simple, non-recursive, highly efficient, and fast. Experiments on 30 benchmark problems, 60 classic benchmark problems, image evolution problems, and one

triangulated irregular network refinement problem showed that BSD outperformed ABC, WDE, CUCKOO, and JADE. A novel Time-Varying DE (TVDE) was proposed by Sun et al. [41]. It used three time-varying functions to create a new mutation operator and to adaptively adjust the values of two control parameters (crossover rate and scaling factor) during evolution. TVDE was evaluated against seven other DE variants on CEC 2014 and four real-world problems and emerged as the top-performing algorithm across all tested algorithms.

To summarize, many DE variants have been proposed to enhance performance, including integrated different mutation strategies, control parameter adaptation schemes, or combinations of DE with other algorithms. These variants were effective in balancing global exploration and local exploitation, improving population diversity, increasing convergence accuracy, and overcoming premature convergence. However, there are still several challenges that remain unresolved completely in DE algorithm, including:

1. Ineffective control parameter adaptation and inadequate mutation strategy, which can lead to slow convergence and poor optimization,
2. Low solution efficiency, insufficient diversity, slow convergence speed and high search stagnation possibility and
3. Premature convergence and getting stuck in local optima.

This study addressed existing challenges and further enhanced DE algorithms. We introduce an Adaptive Differential Evolution Algorithm with Multiple Crossover Strategy Scheme (ADEMCS). Our approach includes an adaptive mutation strategy that enables the algorithm to autonomously determine mutation control parameters based on the characteristics of the problems. Additionally, we integrated DE with a hunting coordination operator from the reptile search algorithm [42] to enhance local search ability and convergence rates.

Unlike previous adaptive mutation strategy variants, our method introduced two control mutation operators: one dynamically adjusted based on algorithm conditions and mutant vector performance, and the second remained constant. This unique approach contributed to the effectiveness of our adaptive mutation strategy. Furthermore, we focused on integrating DE with other heuristic algorithms, particularly emphasizing the crossover operator. While previous research generally concentrated on the mutation operator, we believe that optimizing the crossover operator significantly impacts DE performance. Therefore, our modification and optimization of the crossover operator further enhance DE efficiency and effectiveness.

The remainder of the article is organized as follows: In Section 2, the classical differential evolution is described. In Section 3, our new Adaptive Differential Evolution Algorithm with Multiple Crossover Strategy Scheme (ADEMCS) is thoroughly explained. Section 4 presents the results and discusses the experiments. Section 5 concludes, highlighting the contributions made and pointing out potential avenues for future research.

2. Differential Evolution Algorithm

Differential Evolution (DE) is an optimization technique that operates on a population of candidate solutions, iteratively adjusting them to seek the optimal solution for a given problem. The method, first introduced by Storn & Price (1997) [13], has been used widely in various fields, including engineering [31, 43–45], health science [46, 47], and manufacturing [48]. The core principle generates new candidate solutions by combining the characteristics of existing solutions within the population. Specifically, in each iteration (or evolutionary steps), a new candidate solution is created by adding the weighted difference between two randomly chosen solutions to make a new solution. This new candidate solution is then evaluated against the current solution, and if it is found to be superior, it replaces the current solution. The algorithm is further explained in the following sub-sections.

2.1. Population Initialization

The initial step in differential evolution is population initialization, which creates potential solutions for the optimization problem. This step randomly generates *NP* target vectors, represented as $X_i^G = (x_{i\,1}\ x_{i\,2}\ \ldots\ x_{i\,j}\ \ldots\ x_{i\,d})$, from a uniform or Gaussian distribution, where *G* is the generation number; $i = 1, 2, \ldots, NP$; *NP* is the number of individuals in the population (the population size) and $j = 1, 2, \ldots, d$; *d* is the number of dimensions (or variables). Each element of a target vector is generated by using (1).

$$x_{i,j} = LB_j + rand[0,1] \times (UB_j - LB_j) \quad (1)$$

where LB_j and UB_j refer to lower and upper boundaries of the search space in dimension, j. The term $rand[0\ 1]$ denotes a real number randomly generated in [0, 1]. These bounds are used to ensure that the generated candidate solutions fall within the acceptable range of the problem.

2.2. Mutation

The second step involves generating a mutant vector $V_i^G = (v_{i\,1}\ v_{i\,2}\ \ldots\ v_{i\,j}\ \ldots\ v_{i\,d})$ through the use of a mutation operator. In the DE mutation operation, a new or mutant vector is generated by combining the difference of two or more

parent vectors. There are several mutation strategies, each with its own set of advantages and disadvantages. Some commonly used examples include the following:

- DE/best/1 [24, 25].

$$V_i^G = X_{best}^G + F(X_{r1}^G - X_{r2}^G) \quad (2)$$

- DE/best/2 [24, 25].

$$V_i^G = X_{best}^G + F(X_{r1}^G - X_{r2}^G) + F(X_{r3}^G - X_{r4}^G) \quad (3)$$

- DE/rand/1 [24, 25].

$$V_i^G = X_{r1}^G + F(X_{r2}^G - X_{r3}^G) \quad (4)$$

- DE/rand/2 [24, 25].

$$V_i^G = X_{r1}^G + F(X_{r2}^G - X_{r3}^G) + F(X_{r4}^G - X_{r5}^G) \quad (5)$$

- DE/current-to-best/1 [25].

$$V_i^G = X_i^G + F(X_{best}^G - X_i^G) + F(X_{r1}^G - X_{r2}^G) \quad (6)$$

- DE/current-to-pbest/1 [48].

$$V_i^G = X_i^G + F\left(X_{pbest}^G - X_i^G\right) + F(X_{r1}^G - X_{r2}^G) \quad (7)$$

where the best vector in the current generation is represented by X_{best}^G and X_{pbest}^G represents the best individual from the top p% of the current population. Additionally, there are five other random target vectors in the current population, represented by X_{r1}^G, X_{r2}^G, X_{r3}^G, X_{r4}^G, and X_{r5}^G. It is important to note that these random target vectors are not the same as X_i^G. The mutation control parameter, also known as the scaling factor or F, plays a crucial role in DE. Its purpose is to control the degree of difference between the target vector and the mutant vector. A higher value of F results in a larger difference and a wider exploration space, while a lower value results in a smaller difference and a greater chance of exploitation.

It should be emphasized that the mutation operation is a vital aspect of DE, as it allows the algorithm to explore new regions of the search space and generate diverse candidate solutions that have the potential to be better than the current best solution

2.3. Crossover

The third step performs a crossover operation to generate a trial vector, $U_i^G = \left(u_{i\,1}\ u_{i\,2}\ \ldots\ u_{i\,i}\ \ldots\ u_{i\,d}\right)$. This operation combines information from the target vector, and the mutant vector to create a new vector that has the potential to be better than either of the two vectors. There are several types of crossover operators, one commonly used is called binomial crossover, defined as follows:

$$u_{i,j} = \begin{cases} v_{i,j}; & if\ rand(0,1] \le CR\ or\ j = j_{rand} \\ x_{i,j}; & otherwise \end{cases} \quad (8)$$

In binomial crossover, a single random parameter, the crossover probability, $CR \in (0,1]$, is used to control the degree of genetic mixing between the parents. A higher value of *CR* results in a greater chance of genes from the mutant vector being passed on to the trial vector, while a lower value of *CR* results in a greater chance of the trial vector being identical to the target vector. Additionally, $j_{rand} \in [1, d]$, represents a randomly selected index from the set of indices of the dimensions. This condition ensures that the trial vector U_i^G is different from both the target vector X_i^G and the mutant vector V_i^G.

It is important to note that the crossover operation is closely related to the mutation operation in DE, as both are used to generate new candidate solutions. The main difference is that the crossover operation uses information from existing solutions in the entire population, whereas the mutation operation just generates new solutions from existing solutions.

2.4. Selection

The last step is the selection step. This step compared the fitness of the trial vector to that of the target vector. If the trial vector has a better fitness, it is chosen as the new target vector and replaces the previous one. If the trial vector has a lower fitness, the target vector remains unchanged. This step can be expressed formally as follows:

$$X_i^{G+1} = \begin{cases} U_i^G; & if\ f(U_i^G) \le f(X_i^G) \\ X_i^G; & otherwise \end{cases} \quad (9)$$

where $f(U_i^G)$ and $f(X_i^G)$ denote the fitness of the trial and target vectors, respectively.

This step is crucial as it allows the algorithm to decide whether to adopt a new solution that has the potential to be better than the current best or to stay with the current best solution. It is also essential for the algorithm to converge to a satisfactory solution.

3. Proposed Methodology

Differential Evolution (DE) has been used for a diverse range of problems. However, like any optimization technique, it also has its limitations. One of the main drawbacks of DE is the sensitivity of its performance to the choice of control parameters, such as the mutation control operator. These parameters are crucial in helping algorithms explore the solution space effectively. Using a fixed control mutation parameter, where the same values are applied to all problems, DE will not be able to adapt to the unique characteristics of the problem, resulting in suboptimal solutions and poor performance. Furthermore, using a single crossover strategy also limits the adaptability of the algorithm to the problem at hand, as different problems require different strategies to achieve optimal results. If a single strategy is used, the algorithm will not be able to efficiently balance the exploration and exploitation abilities, leading it to get stuck at a local optimum. Therefore, it is crucial to carefully select the control parameters and crossover strategy to match the unique characteristics of the problem to attain an optimal solution.

In this paper, we tackle the challenges of fixed control parameters and relying on a single crossover strategy. Our new adaptive mutation strategy automatically adjusts the control parameter values based on the problem's characteristics, leading to better results. Additionally, we suggested using a multiple crossover strategy approach during the optimization tasks. By doing so, the algorithm can better avoid getting stuck in local optima and significantly enhance its overall performance. To further boost the benefits of using multiple crossover strategies, we introduced a comprehensive multiple strategy selection that optimizes their combined impact, making the algorithm more capable of handling complex problems. Our new algorithm, the Adaptive Differential Evolution Algorithm with Multiple Crossover Strategy Scheme (ADEMCS), embodies these innovative techniques. A detailed explanation follows.

3.1. Adaptive Mutation Strategy

In this sub-section, we used a new adaptive mutation strategy to enhance the ability of the mutation strategy to discover optimal solutions during the evolutionary step. The mutation strategy was based on the DE/current-to-pbest/1 strategy by Zhang and Sanderson [49]. This strategy is widely recognized and commonly used due to its numerous advantages. It provided a well-balanced approach for global numerical optimization, effectively combining exploration and exploitation capabilities while adapting to specific problems [32]. The DE/current-to-pbest/1 algorithm demonstrated its superiority or effective competition with traditional and other adaptive DE variants [50]. The DE/current-to-pbest/1 strategy enhanced global exploration by using top-performing individuals, expedited local convergence by leveraging past exploration, and dynamically adjusted mutation parameters, resulting in improved performance when tackling complex optimization problems [51]. The good performance of the DE/current-to-pbest/1 strategy motivated many researchers to base their work on it [52]. However, the primary drawback of the DE/current-to-pbest/1 mutation operator was its tendency to limit the diversity within the population. While it excelled at achieving convergence and exploiting promising solutions, it did not explore the solution space comprehensively. This limitation resulted in a restricted diversity of solutions. Such a lack of diversity can pose a problem in optimization algorithms, potentially leading to premature convergence, where the algorithm becomes stagnant at a local optimum without thoroughly exploring other potentially superior regions of the search space. To address this limitation, researchers have often attempted to modify the strategy or propose new approaches to adapt the mutation and crossover operator parameters. Here, we explored a different avenue to further enhance this strategy. Specifically, we introduced modifications by incorporating two mutation control parameters (F and λ) and adopting a new adaptive strategy to adjust these parameters. The modified version of this mutation operator was described as follows:

$$V_i^G = X_i^G + \lambda\left(X_{pbest}^G - X_i^G\right) + F(X_{r1}^G - X_{r2}^G) \quad (10)$$

where X_{pbest}^G is the best individual from the top p% of the current population, with p was set to the default of 5. X_{r1}^G represents a randomly selected vector from the current population, while X_{r2}^G is a random vector from the union of the current population and the external archive. The archive started empty and gathered solutions that were not selected to proceed to the next generation. If it grew too large, reaching a predetermined limit, a few solutions would be randomly removed to maintain its size. This archive played a crucial role by retaining valuable vectors from previous generations, preventing the algorithm from prematurely converging to local optima and ensuring population diversity.

In addition, the role of F and λ' in the modified mutation strategy' was to scale the difference between the p best and the target vectors and the difference between two random vectors. They were used as the first and second mutation

control parameters, important in determining exploration and exploitation of the solution space. The value of *F* was set to 0.5 while λ updated as follows:

$$\lambda_{i+1}^{G} = \begin{cases} \lambda_i^G + (\lambda_i^G \times C \times lr);\ if\ f(V_i^G) \le f(X_i^G) \\ \lambda_i^G - (\lambda_i^G \times C \times lr);\ otherwise \end{cases} \quad (11)$$

where *C* is an adaptation rate control for controlling the step size in changing the mutation parameters. *C* was calculated from

$$C = \left| \frac{f(V_i^G) - f(X_i^G)}{f(X_i^G) + \varepsilon} \right| \quad (12)$$

where ε is a small constant used for avoiding division by zero. Based on this scenario, when λ was increased, the influence of the difference between the *p best* and the target vectors on the mutation strategy became stronger. This meant that the algorithm would focus more on exploring the direction of *p best* vectors in the solution space. Conversely, when λ decreased, the influence of the difference between *p best* and the target vectors became weaker. This encouraged the algorithm to explore the solution space more widely and consider a broader range of possibilities.

The learning rate, *lr*, used in controlling adaptation rate for the mutation parameters. It was a tunable parameter that influenced the speed that the mutation parameters were updated. A well-tuned learning rate helped the algorithm find a balance between global exploration and local exploitation, leading to faster convergence and improved performance.

Note that it was important to keep the mutation control parameter λ in the range (0,1] to prevent negative effects on the performance. To achieve this, if $(\lambda > 1) \cup (\lambda \le 0)$, it was reset to 0.5. This ensured that the algorithm was able to function effectively and produced optimal results.

The strategy for controlling mutation adaptively is thought to enhance performance by allowing it to adapt to the specific features of the problem and thereby enhance its capacity to find optimal solutions.

3.2. Multiple Crossover Strategy Scheme

Crossover operation is a critical aspect of differential evolution algorithms as it significantly impacts exploration and exploitation capabilities. Various crossover operators are available; however, no single operator is suitable for all types of problems. To overcome this limitation, this paper investigates the use of more than one crossover operator to enhance performance. Our multiple crossover strategy used two crossover operators: a binomial operator and a hunting coordination operator from RSA [42], outlined in the following paragraph.

RSA demonstrated effectiveness when addressing a diverse set of optimization problems, consistently delivering good performance. Its contributions collectively lead to a streamlined, balanced, and efficient optimization task [53]. Its advantages included a distinctive hybrid methodology, a good track record in addressing complex challenges, a direct emulation of natural foraging principles, and the ability to balance exploration and exploitation for solution discovery [54]. RSA achieved its optimization power by mimicking two key natural behaviors: encircling and hunting, akin to strategies used by reptiles like crocodiles. The encircling behavior allowed RSA to explore globally, while the hunting behavior enabled a thorough local search. These mechanisms empowered RSA to engage in a refined and comprehensive exploration, thereby enhancing the quality of solutions [55].

