JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 6 No. 1 (2025) 289-307
https://publisher.uthm.edu.my/ojs/index.php/jscdm

mLPB: A Modified Learner Performance-Based Behavior Algorithm Through Differential Evolution

Abbas M. Ahmed[1,2]*, Tarik A. Rashid[3]

[1] *Darbandikhan Technical Institute, Sulaimani Polytechnic University, Sulaimani, KR, IRAQ*
[2] *Information Institute for Postgraduate Studies, Information Technology & Communications University, Baghdad, IRAQ*
[3] *Computer Science and Engineering, University of Kurdistan Hewler, Erbil, KR, IRAQ*

*Corresponding Author: abbas.ahmed@spu.edu.iq
DOI: https://doi.org/10.30880/jscdm.2025.06.01.019

Article Info

Received: 24 February 2025
Accepted: 9 June 2025
Available online: 30 June 2025

Keywords

Metaheuristics, learner performance-based behavior, differential evolution, optimization algorithms, mLPB.

Abstract

This research work presents a modified learner performance-based behavior (mLPB). It blends the learner performance-based behavior (LPB) method with the differential evolution (DE) algorithm to achieve better optimization performance. Although the LPB algorithm performs well, its limitations in the exploitation phase and slower convergence prevent it from achieving optimal results. These issues must be addressed, as they render the LPB algorithm less competitive compared to other well-established algorithms designed to tackle complex optimization problems. Inspired by the aforementioned LPB algorithm, but aiming to enhance its performance further, we replace its crossover and mutation operations with those of the DE to foster a better equilibrium between the phases of exploration and exploitation. Extensive experiments are conducted to evaluate the performance of mLPB in comparison to Genetic Algorithm (GA), Dragonfly Algorithm (DA), and Particle Swarm Optimization (PSO) on various benchmark functions, including the CEC-C06 2019 test functions. In addition, the study reviews other leading methods, including FDO, GOOSE, Leo, IFDO, FOX, CLPB, WOA, and SSA. Following this, the method works on engineering problems such as designing pressure vessels and solving the traveling salesman problem (TSP). It is quite clear from the findings that the algorithm we tested works better than the others tried in the study. Based on Wilcoxon tests, mLPB increases the processing speed, speed of convergence, and all aspects of optimization, outperforming most of the algorithms in solving large problems.

1. Introduction

Global optimization problems (GOPs) have gained attention in recent decades due to the unique advantages of meta-heuristic algorithms [1]. Nature acts as a source of inspiration for numerous metaheuristic algorithms, commonly known as nature-inspired algorithms (NIAs) [2]. GPO problems are solved using gradient-based and direct approaches, with gradient-based relying on derivatives and direct relying on objective and constraint function evaluations [3].

Originating from the concept of making the process of school leavers entering universities less difficult, a novel algorithm named LPB was presented in 2020. When matched against popular meta-heuristic methods, this one is shown to perform better. This algorithm has been applied to different optimization tasks, for example, the generalized assignment problem (GAP)[4], the (TSP) [5], the IoT service placement problem (SPP) [6], and so forth. The LPB algorithm has garnered researchers' attention due to its innovative approach, inspired by educational processes, which offers a fresh perspective on optimization. Additionally, its ability to effectively handle large optimization problems, along with superior performance in various test functions compared to established algorithms like GA and PSO, makes it a compelling choice for researchers [4]. It was also mentioned that the LPB algorithm, despite its competitive performance in optimization, faces significant challenges in the exploitation phase. Its slow convergence and inefficient exploitation, reliance on traditional crossover and mutation operators, and loss of population diversity hinder its effectiveness in large-scale and complex optimization problems. The primary challenge lies in improving exploitation capabilities, reducing convergence speed, and reducing processing time to make it more suitable for complex optimization tasks. Although LPB uses powerful ways to provide global search capability, it converges slowly and demonstrates weaknesses in the exploitation phase. Hence, using hybrid meta-heuristics enhances each technique more efficiently. Also, tests have shown that hybrid forms work faster and better than individual styles. The core contribution lies in the mLPB hybrid algorithm, which integrates DE's powerful search operators, improves exploration and exploitation balance, avoids local optima, and achieves faster convergence, demonstrating superior performance in problems of large-scale optimization. The key goals of this research work are to:

1. Create a new variant of the LPB algorithm named mLPB to improve its exploration–exploitation trade-off.
2. Enhance the speed and lower the processing time of LPB to improve its performance.
3. Reinforce the offered hybrid technique utilizing the popular algorithms such as PSO, GA, and DA on various benchmark functions by way of comparison.

In the research manuscript, after Section 1, Section 2 presents reviews and a review of previous literature. Section 3 covers the current approaches to the standard LPB and DE algorithms. Section 4 describes the suggested method, and the pseudocode is attached. Section 5 states that the suggested method was tested on the Classical Benchmark Test Functions (BTFs) and other applications. Finally, key concluding points are outlined.

2. Related Works

Metaheuristic optimization refers to a higher-level method or heuristic that is specifically developed to find a solution to an optimization issue that is sufficiently satisfying or approximate [7]. There are three fundamental approaches to developing optimization methodology: enhancing current optimizers, combining two or more existing optimizers, and creating completely novel optimizers [8]. Moreover, achieving an appropriate equilibrium remains a difficult and unresolved problem. This area of study is characterized by its considerable dynamism, resulting in the emergence of numerous novel metaheuristic optimization algorithms throughout the years.

Based on Darwinian evolution and genetic algorithms, the LPB algorithm has demonstrated efficacy in solving optimization issues via adaptive learning. It is different from others because it can adjust properly to population changes, giving a good platform for optimization. An improved method called the Improved Learner Performance-Based (ILPB) algorithm has been developed to address the issue of balancing exploration and exploitation found in the original LPB method. It brings together simulated binary crossover (SBX) and modifies the way mutation is done. The SBX algorithm improves its exploration–exploitation trade-off the resources, while the mutation process implements a dynamic and adaptable search strategy. The ILPB algorithm partitions the population into "good" and "bad" groups according to fitness values, enhancing convergence speed and efficacy in addressing large-scale optimization issues.

However, it is subject to constraints such as dependence on parameter tweaking and heightened computing complexity [9]. Updating the LPB algorithm to the Multi-Objective Learner Performance-Based Behavior (MOLPB) algorithm. The proposed modifications encompass a Pareto-based methodology, a crowding distance algorithm, and an external repository for non-dominated solutions. These adjustments guarantee that the algorithm investigates several regions of the solution space without excessively prioritizing a specific goal. The MOLPB algorithm is a multi-objective optimization technique designed for providing nondominated results in handling simultaneously many conflicting objectives. The algorithm uses crowding distance to preserve diversity, partition the population into subpopulations, and store non-dominated solutions. Furthermore, it provides exceptional diversity and convergence, resilient performance, and effective management of intricate challenges. Nevertheless, it often exhibits computational complexity and parameter sensitivity, rendering it well-suited for handling extensive populations [10].

Rather than relying on stochasticity, the new algorithm called Chaotic Learner Performance-based Behavior (CLPB) improves the quality of solutions. It creates sub-populations of the initial population, which allows the best individuals to crossover between them, which in turn helps the exploitation phase to make solutions better. There are some disadvantages to using CLPB that address the increase in complexity: it is dependent on how good the chaotic maps are used; it could affect performance; and the increased complexity could lead to overfitting and might not generalize to other problems. The increased complexity could lead to increased processing time in some cases. While CLPB is a major improvement, it causes new complications [11].