We integrated RSA principles into our ADEMCS to enhance the local exploitation potential. By reinterpreting the concept of the best-obtained RSA solution in the current best vector in the population, we used a 'hunting coordination operator' as the crossover operator to generate the trial vector:

$$u_{i\,j} = x_{best\,j} \times p_{i\,j} \times rand(0\ 1) \quad (13)$$

where $P_i = \left(p_{i\,1}\ p_{i\,2}\ \ldots\ p_{i\,j}\ \ldots\ p_{i\,d}\right)$ denotes the fractional difference between the best vector and the target vector. *i*, calculated by:

$$p_{i\,j} = \alpha + \frac{x_{i\,j} - M_i}{x_{best\,j} \times \left(UB_j - LB_j\right) + \varepsilon} \quad (14)$$

The sensitivity parameter, α = 0.1, was used for regulating the precision of exploration (the dissimilarity between candidate solutions) for the trial vector throughout the iteration steps. UB_i and LB_i represented the upper and lower bounds of the *j*th dimension of the target vector, *i*. The average position of the target vector for the next generation, denoted as *M*, was determined by:

$$M_i = \frac{1}{d}\sum_{j=1}^{d} x_{i\,j} \quad (15)$$

M guides the generation of trial vectors by providing a reference point in the search space. By calculating the average position of the target vectors, we obtained an approximation of the potentially promising region for the current iteration. This helped steer the algorithm towards areas with better solutions, contributing to faster convergence.

By incorporating RSA's 'hunting coordination operator', our algorithm gained enhanced local search capability. A trial vector was generated based on the current best vector in the population, controlled by *P* and *α,* and considering the average position, *M*, of the target vectors. This strategy allowed our algorithm to refine its solutions iteratively, move towards promising search space regions, converge faster, and have better overall performance. Combining with the 'hunting coordination operator' brought together the strengths of both algorithms.

One of the most important things when using multiple crossover strategies is operator selection, which involves dynamically choosing the most suitable operator for each iteration, based on the problem's characteristics and the algorithm's progress. By incorporating a well-designed operator selection mechanism, the algorithm gained the flexibility to respond to the changing landscape of the problem, converging faster and discovering high quality solutions efficiently. For our operator selection, the crossover operator used to generate the trial vector U_i^G was selected by:

$$U_i^G = \begin{cases} Binomial; \ if \ rand(0\ 1] \leq SCR \\ RSA \quad\quad ; \ otherwise \end{cases} \tag{16}$$

where SCR denotes the success rate of the crossover. Initially, *SCR* = 0.5 to give an equal chance for both operators to be selected to create the trial vector. To harness the maximum benefit and improve performance through the use of multiple crossover strategies, we adopted a new approach for updating the probability of *S*: it was set to update every five generations, following Equation 17;

$$SCR = \frac{SC1}{SC1 + SC2} \tag{17}$$

where *SC1* and *SC2* represented the success rates of the first crossover operator (binomial) and the second crossover operator (RSA), respectively.

This update interval ensured a balanced exploration of the search space, while effectively managing the exploitation of promising solutions through adaptive operator selection. Additionally, to mitigate any potential adverse effects caused by multiple crossover strategies, such as premature convergence, we introduced multiple crossovers, when the number of generations, denoted as *G*, reached or exceeded a threshold, $M_c \times G_{max}$. Here, M_c = 0.15, representing the proportion of generations, at which multiple crossovers were triggered and G_{max} was the maximum number of generations.

In essence, this meant that multiple strategies for crossover were initiated when the number of generations reached 15% of the maximum number of generations. This carefully chosen threshold allowed us to control the application of multiple crossover strategies, preventing them from being used too early in the iterations, which could lead to premature convergence. Instead, they were introduced later in evolution, when the algorithm had sufficient exploration time to traverse the search space freely.

A more comprehensive analysis of the value of Mc and its impact on the algorithm's performance is provided in Section IV, allowing us to gain deeper insights into how the selection of this threshold influenced the ADEMCS convergence rate, exploration-exploitation balance, and overall optimization efficiency.

3.3. Adaptive Differential Evolution Algorithm with Multiple Crossover Strategy Scheme

A limitation of DE algorithms is their dependence on mutation strategies. When it uses a poor mutation strategy, it can become trapped in local optima. Additionally, limitations of crossover operators balancing exploration and exploitation capabilities hinder the discovery of high-quality solutions. Our ADEMCS was designed to address these issues by incorporating an adaptive mutation control mechanism and using multiple strategies in the crossover task to strike a balance between exploration and exploitation. The steps of ADEMCS are set out in Algorithm 1, with a corresponding flowchart in Figure 1. Further details follow:

Step 1: Define parameter values - population size (*NP*), number of dimensions (*d*), boundaries of the search space, maximum number of generations (G_{max}), mutation parameters (*F* and λ) and crossover parameter (*CR*).

Step 2: Generate the initial population randomly by Equation 1.

Step 3: Evaluate the fitness of each target vector in the population.

Step 4: Apply mutation to generate a mutant vector by Equation 10.

Step 5: Evaluate the fitness of the mutant vector.

Step 6: If the fitness of the mutant vector is less than or equal to the fitness of the target vector, increase the mutation control parameter λ by Equation 11, otherwise decrease λ using the same equation.

Step 7: Apply crossover to generate a trial vector, if the number of generations is less than or equal to the product of the multiple crossover parameter and the maximum number of generations, then generate a trial vector by Equation 8, otherwise generate a trial vector as follows:

- Generate a random number, $r \in (0,1]$, if r less than or equal to the success rate of the crossover strategy, generate a trial vector with Equation 8, otherwise generate a trial vector with Equation 13.

Step 8: Evaluate the fitness of the trial vector.

Step 9: Apply a selection operation:

- Comparing the fitness of the trial vector with that of the target vector of the current population. If the trial vector is better than the target vector of the current generation, keep the trial vector to the next population and store the target vector of the current population in an external archive.
- Check the size of the external archive. If it is larger than the allowed size, randomly delete one vector from the external archive to reduce its size to the allowed size.

Step 10: If the number of generations is divisible by five without leaving any remainder, update the value of the success rate of the trial strategy.

Step 11: Repeat step 4-10 until one of the stopping criteria is met.

Algorithm 1 Pseudocode of the ADEMCS Algorithm

```
Initialize the parameter values;
Randomly generate the initial population by using (1);
Evaluate the fitness of the initial population;
G = 1
while (G < G_max)
    if G % 5 = 0
        SCR ← 0.5;
    end if
    for i = 1: NP
        Generate the mutant vector V_i^G by using (10);
        Evaluate the fitness of mutant vector
        if f(V_i^G) ≤ f(X_i^G)
            Increase λ using (11);
        else
            Decrease λ using (11);
        end if
        if G < Mc × Gmax
            Generate the trial vector U_i^G using (8); Flag1 ← 1; Flag2 ← 0;
        else
            if random number (0,1] ≤ SCR
                Generate the trial vector U_i^G using (8); Flag1 ← 1; Flag2 ← 0;
            else
                Generate the trial vector U_i^G using (13); Flag2 ← 1; Flag1 ← 0;
            end if
        end if
        Evaluate the fitness of trial vector
        if f(U_i^G) ≤ f(X_i^G)
            X_i^(G+1) ← U_i^G; Store X_i^G to external archive;
            if Flag1 = 1
                SC1 ← SC1+1;
            end if
            if Flag2 = 1
                SC2 ← SC2+1;
            end if
        else
            X_i^(G+1) ← X_i^G;
        end if
        if the size of the external archive > NP
            Delete one random vector from the external archive;
        end if
    end for
    if G % GN = 0
        SCR ← SC1/(SC1+SC2);
    end if
    SC1 ← 0; SC2 ← 0;
    G = G+1
end while
Output the best fitness of the population;
```

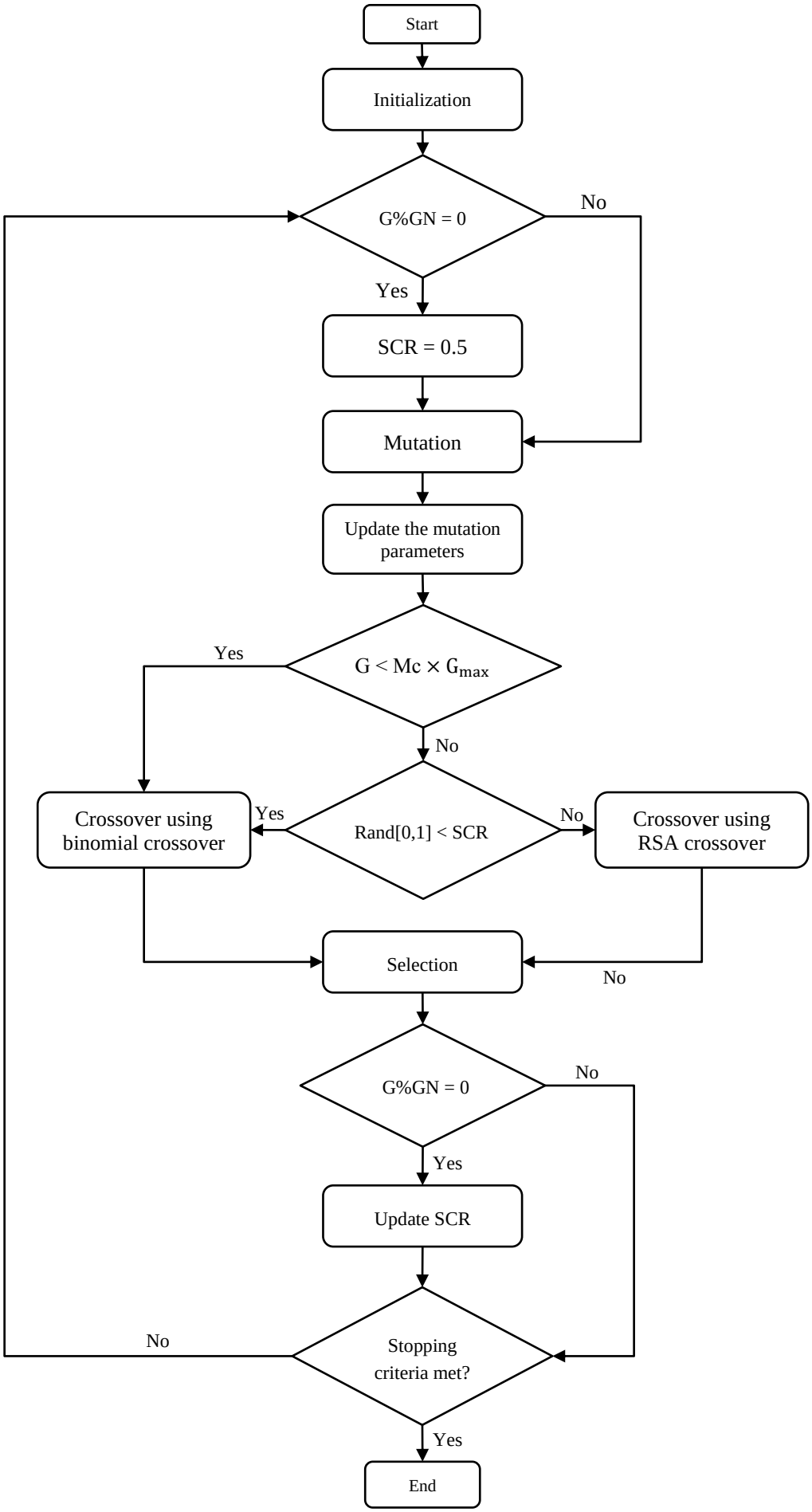

Figure 1. ADEMCS Flowchart

4. Experimental Results and Discussion

We evaluated the performance of our ADEMCS using 32 benchmark functions. These test functions were selected from IEEE CEC2015 [56] and IEEE CEC2017 [57] and divided into two categories: unimodal functions ($f_1 - f_{14}$) and multimodal functions ($f_{15} - f_{32}$). Details of these functions are listed in Table 1.

Table 1. Benchmark functions

Function name	Function description	Domain	$f^*(x)$
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2.$	[-100,100]	0
Elliptic	$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2.$	[-100,100]	0
Bent Cigar	$f_3(x) = \sum_{i=1}^{D} x_i^2 + 10^6 \sum_{i=2}^{D} x_i^2$	[-100,100]	0
Schwefel 1.2	$f_4(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	[-100,100]	0
Schwefel 2.22	$f_5(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[-10,10]	0
Schwefel 2.21	$f_6(x) = max\{\lvert x_i \rvert, 1 \le i \le D\}$	[-100,100]	0
Sum of Different Power	$f_7(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	[-100,100]	0
Sum Squares	$f_8(x) = \sum_{i=1}^{D} i x_i^2$	[-10,10]	0
Discus	$f_9(x) = 10^6 x_i^2 + \sum_{i=1}^{D} x_i^2$	[-100,100]	0
Different Powers	$f_{10}(x) = \sqrt{\sum_{i=1}^{D} \lvert x_i \rvert^{2+4\frac{i-1}{D-1}}}$	[-100,100]	0
Exponential	$f_{11}(x) = -exp(-0.5 \sum_{i=1}^{D} x_i^2)$	[-1,1]	-1
Zakharov	$f_{12}(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5 x_i)^2 + (\sum_{i=1}^{D} 0.5 x_i)^4$	[-5,10]	0
Step	$f_{13}(x) = \sum_{i=1}^{D} (\lvert x_i + 0.5 \rvert)^2$	[-100,100]	0
Noise Quartic	$f_{14}(x) = \sum_{i=1}^{D} i x_i^4 + rand[0,1)$	[-1.28,1.28]	0
Rosenbrock	$f_{15}(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$	[-30,30]	0
Griewank	$f_{16}(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	[-600,600]	0
Rastrigin	$f_{17}(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12,5.12]	0
Apline	$f_{18}(x) = \sum_{i=1}^{D} \lvert x_i \sin x_i + 0.1 x_i \rvert$	[-100,100]	0
Bohachevsky_2	$f_{19}(x) = \sum_{i=1}^{D-1} [x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i)\cos(3\pi x_{i+1}) + 0.3]$	[-100,100]	0
Salomon	$f_{20}(x) = -1\cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	[-100,100]	0
Scaffer2	$f_{21}(x) = \sum_{i=1}^{D} (x_i^2 + x_{i+1}^2)^{0.25} (\sin(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1), x_{D+1} = x_1$	[-100,100]	0
Ackley	$f_{22}(x) = -20exp\left(-0.2\sqrt{\sum_{i=1}^{D} \frac{x_i^2}{D}}\right) - exp\left(\sum_{i=1}^{D} \frac{\cos(2\pi x_i)}{D}\right) + 20 + e$	[-32,32]	0
Weierstrass	$f_{23}(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))] - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k . 0.5)]\right),$ $a = 0.5, b = 3, k_{max} = 20$	[-0.5,0.5]	0
Katsuura	$f_{24}(x) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{\lvert 2^j x_i - round(2^j x_i) \rvert}{2^j}\right)^{\frac{10}{D^{1.2}}}$	[-100,100]	0
HappyCat	$f_{25}(x) = \lvert \sum_{i=1}^{D} x_i^2 - D \rvert^{\frac{1}{4}} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	[-100,100]	0
HGBat	$f_{26}(x) = \lvert (\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2 \rvert^{1/2} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	[-100,100]	0
Scaffer's F6	$f_{27}(x) = \sum_{i=1}^{D} \left(0.5 + \left(\left(\sin\left(\sqrt{x_i^2 + x_{i+1}^2}\right)\right)^2 - 0.5\right) / \left(1 + 0.001(x_i^2 + x_{i+1}^2)\right)^2\right), x_{D+1} = x_1$	[-0.5,0.5]	0
Expanded Scaffer	$f_{28}(x) = f_{27}(x_1, x_2) + f_{27}(x_2, x_3) + \cdots + f_{27}(x_{D-1}, x_D) + f_{27}(x_D, x_1)$	[-5,5]	0
Griewank+Rosenbrock	$f_{29}(x) = f_{16}(f_{15}(x_1, x_2)) + f_{16}(f_{15}(x_2, x_3)) + \cdots + f_{16}(f_{15}(x_{D-1}, x_D)) + f_{16}(f_{15}(x_D, x_1))$	[-5.12,5.12]	0
NCRastrigin	$f_{30}(x) = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10],\ y_i = \begin{cases} x_i, \lvert x_i \rvert < 0.5 \\ \frac{round(2x_i)}{2}, \lvert x_i \rvert \ge 0.5 \end{cases}$	[-10,10]	0
Levy and Montalvo 1	$f_{31}(x) = \frac{\pi}{D}\{10(\sin(\pi y_1))^2 + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10(\sin(\pi y_{i+1}))^2] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y = 1 + \frac{1}{4}(x_i + 1), u(x_i, 10, 100, 4) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	[-10,10]	0
Levy and Montalvo 2	$f_{32}(x) = 0.1\{10(\sin(3\pi x_1))^2 + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + (\sin(3\pi x_{i+1}))^2] + (x_D - 1)^2 [1 + (\sin(2\pi x_D))^2]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	[-5,5]	0

In assessing the effectiveness of ADEMCS, we ran three sets of experiments. The first set focused on sensitivity analysis of the algorithm parameters. In the second set, we analyzed the effect of each proposed strategy on the performance of ADEMCS. These strategies included our new adaptive mutation strategy as well as combining the hunting coordination operator from RSA. Lastly, we evaluated ADEMCS against nine other state-of-the-art DE variants.