In [12], the LPB algorithm utilizes many techniques to update its crossover operations. The proposed study assesses the efficacy of several crossover operators, including LPX (Lagrangian Problem Crossover), SBX (Simulated Binary Crossover), BX (Blended Crossover), and Qubit-X, in terms of their ability to generate offspring and convergence rates. The algorithm enhances its adaptability and search approach by incorporating dynamic parameter adjustments based on population performance. Furthermore, the use of diverse random numbers for testing guarantees resilience in varied situations. Thorough iterative testing, frequently over 500 iterations, is carried out to collect complete performance data. All of these techniques make the LPB algorithm better able to find the best solutions by avoiding local optima.

In a hybrid approach, LPBSA combines LPB and Simulated Annealing (SA). By concentrating on metaheuristics, the LPBSA successfully resolves and lessens the difficulties seen in conventional LPB approaches, improving convergence, resilience, and flexibility. By applying temperature-controlled evolution, the study [13] uses simulated annealing to strengthen the LPB algorithm. This work improves the LPBSA solutions found at certain points, steering it toward better areas, and it performs better than GA, DA, PSO, Lagrange Elementary Optimization (LEO), and the LPB algorithm for handling large-scale optimization problems.

The combined optimization strategy is beneficial in many ways, although it also has limitations. Examples of these are highly parameter selection sensitivity, the need for increased computation complexity, the risk of local optima, the inability to adapt to dynamic environments, and potential implementation complexity.

Merging real-valued genetic algorithms (RGA) with DE to optimize deep groove ball bearings is a technique that performs a global search. As a result, the scope of dynamic load ratings is enhanced with a lower average assessment to the BGA. This GA-DE method is said to be better able to address the problem and more effective and efficient than other alternative methods, such as BGA, concerning continuous optimization [14].

The hybrid algorithm ABC-DE can best be described as developing from an artificial bee colony algorithm (ABC) combined with DE. The unique advantages of the hybrid algorithm, relative to existing algorithms, include its convergence of faster rates of exploration ability, which shows ABC-DE is a leading option for TMA synthesis [15]. They [16] also propose a hybridized PSO strategy to handle the problem of the non-convex economic dispatch, including a new mutation operator within DE to secure some level of DE exploitability while limiting diversity loss, which is often the case with DE.

3. Algorithms Background

The idea behind the LPB algorithm is to accept high school graduates into universities. A number of processes are involved in the admission of students, which allow them to be divided and grouped according to their cumulative rate. Having students work collaboratively improves their behavior and quality. Collaboration allows for the sharing of information amongst teammates during a study (crossover); this will randomly impact the individual personality (mutation). LPB utilizes Selection, crossover, and mutation operators from the GA [4]. Which can be expanded upon in the following sub-sections:

3.1 Incorporating GA Features into LPB

The LPB algorithm integrates elements of genetic algorithms, particularly utilizing crossover and mutation operators, to improve the optimization process by simulating behavioral impacts and interactions among learners in an educational setting. This is an analysis of how the features of genetic algorithms enhance the structure and functionality of LPB. The following is the description of the GA features, which are included in the LPB:

1. Crossover: In LPB, the crossover is utilized to replicate the interchange of learning behaviors among individuals (or "learners") within the algorithm. Similar to genetic algorithms that amalgamate segments of parental solutions to generate novel offspring, LPB employs crossover to enable "learners" to exchange effective study methodologies. This simulates real group study or collaborative learning, where students affect each other's processes, which improves a "population" of solutions.
2. Mutation: LPB also simulates mutation to produce random changes, which captures the changes in a learner's behaviors because of self-improvement or changes in study practice. This operator will assist with avoiding local optima by changing aspects of the solution randomly in the same way a student has to adjust or improve ineffective study methods to reflect university-level expectations.

By incorporating the genetic operators, LPB achieves a compromise of exploration (through random mutations) and exploitation (by selective crossover of successful individuals) and thus can converge on useful solutions while avoiding premature convergence to bad solutions [4].