The following paragraph explains the experimental setup for all algorithms and examines the results of each experiment in detail.

4.1. Experimental Setup

To ensure a fair comparison, we set dimension D = 30 for both the first and second experiments. However, for the third experiment, we used three different dimensions: 30, 50, and 100. The size of populations was fixed at 100, and the maximum number of generations G was set to 1000. Each algorithm was executed for 30 independent trials to obtain reliable and statistically significant results. For performance evaluation, we used Friedman and Wilcoxon's statistic test [58]. The parameter settings for each comparison algorithm are in Table 2.

Table 2. Parameters of the compared algorithms

Algorithm	Parameter values
DE	$CR = 0.9$, $F = 0.5$
DEGH	$F = 0.3$
EDE	H = 100, $M_F(1:H) = M_{CR}(1:H) = 0.5$, $F = randn(M_F, 0.1)$, $CR = randn(M_{CR}, 0.1)$
CIPDE	c = 0.1, $\mu_F = 0.7$, $\mu_{CR} = 0.5$
EJADE	c = 0.1, $\mu_F = \mu_{CR} = 0.5$, p = 0.05, CR = $randn(\mu_{CR}, 0.1)$, $F = randn(\mu_F, 0.1)$
ATLDE	$CR = 0.9$, $\varepsilon = 0.5$
EBDE	H = 100, $p = 0.1$, $M_F(1:H) = M_{CR}(1:H) = 0.5$, $CR = randn(M_{CR}, 0.1)$, $F = randn(M_F, 0.1)$
LSHADE-SPACMA	$Pbest = 0.11$, H = 1.4, F_{CP} =s 0.5, $Arc\_rate = 5$, c = 0.8
IMMSADE	$\tau = 0.7$, $\lambda \in [0.7,1.0]$, $F \in [0.1,0.8]$, $CR \in [0.3.10]$
DEPSO	$c_1 = c_2 = 2$, $\omega \in [0.4,0.9]$, $CR \in [0.3,1.0]$, $F \in [0.1,0.8]$, $NS_{max} = 5$, $\tau = 0.7$, $SEP = 0.4.NP$, $\gamma = 0.001$
ADE	$F = 0.5$, $\lambda = 0.5$, $CR = 0.9$
ADEMCS	$F = 0.5$, $\lambda = 0.5$, $lr = 0.05$, $CR = 0.9$, C = 0.1, $\alpha = 0.1$, MC = 0.15, GN = 05

4.2. Sensitivity Analysis of Algorithm Parameters

In the first part of the experiments, we analyzed the parameter sensitivity to gain an understanding of their impact on the performance. The analysis involved investigating two crucial aspects: the learning rate of the proposed adaptive mutation strategy and the number of generations initiating and updating the probability of multiple strategies for the crossover operator. By systematically varying these parameters and observing their effects, we aimed to fine-tune the algorithm and identify optimal parameter settings that lead to improved optimization outcomes. The insights gained from this comprehensive sensitivity analysis laid the foundation for the subsequent experiments and provided valuable guidance for enhancing ADEMCS efficiency and effectiveness in solving optimization problems.

4.3. Analyzing the Effect of the Learning Rate

The Friedman rank test results in Figure 2 showed that the learning rate $lr = 0.05$ achieved the highest rank, followed by $lr = 0.3$, $lr = 0.1$, $lr = 0.2$, and $lr = 0.01$. However, when analyzed using Wilcoxon's test, as shown in Table 3, comparing $lr = 0.05$ with the other learning rate parameters, the results indicated that there were no statistically significant differences between the performance of ADEMCS using learning rates ranging from 0.01 to 0.3. The lack of significant differences among the learning rates suggests that all the learning rates have a comparable effect on enhancing ADEMCS performance. The reason behind this lies in how the ADEMCS algorithm dynamically adjusted its mutation parameters based on the learning rate. Regardless of the specific learning rate value used, the algorithm adapted its behavior effectively to optimize solutions. In other words, the algorithm's ability to fine-tune its mutation parameters compensated for any differences introduced by varying learning rates. Consequently, the observed differences in performance between different learning rates were not statistically significant. This suggests that the algorithm's adaptive nature rendered the choice of learning rate less influential, ensuring consistent performance across different learning rates. Overall, we still used $lr = 0.05$ because it has better performance compared to other learning rates, even though statistically it did not have a significant effect.

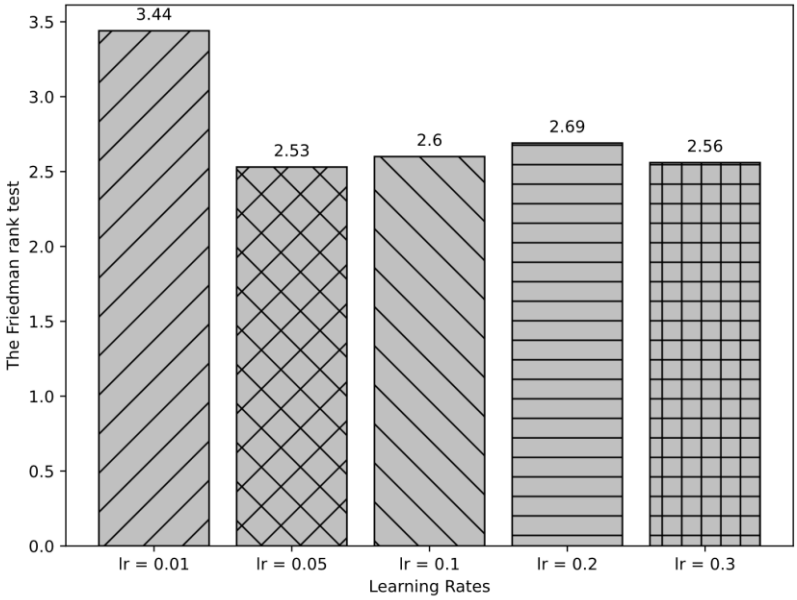

Figure 2. Friedman rank test for different learning rates value

Table 3. Wilcoxon's test for different learning rates value

Lr = 0.05 vs.	p-value	α = 0.05	α = 0.1
Lr = 0.01	5.6E-02	no	yes
Lr = 0.1	7.3E-01	no	no
Lr = 0.2	8.6E-01	no	no
Lr = 0.3	1.6E-01	no	no

4.4. Analyzing the Impact of the Number of Generations on Initiating and Updating the Probability of Multiple Crossover Strategies

Determining the optimal values for the number of generations to initiate the multiple crossover strategies, MC, and the number of generations to update the probability of multiple crossover strategies, GN, is important to maximizing ADEMCS performance. To achieve this, we ran two sets of experiments to find the best combination for MC and GN.

In the first set, MC = 0.15 × Gmax was compared with five different parameters for GN: 3, 5, 10, 15, and 20. The Friedman rank test in Figure 3 showed that the combination MC = 0.15 × Gmax with GN = 5 achieved the highest rank. Following this, MC = 0.15 × Gmax with GN = 10 obtained the second-highest rank; MC = 0.15 × Gmax with GN = 20 ranked third; MC = 0.15 × Gmax with GN = 15 ranked fourth; and MC = 0.15 × Gmax with GN = 3 ranked fifth. Moreover, we analyzed the significant difference of best-performing combination (MC = 0.15 × Gmax with GN = 5) and other combinations using the Wilcoxon test. The results in Table 4 showed that MC = 0.15 × Gmax with GN = 5 was significantly different from MC = 0.15 × Gmax with GN = 3. However, with the other combinations, the differences were not significant. Based on these findings, we used the combination MC = 0.15 × Gmax with GN = 5 to obtain the best ADEMCS performance. The reason for selecting GN = 5 is the need to strike a balance between exploration and exploitation in the search space. Updating the probability of multiple crossover strategies every five generations allows for a sufficiently frequent adaptation to the evolving landscape of the problem space. This interval ensures that ADEMCS maintains a dynamic approach, swiftly responding to changes in the search landscape while avoiding premature convergence or stagnation. In contrast, deviations from the five-generation update interval could lead to suboptimal performance.

Table 4. Wilcoxon's test for different GN; MC = 0.15

MC = 0.15×Gmax with GN = 5 vs.	p-value	α = 0.05	α = 0.1
MC = 0.15×Gmax with GN = 3	2.8E-02	yes	yes
MC = 0.15×Gmax with GN = 10	4.6E-01	no	no
MC = 0.15×Gmax with GN = 15	1.3E-01	no	no
MC = 0.15×Gmax with GN = 20	9.2E-01	no	no

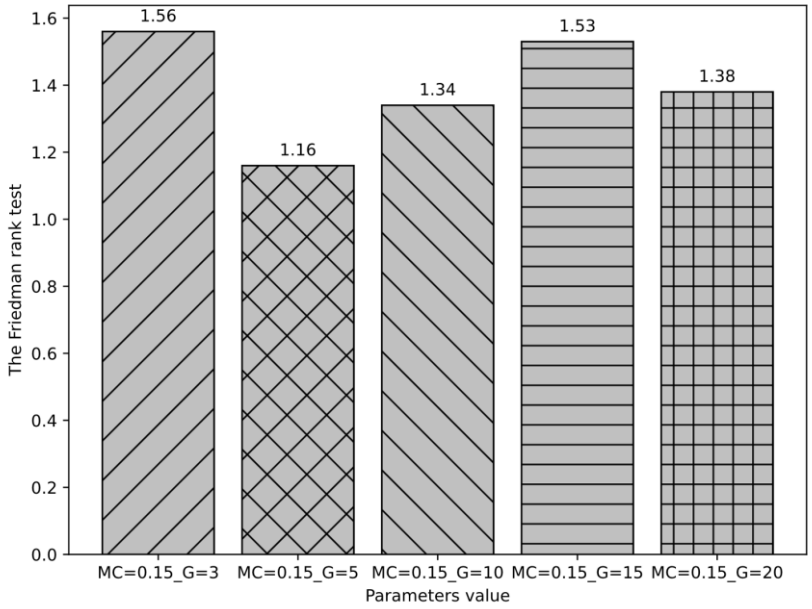

Figure 3. Friedman rank test for different GN; MC = 0.15

In the second set, we focused on GN = 5 and compared it with five different MCs: 0.05, 0.15, 0.25, 0.35, and 0.45. The Friedman rank test result in Figure 4 showed MC = 0.15 × Gmax with GN = 5 achieved the highest rank with a mean rank of 1.19. Following this, MC = 0.25 × Gmax with GN = 5 obtained the second rank; MC = 0.05 × Gmax with GN = 5 ranked third; MC = 0.45 × Gmax with GN = 05 ranked fourth; and MC = 0.35 × Gmax with GN = 5 ranked fifth. The Wilcoxon test in Table 5 indicated that MC = 0.15 × Gmax with GN = 5 exhibited significant improvement compared to MC = 0.05 × Gmax with GN = 5. However, for the other combinations, the results did not show any statistically significant difference. Based on these findings, we can conclude that to achieve the best performance for ADEMCS, the combination value of MC = 0.15 × Gmax with GN = 5 should be used. It can help ADEMCS initiate the multiple crossover strategy at an optimal point during the evolutionary steps. The choice represents 15% of the maximum number of generations, which allows for balanced exploration-exploitation abilities. Initiating multiple crossovers at this proportion of generations ensures that the algorithm has sufficiently explored the search space, reducing the risk of premature convergence. Additionally, setting MC = 0.15 allows for effective utilization of the multiple crossover strategies, enhancing the algorithm's ability to navigate complex optimization landscapes and converge towards high-quality solutions.

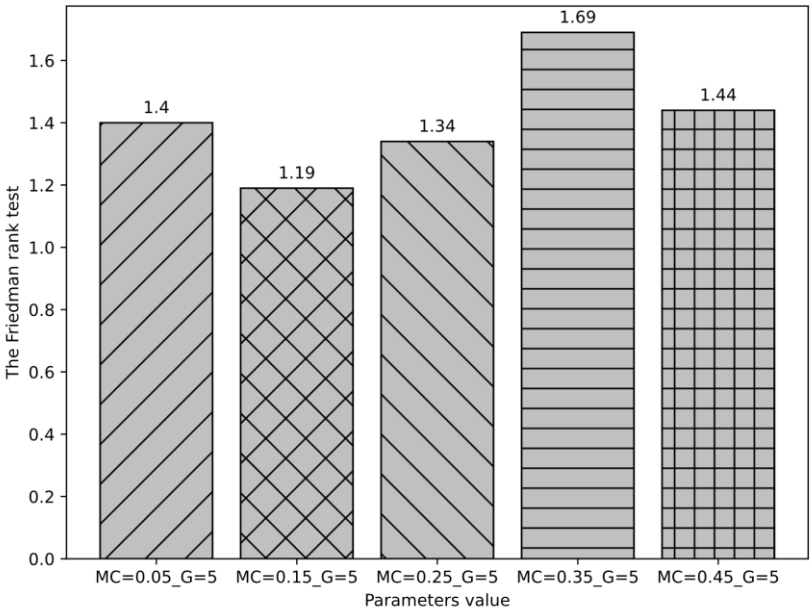

Figure 4. Friedman rank test for different value of MC; GN = 5

Table 5. Wilcoxon's test for different number of MC; GN = 5

MC = 0.15×Gmax with GN = 5 vs.	p-value	α = 0.05	α = 0.1
MC = 0.05×Gmax with GN = 5	2.8E-02	yes	yes
MC = 0.25×Gmax with GN = 5	9.2E-01	no	no
MC = 0.35×Gmax with GN = 5	4.8E-01	no	no
MC = 0.45×Gmax with GN = 5	8.7E-01	no	no

4.5. Effect of Each Proposed Strategy on DE Performance

We evaluated the impact of each new strategy that was adopted in ADEMCS to enhance its performance. For this purpose, we compared the new adaptive mutation strategy (ADE) and the new adaptive differential evolution algorithm with multiple crossover strategy scheme (ADEMCS) against the classical DE. The comparison involved examining the error in global optimum values achieved by each algorithm, analyzing their convergence curves, and conducting the Friedman test and Wilcoxon's test, as shown in Tables 6 and 7, and Figures 5 and 6.

The results in Table 6 indicated that each new strategy contributed to enhancing DE performance. ADE performed better than DE on twenty-nine benchmark functions, with only a slightly worse performance observed in six benchmark functions ($f16$, $f19$, $f22$, $f24$, $f26$, and $f28$). The better performance of ADE over DE was further supported by the results of the Friedman test in Figure 5, where ADE attained the second rank. The convergence results shown in Figure 6 also supported the effectiveness of ADE. The ADE converged much faster than the DE, further demonstrating the impact of the proposed adaptive mutation strategy on enhancing ADE performance. These observations collectively suggested that the introduced adaptive mutation strategy played a significant role in enhancing ADE efficiency and effectiveness in solving complex optimization problems.

ADE's better performance over classical DE was primarily attributed to its adaptive mutation strategy, which dynamically adjusted the first mutation parameter λ based on problem specifics and algorithmic progress while keeping the second mutation parameter $F = 0.5$ to maintain population diversity. This adaptive approach effectively balances exploration and exploitation, allowing ADE to navigate the solution space efficiently. By adopting two mutation operators, ADE efficiently explored diverse solution regions while refining promising solutions, leading to faster convergence compared to classical DE. Moreover, the strategy prevented premature convergence by maintaining population diversity and adapting to unique problem characteristics. Overall, the adaptive mutation strategy enhanced ADE's performance, positioning it as a promising and efficient approach for solving complex optimization problems.

In addition, according to the Friedman test in Figure 6, ADEMCS demonstrated better performance than DE and ADE, achieving the first rank, and significantly outperformed them based on the Wilcoxon test results in Table 7. Based on Table 6, ADEMCS consistently performed better than DE across the majority of optimization problems, only showing slightly worse performance on $f26$. Additionally, when compared to ADE, ADEMCS obtained better results in twenty-nine benchmark problems, with only slightly worse performance observed in two functions ($f25$ and $f29$). By using the multiple crossover strategy, ADEMCS demonstrated the capability to achieve global or near-optimum solutions more effectively. In terms of convergence speed, as shown in Figure 4, we found that ADEMCS converged faster than the other algorithms. This indicated the high efficiency of the multiple crossover strategy in guiding DE towards improved solutions in challenging optimization problems.

The Multiple Crossover Strategy significantly enhanced the ADEMCS performance. Thanks to the Hunting Coordination Operator, ADEMCS had gained an improved local search potential. This operator refined trial vectors based on the current best vector, promoting iterative refinement and facilitating movement toward promising regions in the solution space. The mechanism for updating the crossover probability in each iteration effectively balanced exploration and exploitation. Additionally, the controlled application of multiple crossover strategies, initiated after a set threshold of generations, prevented premature convergence and ensured optimal use of these strategies. Through these enhancements, ADEMCS achieved superior convergence rates and outperformed both DE and ADE in various benchmark functions, showcasing its effectiveness in solving complex optimization problems.

Table 6. Results for DE, ADE, and ADEMCS

Function Number	DE	ADE	ADEMCS	Function Number	DE	ADE	ADEMCS
	Mean and SD	Mean and SD	Mean and SD		Mean and SD	Mean and SD	Mean and SD
$f1$	7.98E-08	1.39E-62	**0.00E+00**	$f17$	1.86E+02	1.68E+02	**0.00E+00**
	4.64E-08	3.36E-62	**0.00E+00**		1.11E+01	1.23E+01	**0.00E+00**
$f2$	5.52E-05	3.44E-57	**0.00E+00**	$f18$	4.47E-02	7.79E-03	**0.00E+00**
	3.31E-05	8.16E-57	**0.00E+00**		3.44E-02	1.11E-02	**0.00E+00**
$f3$	6.25E-02	1.43E-53	**0.00E+00**	$f19$	4.54E-06	1.33E-01	**0.00E+00**
	4.57E-02	3.83E-53	**0.00E+00**		4.47E-06	3.67E-01	**0.00E+00**
$f4$	5.05E+01	1.32E-16	**0.00E+00**	$f20$	3.05E-01	1.84E-01	**0.00E+00**
	2.33E+01	3.65E-16	**0.00E+00**		2.09E-02	3.18E-02	**0.00E+00**
$f5$	7.01E-04	1.34E-30	**0.00E+00**	$f21$	8.42E+01	7.06E+01	**0.00E+00**
	2.39E-04	4.10E-30	**0.00E+00**		9.43E+00	5.01E+00	**0.00E+00**
$f6$	4.41E-01	1.48E-07	**0.00E+00**	$f22$	9.18E-05	1.45E-01	**0.00E+00**
	7.00E-01	6.22E-08	**0.00E+00**		2.62E-05	3.82E-01	**0.00E+00**
$f7$	1.69E-13	1.14E-100	**0.00E+00**	$f23$	6.40E-02	5.26E-02	**0.00E+00**
	5.67E-13	6.14E-100	**0.00E+00**		1.24E-02	2.88E-01	**0.00E+00**
$f8$	1.02E-08	1.02E-63	**0.00E+00**	$f24$	1.94E+00	2.23E+00	**0.00E+00**
	5.22E-09	1.68E-63	**0.00E+00**		2.63E-01	2.52E-01	**0.00E+00**
$f9$	1.40E-07	1.11E-61	**0.00E+00**	$f25$	4.68E-01	**3.35E-01**	3.51E-01
	9.91E-08	1.67E-61	**0.00E+00**		6.13E-02	**5.97E-02**	6.05E-02
$f10$	6.01E-06	9.64E-40	**0.00E+00**	$f26$	**3.60E-01**	4.64E-01	3.99E-01
	2.76E-06	2.99E-39	**0.00E+00**		**1.29E-01**	1.80E-01	1.11E-01
$f11$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$f27$	3.97E-12	**0.00E+00**	**0.00E+00**
	1.84E-12	**7.71E-17**	**0.00E+00**		2.94E-12	**0.00E+00**	**0.00E+00**
$f12$	5.18E-08	1.30E-61	**0.00E+00**	$f28$	4.47E-12	6.56E+00	**0.00E+00**
	7.27E-08	1.72E-61	**0.00E+00**		1.99E-12	4.74E-01	**0.00E+00**
$f13$	9.40E-08	**0.00E+00**	**0.00E+00**	$f29$	1.84E+00	**1.16E+00**	1.80E+00
	6.04E-08	**0.00E+00**	**0.00E+00**		8.78E-01	**1.99E-01**	3.55E-09
$f14$	1.45E-02	2.10E-03	**1.52E-05**	$f30$	1.57E+02	1.36E+02	**0.00E+00**
	3.91E-03	8.23E-04	**1.03E-05**		1.08E+01	1.28E+01	**0.00E+00**
$f15$	2.25E+01	3.40E+00	**2.43E+00**	$f31$	6.90E-11	**1.57E-32**	**1.57E-32**
	6.92E-01	2.43E+00	**2.57E+00**		6.20E-11	**5.57E-48**	**5.57E-48**
$f16$	2.50E-04	4.51E-03	**0.00E+00**	$f32$	8.86E-10	**1.35E-31**	**1.35E-31**
	1.35E-03	7.17E-03	**0.00E+00**		1.38E-09	**4.45E-47**	**4.45E-47**

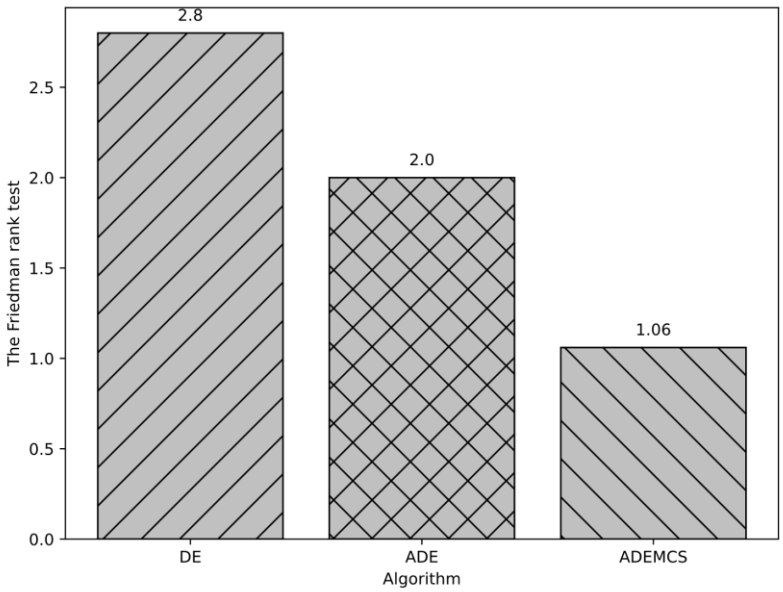

Figure 5. Friedman rank test for DE, ADE, and ADEMCS

Table 7. Wilcoxon's test for ADEMCS, DE, and ADE

ADEMCS vs.	p-value	α = 0.05	α = 0.1
DE	7.950E-07	yes	yes
ADE	1.880E-04	yes	yes

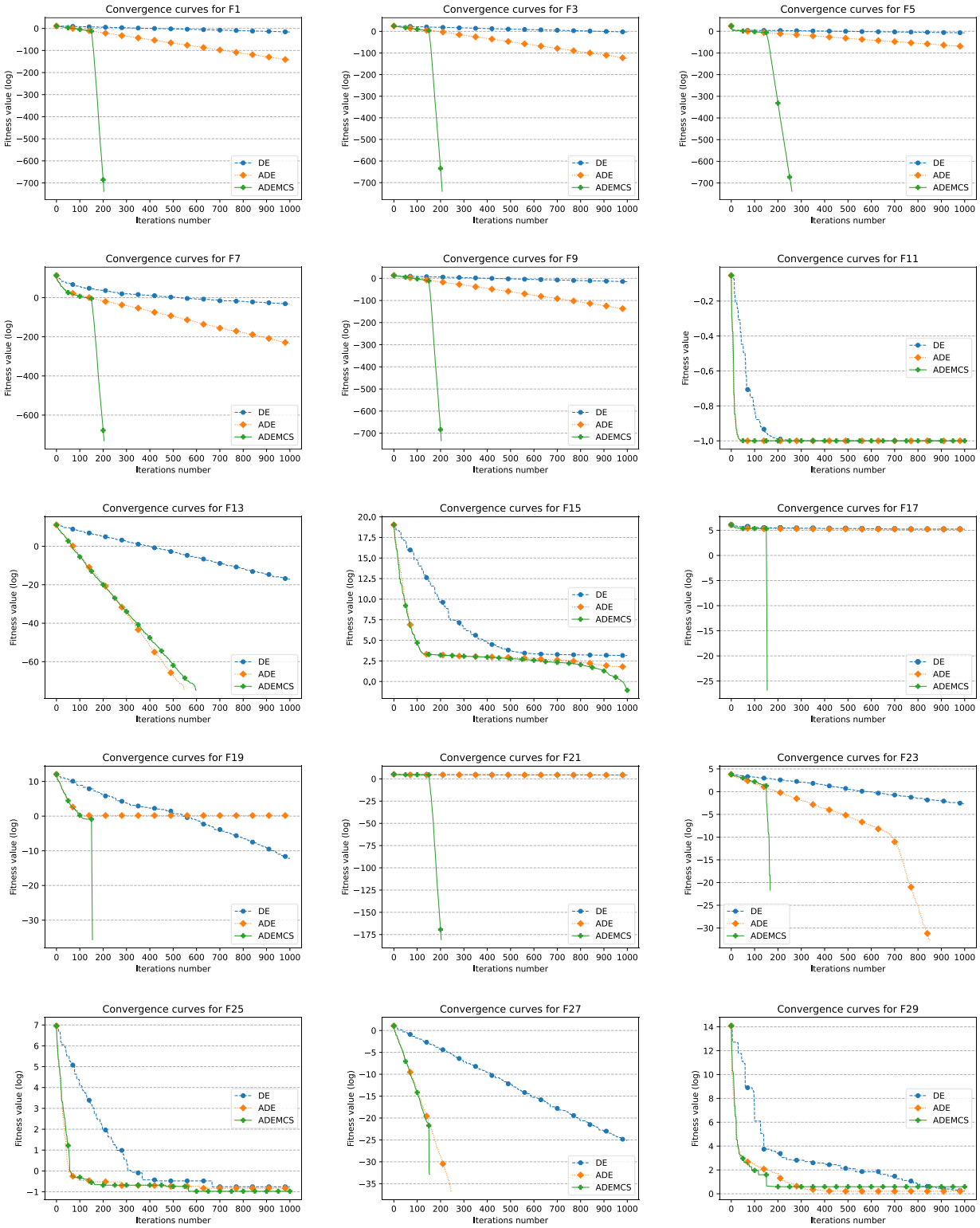

Figure 6. Convergence curves for DE, ADE, and ADEMCS

4.6. Comparison with State-of-the-Art DE Variants

In our third experiment, we ran a comparative analysis of our proposed algorithm ADEMCS with nine state-of-the-art DE variants, namely IMMSADE [59], CIPDE [60], EBDE [61], EDE [61], EJADE [4], LSHADE-SPACMA [50],

DEPSO [37], ATLDE [62], and DEGH [63]. The results of these compared algorithms were sourced from Zhong et al. [63]. The main goal of this experiment was to thoroughly evaluate the performance of ADEMCS in solving various optimization problems compared to other established algorithms. To accomplish this goal, we assessed the error-best global values and compared their performances using metric addition, subtraction, and equality tests. Additionally, we employed two statistical tests, namely Friedman and Wilcoxon tests, to gain deeper insights into the overall performance differences between ADEMCS and the other DE variants. In the subsequent paragraph, we will elaborate on the funding for this research.

Table 9 displays the results of the experiment run with a dimension D = 30. Our new proposed algorithm, ADEMCS, demonstrated its capability to achieve global or near-global optimal solutions across twenty-nine benchmark functions (*f*1-*f*14, *f*16-*f*24, *f*27-*f*32). The best-known solution of function *f*15 was obtained by EJADE, while the best-known solutions of functions *f*25 *were* achieved by CIPDE and *f*26 *were* gained by EDE. Furthermore, as indicated in Table 8, ADEMCS exhibited superior performance compared to the nine state-of-the-art DE variants. In particular, ADEMCS obtained a better result than DEGH on 8 functions, EDE on 25 functions, CIPDE on 23 functions, EJADE on 29 functions, ATLDE on 26 functions, EBDE on 26 functions, LSAHDE-SPACMA on 24 functions, IMMSADE on 28 functions, and DEPSO on 27 functions. Additionally, we analyzed the performance of our proposed method, ADEMCS, in solving both unimodal (*f*1-*f*14) and multimodal functions (*f*15-*f*32). According to the results in Table 9, it was evident that ADEMCS achieved superior performance on all unimodal functions and consistently approached or attained the global optimum value for most unimodal functions, with the exception of *f14*, where it achieved a near-global optimal value. Notably, among the comparison methods, DEGH also demonstrated commendable results by locating the global optimum value for most unimodal functions, only slightly worse than our proposed method, ADECMS, on *f*13 and *f*14. The comparison demonstrated that ADEMCS has better performance across a wide range of optimization tasks.

It is important to note that ADEMCS employed a different strategy compared to DEGH, characterized by two main differences. Firstly, while ADEMCS adopted a single adaptive mutation strategy, DEGH used multiple (four) adaptive mutation strategies. Secondly, in the crossover steps, ADEMCS integrated multiple (two) crossover strategies, whereas DEGH used a single crossover strategy. Essentially, while ADEMCS aimed to enhance DE performance through modifications to the crossover operator, DEGH pursued the same goal through adjustments to mutation strategies. The rationale behind our decision to enhance the crossover operator stemmed from our observation of DE, where we found that the crossover operator significantly impacted DE's exploration ability but often lacked exploitation capabilities. To address this imbalance, we incorporated multiple crossover strategies by leveraging the reptile search algorithm, known for its good exploitation abilities. This enhancement of the crossover operator demonstrated good performance in terms of overall ADEMCS effectiveness.

Regarding multimodal functions, ADEMCS achieved better performance over other comparison algorithms. Specifically, ADEMCS was able to obtain the global or near-global optimum value for 15 out of 18 multimodal functions, showing only slightly worse performance on 3 multimodal functions (*f*15, *f*25, and *f*26). For *f*15, the best result was obtained by EJADE, although it only achieved a near-global optimum value. It is important to note that *f*15, known as the Rosenbrock function, presented complex multimodal characteristics, making convergence towards the global minimum problematic for gradient descent methods due to its narrow, curved valley and the presence of flat regions. Furthermore, for *f*25, the best result was achieved by CIPDE, while EDE obtained the best result on *f*26.

In addition, the Friedman test shown in Figure 7 further supports the good performance of ADEMCS, as it achieved the first rank. Finally, we used a Wilcoxon test to analyze the differences in performance of ADEMCS compared to other algorithms, for D = 30. The findings from Table 10 revealed that ADEMCS exhibited significantly higher performance compared to the other algorithms, affirming its capabilities and effectiveness.