3.2 LPB Procedures

The LPB algorithm [17] follows a specific sequence of steps to optimize the learner grouping and adaptation steps. At each stage, the algorithm improves efficiency through systematic improvements in the characteristics and behavior of the individuals, via a series of procedures. The LPB algorithm proceeds as follows:

1. Randomly produce an M population and assess the fitness of all of M, then evaluate the sum and average parameters to extract the maximum fitness value from M.
2. A randomly selected subset O is sorted in descending order based on fitness and divided into "Good" and "Bad" groups, with the highest fitness value chosen from each group.
3. Proceed to compare M people with O, Good maximum and Bad maximum values to partition M.
4. Individual selection.
5. Implement crossover between the chosen persons and the good individuals; then implement mutation on newly created individuals (Child).
6. Determine the fitness level of newly created persons.
7. Proceed to determine the total, mean, and maximum values between the new offspring and the forebears.
8. A new subpopulation called the "O subpopulation" is created within the new population M.
9. The steps from 2 to 7 will be repeated as necessary until the stopping condition is fulfilled or until a predefined number of iterations is reached, and then an optimal solution is returned.

The LBP algorithm's phases of exploitation and exploration are especially modified by the mLPB proposal to achieve a higher level of performance and efficiency. In the exploitation stage, LPB picks a subset of the population and breaks it up into smaller subgroups. The most suitable entities are then handpicking based on their fitness. The objective of this phase is to enhance current solutions by carefully choosing and enhancing the most exceptional individuals from the community. Different from the first phase, the generation of novel solutions through crossover and mutation on the subgroup and the whole population takes place during the exploration phase. The purpose of this phase is to explore new territory in the search space, to keep it varied and more likely to reveal solutions that are better than others [5].

The process for categorizing individuals from a population (M) into three groups: Bad, Good, and Perfect, based on their fitness values. The process begins by separating the M population into individual elements (M[i]). Should an individual's fitness match or go below that of the top score in the Bad group, they are grouped as Bad. Should a person's fitness exceed or match the highest fitness of the group, then they are considered Good. Should the fitness of the individual be higher than the Bad group as well as the Good group, then the fitness is ranked Perfect. Repeat until all individuals are grouped into one of the three groups. Applying crossover and mutation functions in the LPB algorithm will utilize multiple population groups: "Perfect," "Good," and "Bad." The algorithm must check if the "Perfect" group is empty before it applies the crossover with the "Perfect" and "Good" population, then it will apply crossover to the "Good" and "Perfect" population. Whereupon, a new population is formed, the mutation process is performed, and the new population is returned. This process ensures that individuals with higher fitness are combined for improved population generation [17]. The LPB method is constrained by challenges such as prolonged processing time, sluggish convergence rate, possible local optima, and the need for enhancement in the exploitation phase. Unlike PSO and GA, which have strong feedback mechanisms and population diversity control to balance exploration and exploitation, LPB relies on basic selection and crossover without adaptive behavior. So LPB converges slowly and may get trapped in local optima; hence, it needs to be structurally enhanced to be competitive with other advanced metaheuristics [11].

3.3 Classical DE

DE works like the natural process of evolution to steer its population to the best solution available by utilizing the dissimilarity among individuals to direct its search within the solution space. It mainly consists of four major strategies: population initialization, mutation, crossover, and selection [18]. The DE focuses on recognizing and scaling two unique vectors found in the same population. The population gets a mutation individual vector when adding a different third individual vector. The entity mutant and the parent vectors cross, with the chance that an attempt to create an individual vector will succeed. In the last phase, the parent vector that has the highest fitness value is compared to the best trial vector using greedy selection. The fitter individual, i.e., the one which has that has the best performance, is thereafter transmitted to the next offspring. Details about how the DE evolves are given below [19].

3.3.1 Initialization Strategy

Before optimization starts, the first step is to create an initial group of candidates. Eq.(1) states that each decision entity in the starting population is assigned a value within the boundaries. [20]

$$x_{ij}^{0} = a_j + rand_j.\left(b_j + a_j\right) \tag{1}$$

The interval [0,1] represents a uniformly distributed number, while boundaries aj and bj represent the minimum and maximum values for the jth decision parameter.