Table 8. Performance of ADEMCS with nine state-of-the-art DE variants using Addition (+), Subtraction (-), and Equality (=) Tests

ADEMCS vs.	D = 30 +/-/=	D = 50 +/-/=	D = 100 +/-/=
DEGH	8/0/24	7/1/24	5/4/23
EDE	25/2/5	30/2/0	28/4/0
CIPDE	23/3/6	27/4/1	28/4/0
EJADE	29/2/1	29/3/0	28/4/0
ATLDE	26/0/6	26/0/6	26/0/6
EBDE	26/2/4	30/2/0	24/2/6
LSAHDE-SPACMA	24/2/6	27/3/2	28/4/0
IMMSADE	28/0/4	29/0/3	29/1/2
DEPSO	27/0/5	25/0/7	24/0/8

Table 9. Results for ADEMCS with nine state-of-the-art DE variants (D = 30)

Function Number	DEGH	EDE	CIPDE	EJADE	ATLDE	EBDE	LSHADE-SPACMA	IMM SADE	DEPSO	ADEMCS
	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD
$f1$	**0.00E+00**	2.32E-38	2.91E-43	2.10E-39	1.50E-54	1.40E-48	1.60E-63	2.21E-29	3.35E-93	**0.00E+00**
	0.00E+00	3.98E-38	1.19E-42	3.34E-39	2.50E-54	5.00E-48	2.74E-63	9.12E-29	1.60E-92	**0.00E+00**
$f2$	**0.00E+00**	1.71E-33	3.62E-40	1.06E-34	1.41E-50	2.27E-43	1.83E-54	1.58E-23	6.00E-96	**0.00E+00**
	0.00E+00	5.63E-33	1.16E-39	3.92E-34	1.92E-50	6.50E-43	3.29E-54	8.61E-23	2.26E-95	**0.00E+00**
$f3$	**0.00E+00**	7.84E-31	3.52E-38	1.02E-31	4.17E-48	3.99E-41	1.07E-51	8.11E-22	7.32E-93	**0.00E+00**
	0.00E+00	2.25E-30	1.09E-37	4.32E-31	1.34E-47	1.45E-40	4.73E-51	3.43E-21	4.01E-92	**0.00E+00**
$f4$	**0.00E+00**	5.94E-05	2.35E-10	8.10E-10	5.68E-53	1.07E-12	4.24E-31	3.07E+00	4.49E-91	**0.00E+00**
	0.00E+00	1.08E-04	8.88E-10	1.37E-09	2.09E-52	2.78E-12	1.55E-30	7.02E+00	2.46E-90	**0.00E+00**
$f5$	**0.00E+00**	5.55E-21	2.27E-20	6.79E-17	2.05E-26	6.77E-26	1.56E-33	8.87E-16	3.89E-50	**0.00E+00**
	0.00E+00	9.88E-21	7.54E-20	1.63E-16	3.84E-26	1.51E-25	1.39E-33	2.95E-15	2.08E-49	**0.00E+00**
$f6$	**0.00E+00**	6.29E-03	1.45E-08	2.01E-04	5.83E-23	7.09E-08	7.05E-23	3.55E-08	1.73E-49	**0.00E+00**
	0.00E+00	7.53E-03	3.89E-08	2.46E-04	7.42E-23	9.93E-08	1.66E-22	6.88E-08	8.89E-49	**0.00E+00**
$f7$	**0.00E+00**	1.86E-35	3.18E-55	5.59E-71	2.68E-115	2.86E-64	2.61E-69	1.33E-83	3.44E-113	**0.00E+00**
	0.00E+00	7.83E-35	1.74E-54	2.13E-70	1.06E-114	1.57E-63	1.36E-68	7.22E-83	1.87E-112	**0.00E+00**
$f8$	**0.00E+00**	6.11E-39	5.84E-45	3.06E-40	1.03E-54	4.81E-50	8.50E-68	3.29E-26	2.81E-99	**0.00E+00**
	0.00E+00	7.36E-39	2.33E-44	5.20E-40	3.32E-54	9.75E-50	1.23E-67	1.78E-25	1.53E-98	**0.00E+00**
$f9$	**0.00E+00**	1.66E-36	1.08E-44	4.08E-37	1.04E-53	9.04E-48	9.90E-58	1.86E-27	1.13E-99	**0.00E+00**
	0.00E+00	2.78E-36	2.56E-44	1.79E-36	2.10E-53	2.00E-47	4.49E-57	5.96E-27	4.45E-99	**0.00E+00**
$f10$	**0.00E+00**	1.88E-16	1.79E-27	4.33E-25	1.43E-35	3.08E-25	4.00E-26	1.34E-20	1.04E-54	**0.00E+00**
	0.00E+00	4.26E-16	2.90E-27	5.22E-25	4.98E-35	9.17E-25	9.10E-26	7.29E-20	5.64E-54	**0.00E+00**
$f11$	**0.00E+00**	7.40E-18	**0.00E+00**	1.52E-16	**0.00E+00**	8.51E-17	**0.00E+00**	-5.00E-01	**0.00E+00**	**0.00E+00**
	0.00E+00	2.82E-17	**0.00E+00**	6.17E-17	**0.00E+00**	4.78E-17	**0.00E+00**	5.04E-01	**0.00E+00**	**0.00E+00**
$f12$	**0.00E+00**	2.62E-35	2.76E-48	8.22E-38	4.36E-55	7.27E-48	1.17E-67	6.02E-27	7.93E-99	**0.00E+00**
	0.00E+00	7.06E-35	7.81E-48	2.21E-37	1.59E-54	1.23E-47	2.19E-67	3.16E-26	4.01E-98	**0.00E+00**
$f13$	6.17E-20	1.75E-32	**0.00E+00**	**0.00E+00**	4.62E+00	**0.00E+00**	**0.00E+00**	5.69E-03	2.21E+00	**0.00E+00**
	7.02E-20	3.27E-33	**0.00E+00**	**0.00E+00**	1.00E+00	**0.00E+00**	**0.00E+00**	2.45E-03	3.15E-01	**0.00E+00**
$f14$	3.90E-03	4.49E-03	1.60E-03	2.19E-03	1.48E-03	3.09E-03	2.56E-03	3.19E-01	4.14E-01	**1.52E-05**
	2.04E-03	2.21E-03	1.20E-03	1.09E-03	6.33E-04	1.45E-03	1.53E-03	2.34E-01	2.81E-01	**1.03E-05**
$f15$	2.56E+01	1.29E+01	7.46E-01	**6.67E-01**	2.89E+01	7.41E-01	9.52E+00	2.58E+01	2.80E+01	2.43E+00
	3.93E-01	1.08E+00	7.49E-01	**1.51E+00**	3.24E-02	1.43E+00	1.68E+00	2.12E-01	3.35E-01	2.57E+00
$f16$	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.47E-04	**0.00E+00**	6.15E-03	9.04E-04	**0.00E+00**	1.28E-03	**0.00E+00**
	0.00E+00	**0.00E+00**	**0.00E+00**	1.35E-03	**0.00E+00**	1.06E-02	2.86E-03	**0.00E+00**	4.96E-03	**0.00E+00**
$f17$	**0.00E+00**	9.09E+00	1.10E+00	2.43E+01	6.09E+00	5.85E-01	9.05E+00	4.38E+01	3.88E-09	**0.00E+00**
	0.00E+00	1.80E+00	1.07E+00	1.29E+01	3.34E+01	1.06E+00	2.71E+00	2.96E+01	2.13E-08	**0.00E+00**
$f18$	**0.00E+00**	1.36E-02	6.10E-03	1.09E-12	1.56E-25	4.85E-16	3.18E-15	4.06E-14	5.28E-51	**0.00E+00**
	0.00E+00	6.64E-03	1.29E-03	5.90E-12	4.48E-25	5.14E-16	1.57E-14	1.55E-13	1.36E-50	**0.00E+00**
$f19$	**0.00E+00**	7.40E-18	**0.00E+00**	2.04E-16	**0.00E+00**	3.99E-01	2.75E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	2.82E-17	**0.00E+00**	1.64E-16	**0.00E+00**	6.27E-01	1.05E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f20$	**0.00E+00**	1.83E-01	1.07E-01	2.93E-01	5.00E-02	2.43E-01	3.63E-01	1.42E-01	9.99E-02	**0.00E+00**
	0.00E+00	3.79E-02	2.54E-02	6.91E-02	5.08E-02	5.04E-02	8.50E-02	4.78E-02	1.21E-07	**0.00E+00**
$f21$	**0.00E+00**	3.70E+00	2.53E+00	1.29E+01	1.69E+01	2.69E+00	1.24E+00	3.09E+01	6.12E-02	**0.00E+00**
	0.00E+00	7.72E-01	7.45E-01	8.01E+00	2.42E+01	5.36E-01	3.30E-01	7.16E+00	8.24E-02	**0.00E+00**
$f22$	**0.00E+00**	5.68E-15	3.55E-15	2.29E-14	3.55E-15	6.28E-15	3.55E-15	4.38E-15	0.00E+00	**0.00E+00**
	0.00E+00	1.77E-15	0.00E+00	6.31E-15	0.00E+00	1.53E-15	0.00E+00	3.89E-15	0.00E+00	**0.00E+00**
$f23$	**0.00E+00**	**0.00E+00**	1.10E-12	8.88E-02	**0.00E+00**	1.09E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	**0.00E+00**	4.30E-12	7.20E-02	**0.00E+00**	1.65E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f24$	**0.00E+00**	1.83E-02	1.89E-02	7.36E-02	1.40E+00	7.86E-03	3.78E-02	6.24E-01	6.49E-01	**0.00E+00**
	0.00E+00	1.73E-03	3.73E-03	4.85E-02	1.16E+00	2.32E-03	4.31E-03	1.04E-01	1.05E-01	**0.00E+00**
$f25$	3.59E-01	2.19E-01	**1.40E-01**	2.38E-01	5.80E-01	2.30E-01	2.53E-01	4.55E-01	8.32E-01	3.51E-01
	4.85E-02	2.68E-02	**3.66E-02**	6.01E-02	1.20E-01	5.30E-02	3.94E-02	5.40E-02	8.93E-02	6.05E-02
$f26$	4.18E-01	**3.48E-01**	3.71E-01	4.09E-01	4.40E-01	4.39E-01	3.95E-01	3.93E-01	4.98E-01	3.99E-01
	2.93E-02	**9.30E-02**	8.88E-02	1.33E-01	3.62E-02	1.94E-01	1.51E-01	3.92E-02	5.87E-03	1.11E-01
$f27$	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.48E-17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	**0.00E+00**	**0.00E+00**	8.11E-17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f28$	**0.00E+00**	7.37E-01	5.49E-01	5.70E-01	**0.00E+00**	5.43E-01	4.12E-01	3.43E+00	4.38E+00	**0.00E+00**
	0.00E+00	8.86E-02	9.74E-02	6.53E-02	**0.00E+00**	7.57E-02	9.71E-02	4.78E-01	1.12E+00	**0.00E+00**
$f29$	7.67E+00	2.80E+00	2.01E+00	2.95E+00	1.32E+01	2.23E+00	3.18E+00	1.21E+01	1.15E+01	**1.80E+00**
	5.40E-01	2.07E-01	2.70E-01	3.78E-01	1.30E+00	1.82E-01	1.11E+00	8.27E-01	3.77E-01	**3.55E-09**
$f30$	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.41E+01	6.97E+00	1.10E-07	1.39E+01	1.68E+01	**0.00E+00**	**0.00E+00**
	0.00E+00	**0.00E+00**	**0.00E+00**	1.34E+01	2.69E+01	6.02E-07	4.21E+00	8.02E+00	**0.00E+00**	**0.00E+00**
$f31$	1.04E-22	**1.57E-32**	**1.57E-32**	2.78E-32	2.52E-01	**1.57E-32**	**1.57E-32**	2.61E-04	6.99E-02	**1.57E-32**
	1.26E-22	**5.57E-48**	**5.57E-48**	1.35E-32	9.43E-02	**5.57E-48**	**5.57E-48**	8.77E-05	2.08E-02	**5.57E-48**
$f32$	1.19E-21	1.42E-31	1.36E-31	**1.35E-31**	1.20E+00	**1.35E-31**	**1.35E-31**	9.31E-03	3.54E-01	**1.35E-31**
	2.02E-21	1.24E-33	2.77E-33	**6.68E-47**	3.83E-01	**6.68E-47**	**6.68E-47**	3.99E-03	8.20E-02	**4.45E-47**

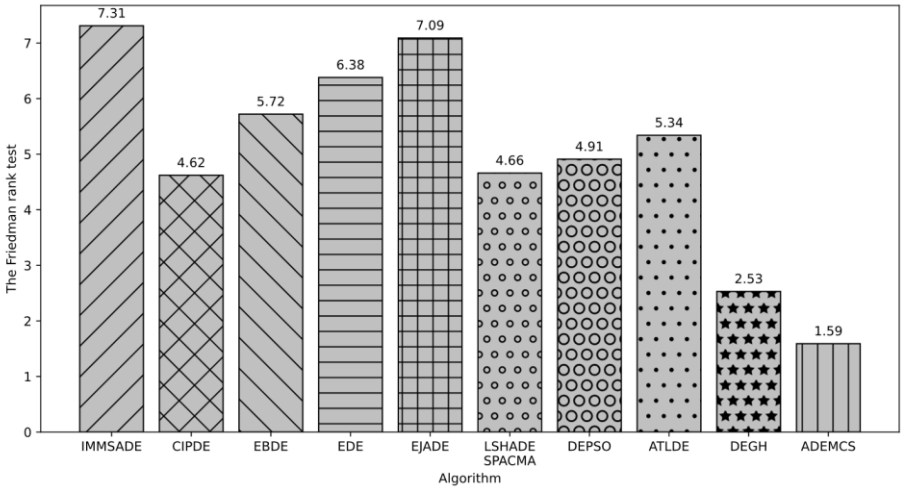

Figure 7. Friedman rank test for ADEMCS with nine state-of-the-art DE variants (D = 30)

Table 10. Wilcoxon's test for ADEMCS vs. nine state-of-the-art DE variants (D = 30).

ADEMCS vs.	p-value	α = 0.05	α = 0.1
DEGH	5.1E-02	no	yes
EDE	3.8E-03	yes	yes
CIPDE	6.6E-03	yes	yes
EJADE	2.5E-04	yes	yes
ATLDE	5.6E-06	yes	yes
EBDE	6.6E-03	yes	yes
LSAHDE-SPACMA	3.1E-03	yes	yes
IMMSADE	2.1E-05	yes	yes
DEPSO	5.6E-06	yes	yes

For analyzing the performance of ADEMCS with D = 50, Table 11 demonstrated that our new algorithm achieved global or near-global optimal solutions across twenty-eight benchmark functions (*f*1-*f*14, *f*16-*f*24, *f*26-*f*30). Notably, function *f*15 was achieved by EJADE, while functions *f*25 and *f*32 were achieved by CIPDE and LSHADE-SPACMA, respectively. Additionally, the best result for function *f*31 was achieved by both CIPDE and SPACMA. Specifically, ADEMCS obtained better results than DEGH on 7 functions, EDE on 30 functions, CIPDE on 27 functions, EJADE on 29 functions, ATLDE on 26 functions, EBDE on 30 functions, LSAHDE-SPACMA on 27 functions, IMMSADE on 29 functions, and DEPSO on 25 functions. This comparison underscored the good performance of ADEMCS across a diverse set of optimization tasks.

For both unimodal and multimodal functions with D = 50, our new method, ADEMCS, demonstrated better performance by achieving better results for all unimodal functions compared to other comparison algorithms. In the case of multimodal functions, ADEMCS performed better than other comparison algorithms on 14 out of 18 multimodal functions. It was slightly worse than some other algorithms for functions *f*15, *f*25, *f*31, and *f*32.

In addition, after conducting further analysis of ADEMCS's performance in solving both unimodal and multimodal functions with D = 50 and comparing its performance with that of D = 30, a slight decrease in performance was observed for multimodal functions, while ADEMCS showed stable and robust performance for unimodal functions. This decrease in performance occurred because ADEMCS only used one adaptive mutation strategy, which might not be enough when dealing with multimodal functions with high dimensionality. Overall, despite this limitation, we can conclude that ADEMCS yielded competitive results compared to other comparison algorithms in solving both unimodal and multimodal functions with D = 50.

The results from the Friedman test presented in Figure 8 further reinforced the performance of ADEMCS. It achieved the first rank, providing robust evidence of its good performance over the other comparison algorithms. The findings from the Friedman test presented ADEMCS as the top-performing algorithm. Finally, the Wilcoxon test was conducted to analyze the differences in performance, and the results are presented in Table 12. The findings confirmed that ADEMCS had significantly better performance compared to the other comparison algorithms.

Table 11. Results for ADEMCS with nine state-of-the-art DE variants (D = 50)

Function Number	DEGH	EDE	CIPDE	EJADE	ATLDE	EBDE	LSHADE-SPACMA	IMM SADE	DEPSO	ADEMCS
	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD
$f1$	**0.00E+00**	4.70E-20	6.43E-34	5.33E-21	7.20E-54	3.56E-25	7.12E-36	2.07E-20	2.92E-95	**0.00E+00**
	0.00E+00	1.41E-19	1.54E-33	9.49E-21	1.18E-53	6.06E-25	1.63E-35	1.03E-19	1.57E-94	**0.00E+00**
$f2$	**0.00E+00**	1.68E-15	7.56E-29	4.17E-16	2.30E-49	1.26E-19	3.11E-12	1.19E-16	2.25E-94	**0.00E+00**
	0.00E+00	7.21E-15	1.14E-28	7.11E-16	6.00E-49	5.82E-19	1.32E-11	6.46E-16	7.88E-94	**0.00E+00**
$f3$	**0.00E+00**	4.05E-12	1.02E-27	6.74E-14	1.38E-47	3.18E-17	2.87E-24	3.07E-15	7.03E-87	**0.00E+00**
	0.00E+00	1.48E-11	2.17E-27	2.25E-13	3.17E-47	9.51E-17	9.59E-24	1.63E-14	3.85E-86	**0.00E+00**
$f4$	**0.00E+00**	3.73E+00	3.32E-02	4.96E-02	1.59E-52	8.06E-03	2.48E-03	5.77E+02	1.56E-90	**0.00E+00**
	0.00E+00	2.75E+00	3.51E-02	3.69E-02	3.48E-52	7.01E-03	3.24E-03	1.03E+03	8.42E-90	**0.00E+00**
$f5$	**0.00E+00**	1.25E-12	1.12E-17	3.02E-08	2.84E-26	3.86E-13	1.18E-23	1.97E-12	1.37E-51	**0.00E+00**
	0.00E+00	1.66E-12	1.10E-17	4.42E-08	5.51E-26	7.37E-13	6.48E-24	7.76E-12	6.72E-51	**0.00E+00**
$f6$	**0.00E+00**	1.65E+00	3.72E-01	1.70E+00	8.70E-23	6.26E-01	1.07E-01	2.30E-05	5.06E-47	**0.00E+00**
	0.00E+00	7.73E-01	2.37E-01	7.41E-01	1.73E-22	3.56E-01	6.23E-02	4.93E-05	2.75E-46	**0.00E+00**
$f7$	**0.00E+00**	1.46E-06	2.90E-27	8.34E-33	1.67E-113	1.91E-21	4.44E-25	1.01E-65	4.10E-115	**0.00E+00**
	0.00E+00	6.55E-06	1.58E-26	2.25E-32	6.21E-113	9.86E-21	1.80E-24	5.47E-65	2.25E-114	**0.00E+00**
$f8$	**0.00E+00**	4.52E-21	9.36E-35	2.19E-21	5.40E-54	1.44E-25	2.42E-41	2.68E-24	2.26E-97	**0.00E+00**
	0.00E+00	1.03E-20	8.78E-35	4.08E-21	2.48E-53	2.50E-25	1.11E-40	1.32E-23	9.99E-97	**0.00E+00**
$f9$	**0.00E+00**	1.96E-19	4.82E-33	7.29E-20	2.89E-53	4.23E-24	2.58E-26	2.09E-23	4.77E-95	**0.00E+00**
	0.00E+00	6.00E-19	7.82E-33	1.52E-19	1.03E-52	1.19E-23	6.25E-26	1.12E-22	2.61E-94	**0.00E+00**
$f10$	**0.00E+00**	3.72E-08	3.12E-16	5.97E-13	1.80E-32	4.37E-12	8.96E-11	2.14E-15	4.43E-53	**0.00E+00**
	0.00E+00	7.10E-08	3.43E-16	6.78E-13	2.98E-32	5.19E-12	2.42E-10	1.16E-14	2.42E-52	**0.00E+00**
$f11$	**0.00E+00**	1.22E-16	9.25E-17	6.22E-16	**0.00E+00**	1.44E-16	**0.00E+00**	-5.00E-01	**0.00E+00**	**0.00E+00**
	0.00E+00	3.39E-17	4.21E-17	1.76E-16	**0.00E+00**	5.17E-17	**0.00E+00**	5.04E-01	**0.00E+00**	**0.00E+00**
$f12$	**0.00E+00**	3.31E-19	4.90E-32	8.00E-21	1.99E-54	1.80E-24	6.06E-47	5.70E-17	1.70E-98	**0.00E+00**
	0.00E+00	4.44E-19	1.11E-31	2.01E-20	4.98E-54	2.24E-24	1.83E-46	3.08E-16	9.11E-98	**0.00E+00**
$f13$	2.55E-10	2.06E-20	2.95E-32	2.27E-21	9.53E+00	3.48E-25	5.75E-32	3.62E-02	6.31E+00	**1.59E-32**
	2.36E-10	5.50E-20	1.23E-32	2.64E-21	8.98E-01	5.47E-25	1.69E-32	1.15E-02	4.62E-01	**3.16E-32**
$f14$	5.07E-03	7.61E-03	2.77E-03	9.90E-03	1.42E-03	1.20E-02	6.24E-03	4.04E-01	3.57E-01	**2.14E-05**
	2.53E-03	3.30E-03	1.52E-03	3.75E-03	6.74E-04	5.62E-03	2.86E-03	2.45E-01	2.41E-01	**2.05E-05**
$f15$	4.62E+01	4.52E+01	3.32E+01	**2.98E+01**	4.89E+01	3.98E+01	3.71E+01	4.62E+01	4.81E+01	3.34E+01
	2.84E-01	1.29E+01	1.46E+01	**1.13E+01**	3.81E-02	1.89E+01	1.31E+01	3.55E-01	3.48E-01	3.58E+00
$f16$	**0.00E+00**	4.93E-04	1.23E-03	1.48E-03	**0.00E+00**	4.51E-03	4.76E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	1.88E-03	3.22E-03	3.40E-03	**0.00E+00**	6.47E-03	6.72E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f17$	**0.00E+00**	5.06E+01	3.04E+01	7.13E+01	**0.00E+00**	4.85E+00	1.79E+01	1.01E+02	3.32E-02	**0.00E+00**
	0.00E+00	4.17E+00	3.96E+00	5.00E+01	**0.00E+00**	2.75E+00	3.67E+00	8.39E+01	1.82E-01	**0.00E+00**
$f18$	**0.00E+00**	1.65E-09	1.61E-02	1.48E-04	3.23E-26	7.29E-12	3.47E-08	1.83E-10	5.02E-47	**0.00E+00**
	0.00E+00	4.57E-09	7.54E-03	7.40E-04	5.24E-26	1.83E-11	7.60E-08	5.52E-10	2.74E-46	**0.00E+00**
$f19$	**0.00E+00**	1.59E-01	5.31E-02	6.38E-01	**0.00E+00**	2.40E+00	1.10E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	4.13E-01	1.71E-01	6.82E-01	**0.00E+00**	1.44E+00	1.03E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f20$	**0.00E+00**	2.73E-01	2.33E-01	7.87E-01	5.03E-02	5.40E-01	6.53E-01	1.87E-01	9.99E-02	**0.00E+00**
	0.00E+00	4.50E-02	4.79E-02	1.20E-01	5.04E-02	1.10E-01	1.01E-01	3.36E-02	1.61E-07	**0.00E+00**
$f21$	**0.00E+00**	1.51E+01	1.33E+01	3.15E+01	7.70E-01	1.18E+01	5.76E+00	4.29E+01	9.00E-02	**0.00E+00**
	0.00E+00	1.98E+00	2.55E+00	1.97E+01	2.11E+00	1.31E+00	9.21E-01	1.81E+01	1.44E-01	**0.00E+00**
$f22$	**0.00E+00**	3.36E-11	7.11E-15	2.93E-02	3.55E-15	1.64E+00	3.67E-15	1.26E-12	**0.00E+00**	**0.00E+00**
	0.00E+00	8.02E-11	0.00E+00	1.60E-01	0.00E+00	3.02E-01	6.49E-16	5.58E-12	**0.00E+00**	**0.00E+00**
$f23$	**0.00E+00**	4.70E-03	3.16E-04	1.08E+00	**0.00E+00**	2.69E+00	6.68E-02	3.11E-10	**0.00E+00**	**0.00E+00**
	0.00E+00	1.25E-02	1.33E-03	3.82E-01	**0.00E+00**	1.06E+00	2.26E-01	1.65E-09	**0.00E+00**	**0.00E+00**
$f24$	**0.00E+00**	6.26E-02	4.31E-02	1.53E-01	1.86E+00	1.27E-02	6.26E-02	1.18E+00	1.22E+00	**0.00E+00**
	0.00E+00	6.31E-03	6.88E-03	6.56E-02	1.81E+00	6.58E-03	7.72E-03	1.33E-01	1.52E-01	**0.00E+00**
$f25$	4.97E-01	3.00E-01	**2.74E-01**	4.15E-01	7.17E-01	4.64E-01	4.29E-01	6.34E-01	9.98E-01	5.80E-01
	6.37E-02	4.38E-02	**5.82E-02**	9.54E-02	1.20E-01	8.71E-02	7.21E-02	5.71E-02	9.79E-02	9.33E-02
$f26$	4.40E-01	4.84E-01	4.74E-01	5.49E-01	4.65E-01	5.86E-01	5.25E-01	4.37E-01	5.00E-01	**4.19E-01**
	2.81E-02	1.34E-01	1.15E-01	1.92E-01	3.60E-02	2.60E-01	2.45E-01	6.17E-02	3.82E-05	**7.76E-02**
$f27$	**0.00E+00**	1.48E-17	**0.00E+00**	1.11E-15	**0.00E+00**	7.40E-17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	8.11E-17	**0.00E+00**	9.04E-16	**0.00E+00**	1.68E-16	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
$f28$	**0.00E+00**	2.21E+00	1.83E+00	1.62E+00	1.68E-16	1.66E+00	1.40E+00	9.01E+00	1.15E+01	**0.00E+00**
	0.00E+00	2.53E-01	2.76E-01	2.41E-01	9.22E-16	1.94E-01	1.57E-01	7.50E-01	2.53E+00	**0.00E+00**
$f29$	1.90E+01	7.60E+00	6.35E+00	7.67E+00	2.29E+01	5.90E+00	6.63E+00	2.65E+01	2.09E+01	**2.02E+00**
	1.11E+00	5.99E-01	5.41E-01	1.13E+00	3.36E-02	5.59E-01	1.39E+00	1.06E+00	3.02E-01	**1.98E-07**
$f30$	**0.00E+00**	3.53E+00	3.26E-05	9.12E+01	1.05E-07	1.07E-03	3.28E+01	9.43E+01	**0.00E+00**	**0.00E+00**
	0.00E+00	1.14E+00	1.58E-04	4.08E+01	5.72E-07	4.15E-03	7.36E+00	3.36E+01	**0.00E+00**	**0.00E+00**
$f31$	1.41E-13	5.24E-24	**9.42E-33**	2.08E-24	5.00E-01	8.27E-29	**9.42E-33**	9.47E-04	1.68E-01	1.87E-32
	1.08E-13	1.15E-23	**1.39E-48**	5.75E-24	1.11E-01	1.65E-28	**1.39E-48**	3.81E-04	3.42E-02	2.87E-32
$f32$	3.57E-12	4.29E-23	2.05E-31	7.47E-22	3.66E+00	1.29E-27	**1.45E-31**	6.04E-02	9.58E-01	2.53E-16
	3.28E-12	5.07E-23	7.30E-32	3.19E-21	6.65E-01	3.02E-27	**1.92E-33**	1.82E-02	1.61E-01	9.77E-16

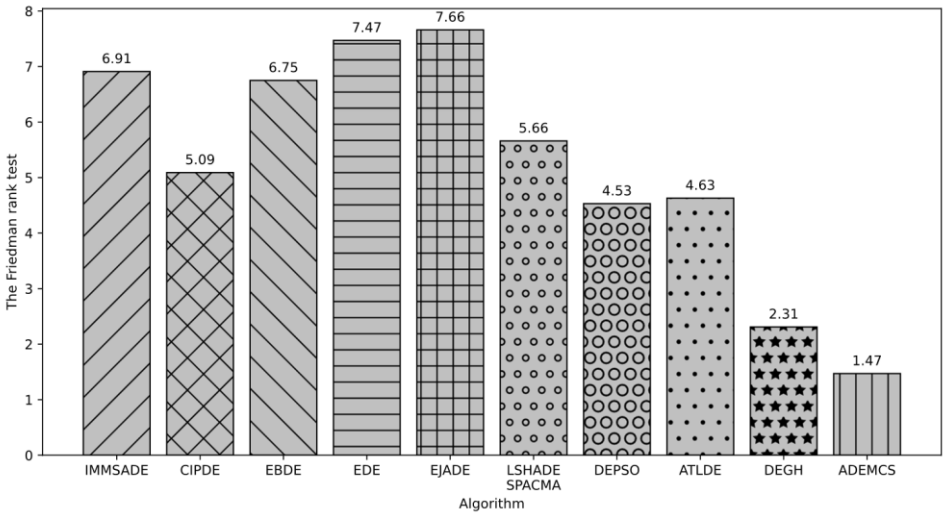

Figure 8. Friedman rank test for ADEMCS with nine state-of-the-art DE variants (D = 50)

Table 12. Wilcoxon's test for ADEMCS vs. nine state-of-the-art DE variants (D = 50)

ADEMCS vs.	p-value	α = 0.05	α = 0.1
DEGH	5.1E-02	no	yes
EDE	1.3E-05	yes	yes
CIPDE	2.9E-04	yes	yes
EJADE	1.1E-04	yes	yes
ATLDE	5.6E-06	yes	yes
EBDE	1.3E-05	yes	yes
LSAHDE-SPACMA	3.3E-05	yes	yes
IMMSADE	2.6E-06	yes	yes
DEPSO	8.3E-06	yes	yes

Table 13 presents the results with the dimension D = 100. ADEMCS achieved global or near-global optimal solutions across twenty-seven benchmark functions (*f*1-*f*12, *f*14-*f*24, *f*27-*f*30). Notably, EDE and DEGH achieved the best-known solutions for functions *f*25 and *f*26, respectively, while CIPDE attained the best results for functions *f*13, *f*31, and *f*32. Moreover, as depicted in Table 8, ADEMCS performance has good results when compared to the nine state-of-the-art DE variants. In detail, ADEMCS performed better than DEGH on 5 functions, EDE on 28 functions, CIPDE on 28 functions, EJADE on 28 functions, ATLDE on 26 functions, EBDE on 24 functions, LSAHDE-SPACMA on 28 functions, IMMSADE on 29 functions, and DEPSO on 24 functions. This comparison highlighted the good performance of ADEMCS across a diverse set of optimization tasks. Furthermore, when solving unimodal and multimodal functions, our new method, ADEMCS, continued to show good performance compared to other comparison algorithms. ADEMCS achieved better results on most unimodal functions, with the exception of being slightly less effective than CIPDE on *f*13. For multimodal functions, ADEMCS performed better than other algorithms on 13 out of 18 functions, with only worse performance observed on *f*25, *f*26, *f*31, and *f*32. Overall, for both unimodal and multimodal functions, we can conclude that our ADEMCS exhibited more robust and stable performance than other comparison algorithms.

The outcomes of the Friedman test, depicted in Figure 9, further validated ADEMCS's good performance by achieving the first rank, providing robust evidence that it performed better over the other comparison algorithms. The results from the Friedman test firmly established ADEMCS as the top-performing algorithm compared to other algorithms. Lastly, the Wilcoxon test was conducted to analyze performance differences, and the results are in Table 14. The findings confirmed that ADEMCS showed significantly different performance compared to other comparison algorithms, reaffirming its capabilities and effectiveness. However, when compared to DEGH, ADEMCS did not show a significant difference.

Table 13. Results for ADEMCS with nine state-of-the-art DE variants (D = 100)

Function Number	DEGH	EDE	CIPDE	EJADE	ATLDE	EBDE	LSHADE-SPACMA	IMM SADE	DEPSO	ADEMCS
	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD	Mean & SD
*f*1	**0.00E+00**	6.91E-08	3.36E-15	4.58E-07	2.54E-53	2.22E-06	1.85E-04	3.03E-18	1.95E-95	**0.00E+00**
	0.00E+00	7.44E-08	2.32E-15	8.46E-07	5.85E-53	3.59E-06	2.37E-04	8.83E-18	8.97E-95	**0.00E+00**
*f*2	**0.00E+00**	3.98E-03	5.43E-10	2.62E-01	4.00E-48	2.31E-02	2.01E+03	1.67E-14	1.43E-91	**0.00E+00**
	0.00E+00	6.90E-03	7.83E-10	6.27E-01	1.22E-47	3.61E-02	3.34E+03	8.56E-14	6.88E-91	**0.00E+00**
*f*3	**0.00E+00**	3.18E-01	1.28E-08	2.60E+00	1.44E-47	3.66E+01	1.04E+03	5.02E-11	3.78E-92	**0.00E+00**
	0.00E+00	6.01E-01	1.23E-08	5.91E+00	3.00E-47	1.64E+02	1.40E+03	1.93E-10	1.88E-91	**0.00E+00**
*f*4	**0.00E+00**	2.37E+02	2.36E+02	3.06E+02	5.20E-50	8.66E+02	5.77E+02	1.04E+04	1.98E-89	**0.00E+00**
	0.00E+00	9.21E+01	7.04E+01	1.33E+02	2.50E-49	1.95E+02	2.38E+02	9.86E+03	1.07E-88	**0.00E+00**
*f*5	**0.00E+00**	1.11E-03	1.48E-07	1.43E-02	5.56E-27	1.06E-03	2.02E-06	4.44E-09	1.41E-46	**0.00E+00**
	0.00E+00	1.72E-03	2.39E-07	2.40E-02	4.77E-27	2.16E-03	1.07E-05	2.17E-08	7.70E-46	**0.00E+00**
*f*6	**0.00E+00**	1.53E+01	8.53E+00	1.26E+01	1.13E-22	9.68E+00	9.38E+00	1.01E-03	1.40E-49	**0.00E+00**
	0.00E+00	1.73E+00	1.13E+00	1.90E+00	1.51E-22	1.31E+00	1.62E+00	3.16E-03	3.71E-49	**0.00E+00**
*f*7	**0.00E+00**	1.38E+42	1.92E+33	6.20E+24	4.82E-115	1.66E+52	3.33E+28	9.83E-51	4.01E-110	**0.00E+00**
	0.00E+00	6.75E+42	1.03E+34	2.11E+25	1.45E-114	9.09E+52	1.83E+29	5.34E-50	1.49E-109	**0.00E+00**
*f*8	**0.00E+00**	2.73E-08	2.03E-15	2.49E-07	7.95E-54	4.37E-07	4.14E-09	5.21E-16	5.34E-97	**0.00E+00**
	0.00E+00	2.79E-08	1.44E-15	4.55E-07	2.06E-53	5.91E-07	4.98E-09	2.45E-15	2.07E-96	**0.00E+00**
*f*9	**0.00E+00**	3.00E-07	1.39E-14	3.10E-06	2.24E-52	1.79E-06	2.22E-03	3.76E-18	3.86E-95	**0.00E+00**
	0.00E+00	4.65E-07	1.04E-14	5.36E-06	5.15E-52	2.28E-06	4.04E-03	1.55E-17	1.47E-94	**0.00E+00**
*f*10	**0.00E+00**	2.37E-03	1.27E-06	5.39E-04	3.58E-30	1.99E-02	5.09E-02	2.52E-11	6.72E-54	**0.00E+00**
	0.00E+00	1.88E-03	6.16E-07	4.33E-04	7.75E-30	1.03E-02	4.31E-02	1.16E-10	1.58E-53	**0.00E+00**
*f*11	**0.00E+00**	3.19E-12	2.66E-16	1.50E-11	0.00E+00	7.66E-11	3.59E-16	-5.00E-01	**0.00E+00**	**0.00E+00**
	0.00E+00	4.24E-12	6.90E-17	1.29E-11	0.00E+00	1.52E-10	8.59E-17	5.04E-01	**0.00E+00**	**0.00E+00**
*f*12	**0.00E+00**	1.25E-08	1.86E-13	1.79E-07	1.87E-51	8.12E-07	2.55E-12	1.55E-13	7.42E-99	**0.00E+00**
	0.00E+00	1.12E-08	1.92E-13	1.81E-07	8.03E-51	1.18E-06	4.47E-12	6.31E-13	2.33E-98	**0.00E+00**
*f*13	9.23E-04	6.33E-08	**5.54E-15**	4.89E-07	2.24E+01	8.20E-07	2.20E-04	5.01E-01	1.77E+01	6.71E-03
	6.82E-04	8.57E-08	**7.33E-15**	6.08E-07	8.51E-01	7.28E-07	2.09E-04	1.70E-01	5.63E-01	3.58E-02
*f*14	5.67E-03	1.30E-01	3.89E-02	1.34E-01	1.47E-03	3.76E-02	3.15E-02	3.99E-01	3.84E-01	**1.53E-05**
	4.66E-03	3.10E-02	1.17E-02	3.56E-02	6.19E-04	1.00E-02	1.44E-02	2.34E-01	2.38E-01	**1.18E-05**
*f*15	9.63E+01	2.45E+02	1.57E+02	1.48E+02	9.89E+01	2.60E+02	1.71E+02	9.67E+01	9.84E+01	**9.15E+01**
	2.34E-01	6.61E+01	5.33E+01	5.25E+01	4.28E-02	5.90E+01	5.07E+01	3.47E-01	2.42E-01	**2.62E+00**
*f*16	**0.00E+00**	1.54E-02	5.40E-03	5.06E-03	**0.00E+00**	6.77E-03	1.76E-02	5.88E-16	1.35E-03	**0.00E+00**
	0.00E+00	2.23E-02	1.26E-02	1.38E-02	**0.00E+00**	1.78E-02	2.08E-02	3.07E-15	7.42E-03	**0.00E+00**
*f*17	**0.00E+00**	9.62E+01	1.81E+02	1.28E+02	**0.00E+00**	2.54E+02	4.22E+01	1.12E+02	**0.00E+00**	**0.00E+00**
	0.00E+00	1.06E+01	8.87E+00	5.62E+01	**0.00E+00**	1.40E+01	5.93E+00	2.09E+02	**0.00E+00**	**0.00E+00**
*f*18	**0.00E+00**	1.31E+01	1.93E+00	1.76E+00	2.04E-26	8.52E-04	4.11E-02	1.26E-08	2.63E-48	**0.00E+00**
	0.00E+00	1.16E+01	3.09E+00	2.59E+00	3.54E-26	1.15E-03	1.98E-02	2.97E-08	1.43E-47	**0.00E+00**
*f*19	**0.00E+00**	7.30E+00	3.94E+00	5.65E+00	**0.00E+00**	7.22E+00	1.61E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	2.94E+00	2.57E+00	3.12E+00	**0.00E+00**	2.74E+00	3.92E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
*f*20	**0.00E+00**	1.44E+00	6.63E-01	2.57E+00	6.14E-02	8.93E-01	1.28E+00	2.10E-01	9.99E-02	**0.00E+00**
	0.00E+00	2.64E-01	9.64E-02	3.13E-01	4.86E-02	1.11E-01	2.36E-01	3.02E-02	9.01E-08	**0.00E+00**
*f*21	**0.00E+00**	5.29E+01	8.12E+01	6.39E+01	6.05E-03	7.97E+01	2.98E+01	3.74E+01	1.85E-02	**0.00E+00**
	0.00E+00	5.76E+00	6.33E+00	1.38E+01	6.47E-03	8.23E+00	9.94E+00	4.06E+01	2.35E-02	**0.00E+00**
*f*22	**0.00E+00**	3.46E+00	1.66E+00	1.90E+00	3.55E-15	1.61E+00	2.32E+00	2.69E-09	**0.00E+00**	**0.00E+00**
	0.00E+00	6.09E-01	3.23E-01	2.27E-01	0.00E+00	2.88E-01	3.51E-01	1.42E-08	**0.00E+00**	**0.00E+00**
*f*23	**0.00E+00**	2.28E+01	3.04E+00	1.37E+01	**0.00E+00**	2.94E+00	2.51E+00	5.09E-06	**0.00E+00**	**0.00E+00**
	0.00E+00	3.69E+00	1.13E+00	2.09E+00	**0.00E+00**	9.32E-01	1.41E+00	2.62E-05	**0.00E+00**	**0.00E+00**
*f*24	**0.00E+00**	7.63E-02	1.65E-01	5.53E-01	2.17E+00	3.03E-01	1.64E-01	2.08E+00	2.10E+00	**0.00E+00**
	0.00E+00	1.35E-02	1.86E-02	1.77E-01	2.11E+00	3.02E-02	1.90E-02	1.53E-01	1.76E-01	**0.00E+00**
*f*25	7.32E-01	5.94E-01	4.97E-01	6.18E-01	8.72E-01	**4.72E-01**	6.70E-01	8.62E-01	1.14E+00	7.31E-01
	7.18E-02	8.92E-02	8.12E-02	1.04E-01	1.24E-01	**7.18E-02**	7.55E-02	5.26E-02	1.06E-01	9.04E-02
*f*26	**4.80E-01**	5.90E-01	5.83E-01	6.00E-01	4.92E-01	5.82E-01	5.63E-01	5.19E-01	5.00E-01	5.00E-01
	1.52E-02	2.75E-01	1.76E-01	2.33E-01	7.42E-03	2.00E-01	2.46E-01	7.04E-02	6.76E-14	0.00E+00
*f*27	**0.00E+00**	1.98E-12	1.70E-16	2.77E-11	**0.00E+00**	6.72E-11	2.89E-16	**0.00E+00**	**0.00E+00**	**0.00E+00**
	0.00E+00	2.55E-12	2.72E-16	3.99E-11	**0.00E+00**	8.22E-11	3.69E-16	**0.00E+00**	**0.00E+00**	**0.00E+00**
*f*28	**0.00E+00**	7.23E+00	8.61E+00	6.88E+00	1.05E-10	9.49E+00	5.98E+00	2.77E+01	2.76E+01	**0.00E+00**
	0.00E+00	5.74E-01	5.41E-01	6.23E-01	5.46E-10	8.67E-01	5.64E-01	1.70E+00	1.06E+01	**0.00E+00**
*f*29	4.56E+01	2.68E+01	2.49E+01	2.83E+01	4.59E+01	2.85E+01	1.29E+01	6.54E+01	4.39E+01	**2.33E+00**
	3.82E-01	3.24E+00	1.32E+00	4.85E+00	3.80E-02	2.08E+00	2.23E+00	7.53E+00	4.10E-01	**3.21E-04**
*f*30	**0.00E+00**	1.77E+01	3.67E+01	2.17E+02	**0.00E+00**	7.98E+01	8.14E+01	3.38E+02	**0.00E+00**	**0.00E+00**
	0.00E+00	3.83E+00	4.74E+00	8.27E+01	**0.00E+00**	1.14E+01	1.40E+01	1.26E+02	**0.00E+00**	**0.00E+00**
*f*31	6.01E-07	1.04E-03	**1.80E-19**	1.04E-03	8.34E-01	2.68E-11	1.58E-15	4.51E-03	3.64E-01	2.07E-03
	2.62E-07	5.68E-03	**1.33E-19**	5.68E-03	1.26E-01	2.86E-11	1.47E-15	2.07E-03	5.49E-02	7.89E-03
*f*32	2.19E-03	1.25E-02	**4.51E-17**	3.33E-03	9.51E+00	5.77E-10	3.70E-04	5.73E-01	4.22E+00	1.50E-02
	5.88E-03	1.87E-02	**3.96E-17**	5.17E-03	5.48E-01	6.91E-10	2.03E-03	1.30E-01	8.46E-01	3.84E-02

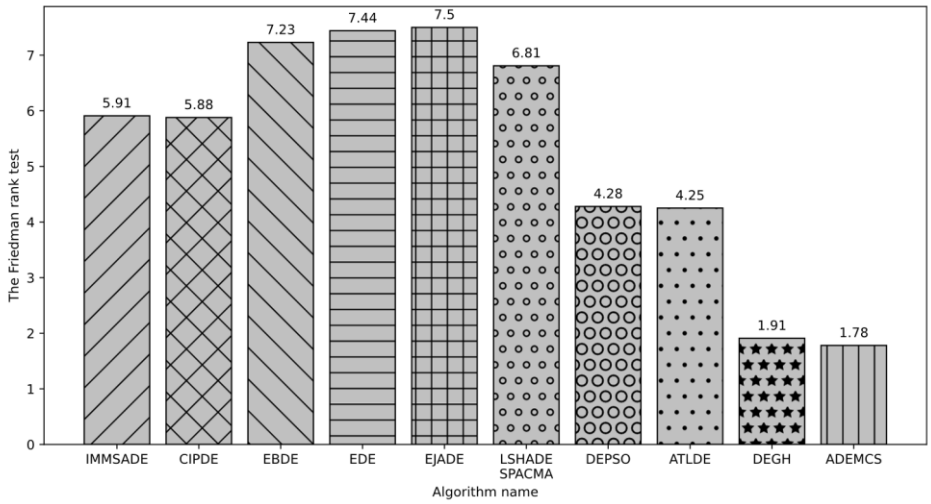

Figure 9. Friedman rank test for ADEMCS with nine state-of-the-art DE variants (D=100)

Table 14. The result of Wilcoxon's test for ADEMCS vs. nine state-of-the-art DE variants (D=100)

ADEMCS vs.	p-value	$\alpha = 0.05$	$\alpha = 0.1$
DEGH	5.2E-01	no	no
EDE	3.9E-05	yes	yes
CIPDE	7.4E-05	yes	yes
EJADE	2.6E-05	yes	yes
ATLDE	5.7E-05	yes	yes
EBDE	3.3E-05	yes	yes
LSAHDE-SPACMA	2.2E-05	yes	yes
IMMSADE	1.7E-06	yes	yes
DEPSO	1.2E-05	yes	yes

In summary, the three sets of experiments consistently showed that ADEMCS achieved the first rank for each different number of dimensions and demonstrated significant improvements compared to other algorithms. However, ADEMCS also displayed limitations as the dimension increased, resulting in decreased numbers of functions that approached global or near-global optimal solutions. Nevertheless, we can conclude that ADEMCS showed good performance in solving optimization problems with different characteristics and dimensions, performing better than the state-of-the-art algorithms in diverse scenarios.

5. Conclusion

This paper presents an Adaptive Differential Evolution algorithm with Multiple Crossover Strategy Scheme (ADEMCS) to address optimization challenges. The ADEMCS algorithm incorporates a novel adaptive mutation strategy and a hunting coordination operator from the reptile search algorithm to enhance its optimization performance. Our research aimed to thoroughly evaluate the performance of ADEMCS by comparing it with nine state-of-the-art DE variants: IMMSADE, CIPDE, EBDE, EDE, EJADE, LSHADE-SPACMA, DEPSO, ATLDE, and DEGH. We meticulously assessed the best global values achieved by each algorithm and used metrics related to addition, subtraction, and equality tests to gauge their effectiveness in handling optimization tasks. Additionally, we employed Friedman and Wilcoxon's tests to gain deeper insights into the performance differences between ADEMCS and the other DE variants.

Our experiments demonstrated the good capabilities of ADEMCS. For dimension D = 30, ADEMCS achieved global or near-global optimal solutions across twenty-nine benchmark functions and performed better than most state-of-the-art algorithms. Similarly, for dimensions D = 50 and D = 100, ADEMCS displayed good performance across multiple benchmark functions, better than the other algorithms in the majority of cases. The good performance of ADEMCS in the comparative analysis, supported by the Friedman test results, clearly established it as the top-performing algorithm for handling diverse optimization tasks. The Wilcoxon test further confirmed its significant performance compared to state-of-the-art algorithms and validated its effective approach to solving optimization problems.

For future work, the ADEMCS algorithm will be tested on more benchmark functions and real-world optimization problems to further verify its effectiveness. Furthermore, the algorithm can be extended to solve more complex optimization problems, such as those with constraints or multiple objectives. Additionally, the algorithm can be

combined with other optimization techniques to improve its performance even further. Another avenue for future work would be to investigate the scalability of the ADEMCS algorithm for large-scale optimization problems. Overall, the proposed ADEMCS algorithm has the potential to solve various optimization problems and will be a promising tool for solving complex optimization problems.

6. Declarations

6.1. Author Contributions

Conceptualization, I.F. and A.T.; methodology, I.F. and A.T.; software, I.F.; validation, I.F. and A.T.; formal analysis, I.F. and A.T.; investigation, I.F. and A.T.; resources, I.F. and A.T; data curation, I.F.; writing—original draft preparation, I.F.; writing—review and editing, A.T.; visualization, I.F.; supervision, A.T.; project administration, I.F. and A.T.; funding acquisition, A.T. All authors have read and agreed to the published version of the manuscript

6.2. Data Availability Statement

The data presented in this study are available in the article.

6.3. Funding

This work was supported by King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7. References

[1] Zheng, B., & Zhang, R. (2020). Intelligent Reflecting Surface-Enhanced OFDM: Channel Estimation and Reflection Optimization. IEEE Wireless Communications Letters, 9(4), 518–522. doi:10.1109/LWC.2019.2961357.

[2] Panwar, K., & Deep, K. (2021). Transformation operators based grey wolf optimizer for travelling salesman problem. Journal of Computational Science, 55. doi:10.1016/j.jocs.2021.101454.

[3] Hartono, N., Ramírez, F. J., & Pham, D. T. (2022). Optimisation of robotic disassembly plans using the Bees Algorithm. Robotics and Computer-Integrated Manufacturing, 78, 102411. doi:10.1016/j.rcim.2022.102411.

[4] Li, S., Gu, Q., Gong, W., & Ning, B. (2020). An enhanced adaptive differential evolution algorithm for parameter extraction of photovoltaic models. Energy Conversion and Management, 205, 112443. doi:10.1016/j.enconman.2019.112443.

[5] Biswas, P. P., Suganthan, P. N., Wu, G., & Amaratunga, G. A. J. (2019). Parameter estimation of solar cells using datasheet information with the application of an adaptive differential evolution algorithm. Renewable Energy, 132, 425–438. doi:10.1016/j.renene.2018.07.152.

[6] Tansui, D., & Thammano, A. (2020). Hybrid Nature-Inspired Optimization Algorithm: Hydrozoan and Sea Turtle Foraging Algorithms for Solving Continuous Optimization Problems. IEEE Access, 8, 65780–65800. doi:10.1109/ACCESS.2020.2984023.

[7] Ahmad, M. F., Isa, N. A. M., Lim, W. H., & Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works. Alexandria Engineering Journal, 61(5), 3831–3872. doi:10.1016/j.aej.2021.09.013.

[8] Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence, 90, 103479. doi:10.1016/j.engappai.2020.103479.

[9] Alorf, A. (2023). A survey of recently developed metaheuristics and their comparative analysis. Engineering Applications of Artificial Intelligence, 117, 105622. doi:10.1016/j.engappai.2022.105622.

[10] Hussain, K., Mohd Salleh, M. N., Cheng, S., & Shi, Y. (2019). Metaheuristic research: a comprehensive survey. Artificial Intelligence Review, 52(4), 2191–2233. doi:10.1007/s10462-017-9605-z.

[11] Ezugwu, A. E., Adeleke, O. J., Akinyelu, A. A., & Viriri, S. (2020). A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems. Neural Computing and Applications, 32(10), 6207–6251. doi:10.1007/s00521-019-04132-w.

[12] Holland, J. H. (2019). Adaptation in Natural and Artificial Systems. Adaptation in Natural and Artificial Systems. MIT press, Massachusetts, United States. doi:10.7551/mitpress/1090.001.0001.

[13] Storn, R., & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4), 341–359. doi:10.1023/A:1008202821328.

[14] Kennedy, J., & Eberhart. R.: Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942–1948. doi:10.1109/ICNN.1995.488968.

[15] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28-39. doi:10.1109/MCI.2006.329691.

[16] Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, 39(3), 459–471. doi:10.1007/s10898-007-9149-x.

[17] Yang, X.-S. S. (2010). A New Metaheuristic Bat-Inspired Algorithm BT - Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, 284, 65–74. doi:10.1007/978-3-642-12538-6_6.

[18] Yang, X. S., & Slowik, A. (2020). Firefly algorithm. In Swarm intelligence algorithms. CRC Press, 163-174. doi:10.1201/9780429422614-13.

[19] Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings, 210–214. doi:10.1109/NABIC.2009.5393690.

[20] Pham, D. T., Castellani, M., Sholedolu, M., & Ghanbarzadeh, A. (2008). The bees algorithm and mechanical design optimisation. ICINCO 2008 - Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics, ICSO, 250–255. doi:10.5220/0001506102500255.

[21] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software, 69, 46–61. doi:10.1016/j.advengsoft.2013.12.007.

[22] Wu, G., Shen, X., Li, H., Chen, H., Lin, A., & Suganthan, P. N. (2018). Ensemble of differential evolution variants. Information Sciences, 423, 172–186. doi:10.1016/j.ins.2017.09.053.

[23] Opara, K. R., & Arabas, J. (2019). Differential Evolution: A survey of theoretical analyses. Swarm and Evolutionary Computation, 44, 546–558. doi:10.1016/j.swevo.2018.06.010.

[24] Li, Y., Wang, S., Yang, H., Chen, H., & Yang, B. (2023). Enhancing differential evolution algorithm using leader-adjoint populations. Information Sciences, 622, 235–268. doi:10.1016/j.ins.2022.11.106.

[25] Li, Y., Wang, S., & Yang, B. (2020). An improved differential evolution algorithm with dual mutation strategies collaboration. Expert Systems with Applications, 153, 113451. doi:10.1016/j.eswa.2020.113451.

[26] Meng, Z., Pan, J. S., & Tseng, K. K. (2019). PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization. Knowledge-Based Systems, 168, 80–99. doi:10.1016/j.knosys.2019.01.006.

[27] Deng, W., Xu, J., Song, Y., & Zhao, H. (2021). Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. Applied Soft Computing, 100, 106724. doi:10.1016/j.asoc.2020.106724.

[28] Mohamed, A. W., & Mohamed, A. K. (2019). Adaptive guided differential evolution algorithm with novel mutation for numerical optimization. International Journal of Machine Learning and Cybernetics, 10(2), 253–277. doi:10.1007/s13042-017-0711-7.

[29] Deng, W., Shang, S., Cai, X., Zhao, H., Zhou, Y., Chen, H., & Deng, W. (2021). Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. Knowledge-Based Systems, 224, 107080. doi:10.1016/j.knosys.2021.107080.

[30] Gao, S., Yu, Y., Wang, Y., Wang, J., Cheng, J., & Zhou, M. (2021). Chaotic Local Search-Based Differential Evolution Algorithms for Optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(6), 3954–3967. doi:10.1109/TSMC.2019.2956121.

[31] Nadimi-Shahraki, M. H., Taghian, S., Mirjalili, S., & Faris, H. (2020). MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. Applied Soft Computing Journal, 97, 106761. doi:10.1016/j.asoc.2020.106761.

[32] Sun, G., Yang, B., Yang, Z., & Xu, G. (2020). An adaptive differential evolution with combined strategy for global numerical optimization. Soft Computing, 24(9), 6277–6296. doi:10.1007/s00500-019-03934-3.

[33] Zhan, Z. H., Wang, Z. J., Jin, H., & Zhang, J. (2020). Adaptive Distributed Differential Evolution. IEEE Transactions on Cybernetics, 50(11), 4633–4647. doi:10.1109/TCYB.2019.2944873.

[34] Sun, G., Li, C., & Deng, L. (2021). An adaptive regeneration framework based on search space adjustment for differential evolution. Neural Computing and Applications, 33(15), 9503–9519. doi:10.1007/s00521-021-05708-1.

[35] Yu, Y., Gao, S., Wang, Y., & Todo, Y. (2019). Global optimum-based search differential evolution. IEEE/CAA Journal of Automatica Sinica, 6(2), 379–394. doi:10.1109/JAS.2019.1911378.

[36] Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., & Zamuda, A. (2019). Distance based parameter adaptation for Success-History based Differential Evolution. Swarm and Evolutionary Computation, 50, 100462. doi:10.1016/j.swevo.2018.10.013.

[37] Wang, S., Li, Y., & Yang, H. (2019). Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. Applied Soft Computing Journal, 81, 105496. doi:10.1016/j.asoc.2019.105496.

[38] Meng, Z., & Pan, J. S. (2019). HARD-DE: Hierarchical Archive Based Mutation Strategy with Depth Information of Evolution for the Enhancement of Differential Evolution on Numerical Optimization. IEEE Access, 7, 12832–12854. doi:10.1109/ACCESS.2019.2893292.

[39] Tian, M., & Gao, X. (2019). Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization. Information Sciences, 478, 422–448. doi:10.1016/j.ins.2018.11.021.

[40] Civicioglu, P., & Besdok, E. (2019). Bernstain-search differential evolution algorithm for numerical function optimization. Expert Systems with Applications, 138, 112831. doi:10.1016/j.eswa.2019.112831.

[41] Sun, G., Xu, G., & Jiang, N. (2020). A simple differential evolution with time-varying strategy for continuous optimization. Soft Computing, 24(4), 2727–2747. doi:10.1007/s00500-019-04159-0.

[42] Abualigah, L., Elaziz, M. A., Sumari, P., Geem, Z. W., & Gandomi, A. H. (2022). Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. Expert Systems with Applications, 191, 116158. doi:10.1016/j.eswa.2021.116158.

[43] Pietropolli, G., Menara, G., & Castelli, M. (2023). A Genetic Programming Based Heuristic to Simplify Rugged Landscapes Exploration. Emerging Science Journal, 7(4), 1037-1051. doi:10.28991/ESJ-2023-07-04-01.

[44] Houssein, E. H., Mahdy, M. A., Blondin, M. J., Shebl, D., & Mohamed, W. M. (2021). Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems. Expert Systems with Applications, 174, 114689. doi:10.1016/j.eswa.2021.114689.

[45] Deng, W., Liu, H., Xu, J., Zhao, H., & Song, Y. (2020). An Improved Quantum-Inspired Differential Evolution Algorithm for Deep Belief Network. IEEE Transactions on Instrumentation and Measurement, 69(10), 7319–7327. doi:10.1109/TIM.2020.2983233.

[46] Liu, L., Zhao, D., Yu, F., Heidari, A. A., Ru, J., Chen, H., Mafarja, M., Turabieh, H., & Pan, Z. (2021). Performance optimization of differential evolution with slime mould algorithm for multilevel breast cancer image segmentation. Computers in Biology and Medicine, 138, 104910. doi:10.1016/j.compbiomed.2021.104910.

[47] Singh, D., Kumar, V., Vaishali, & Kaur, M. (2020). Classification of COVID-19 patients from chest CT images using multi-objective differential evolution–based convolutional neural networks. European Journal of Clinical Microbiology and Infectious Diseases, 39(7), 1379–1389. doi:10.1007/s10096-020-03901-z.

[48] Zhao, F., Zhao, L., Wang, L., & Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flow shop scheduling with minimizing make span criterion. Expert Systems with Applications, 160, 113678. doi:10.1016/j.eswa.2020.113678.

[49] Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation, 13(5), 945–958. doi:10.1109/TEVC.2009.2014613.

[50] Mohamed, A. W., Hadi, A. A., Fattouh, A. M., & Jambi, K. M. (2017). LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. 2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings, 145–152. doi:10.1109/CEC.2017.7969307.

[51] Chen, H., Li, S., Li, X., Zhao, Y., & Dong, J. (2023). A hybrid adaptive Differential Evolution based on Gaussian tail mutation. Engineering Applications of Artificial Intelligence, 119, 105739. doi:10.1016/j.engappai.2022.105739.

[52] Zuo, M., & Guo, C. (2022). DE/current−to−better/1: A new mutation operator to keep population diversity. Intelligent Systems with Applications, 14, 200063. doi:10.1016/j.iswa.2022.200063.

[53] Xiong, J., Peng, T., Tao, Z., Zhang, C., Song, S., & Nazir, M. S. (2023). A dual-scale deep learning model based on ELM-BiLSTM and improved reptile search algorithm for wind power prediction. Energy, 266, 126419. doi:10.1016/j.energy.2022.126419.

[54] Emam, M. M., Houssein, E. H., & Ghoniem, R. M. (2023). A modified reptile search algorithm for global optimization and image segmentation: Case study brain MRI images. Computers in Biology and Medicine, 152, 106404. doi:10.1016/j.compbiomed.2022.106404.

[55] Ekinci, S., Izci, D., Abu Zitar, R., Alsoud, A. R., & Abualigah, L. (2022). Development of Lévy flight-based reptile search algorithm with local search ability for power systems engineering design problems. Neural Computing and Applications, 34(22), 20263–20283. doi:10.1007/s00521-022-07575-w.

[56] Liang, J. J., Qu, B. Y., Suganthan, P. N., & Chen, Q. (2014). Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 29, 625-640.

[57] Awad, N. H., Ali, M. Z., & Suganthan, P. N. (2017). Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In 2017 IEEE congress on evolutionary computation (CEC), IEEE, 372-379. doi:10.1109/CEC.2017.7969336.

[58] Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 1(1), 3–18. doi:10.1016/j.swevo.2011.02.002.

[59] Wang, S., Li, Y., & Yang, H. (2017). Self-adaptive differential evolution algorithm with improved mutation mode. Applied Intelligence, 47(3), 644–658. doi:10.1007/s10489-017-0914-3.

[60] Zheng, L. M., Zhang, S. X., Tang, K. S., & Zheng, S. Y. (2017). Differential evolution powered by collective information. Information Sciences, 399, 13–29. doi:10.1016/j.ins.2017.02.055.

[61] Mohamed, A. W., Hadi, A. A., & Jambi, K. M. (2019). Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization. Swarm and Evolutionary Computation, 50, 100455. doi:10.1016/j.swevo.2018.10.006.

[62] Li, S., Gong, W., Wang, L., Yan, X., & Hu, C. (2020). A hybrid adaptive teaching–learning-based optimization and differential evolution for parameter identification of photovoltaic models. Energy Conversion and Management, 225, 113474. doi:10.1016/j.enconman.2020.113474.

[63] Zhong, X., Duan, M., Zhang, X., & Cheng, P. (2021). A hybrid differential evolution based on gaining-sharing knowledge algorithm and Harris hawks optimization. PLoS ONE, 16(4 April), 250951. doi:10.1371/journal.pone.0250951.