3.3.2 Mutation Strategy

During a mutation procedure, a mutant vector, also known as a donor vector, is generated. The donation vector v_i^{G+1} of the jth member of the population is formed by numerically combining a set of vectors, the third one being weighted differently, Eq.(2)[20].

$$v_i^{G+1} = x_{r1}^G + F * (x_{r2}^G - x_{r3}^G),\ r_1 \neq r_2 \neq r_3 \tag{2}$$

Given random indices $r_1, r_1, r_1 \in \{1,2,3,\ldots,NP\}$, and the mutation probability parameter is denoted as F.

3.3.3 Crossover Strategy

Utilizing the following method, the base vector is combined with the modified vector to produce the trial vector u_i^{G+1}. The trial vector is shown in Eq. (3).

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, rand(j) \ \leq CR \ or \ j = rand\ n(i), \\ x_{ij}^{G+1} rand(j) \ \leq CR \ and \ j \neq rand\ n(i), \end{cases} \tag{3}$$

j represents the jth outcome from a uniform random generator number, denoted as $\boldsymbol{rand(j)} \in [\mathbf{0}, \mathbf{1}]$. The crossover probability constant, denoted as $\boldsymbol{CR} \in [\mathbf{0}, \mathbf{1}]$, must be chosen by the user. The random selection of the index $\boldsymbol{rand\ n(i)} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots \boldsymbol{D}\}$ guarantees that $\boldsymbol{u_{ij}^{G+1}}$ receives at least one element from $\boldsymbol{v_{ij}^{G+1}}$. All new parent vectors would therefore be unnecessary, so the population would stay the same.

3.3.4 Selection Strategy

In a population, all solutions are given identical chances to be part of choosing parents, regardless of their fitness value. Now the new child x_i^{G+1} generated after mutation & crossover is evaluated. Then the best comparison between the child vector and its parent x_i^G is selected. The best parent is kept in the population if it still comes out better. Eq. (4)[20].

$$x_i^{G+1} \begin{cases} u_i^{G+1}, & f(u_i^{G+1}) < f(x_i^G), \\ x_i^G, & f(u_i^{G+1}) \geq f(x_i^G). \end{cases} \tag{4}$$

Individuals' fitness is increased through many generations as they find various optimum solutions in trying to get the best outcomes.

4. Proposed mLPB Algorithm

In the field of metaheuristics, hybridization mainly involves the integration of the most advantageous characteristics of a few algorithms for producing a novel algorithm, which can be anticipated to surpass its predecessors in solving application-specific or generic benchmark issues [21]. When the balance between how much the algorithm explores and exploits is too weak, the algorithm converges less and still fails to be accurate. It may be hard to prove that the exploration of the standard LPB algorithms is better than the exploitation ability. The analysis in the paper is [17] indicates that LPB's exploitation capability must be strengthened. Overall, LPB performs well, but there is room for improvement in the exploitation phase, which oversees fine-tuning solutions once they are close to optimal. The authors specifically mentioned that to maximize LPB's performance, future work will concentrate on enhancing its exploitation phase. As a result, the majority of current research focuses on enhancing exploitation capabilities.

Many researchers used various techniques, as we mentioned in section 2, to enhance LPB's performance. Recapitulating literature, even though each study has its advantages and disadvantages, and their levels of simplicity, accuracy, and convergence performance differ. Therefore, current studies are aimed at making LPB better at both exploring and exploiting skills. Four primary components make up the fundamental concepts of DE. These components include initializing the population, implementing mutation and crossover techniques, and performing selection. The overarching goal of these elements is to optimize one or more objectives. There is a population of individual learners that are randomly initialized by the mLPB algorithm Fig.1. Each of these learners has its own set of learning parameters. To determine the potential for behavioral change, the frequency, also known as the "success rate," of performing mLPB stages is essential. It is possible to change the original LPB to create a hybrid system by utilizing DE. As a result, a new version called mLPB is formed, which incorporates both learner-based learning and DE processes.

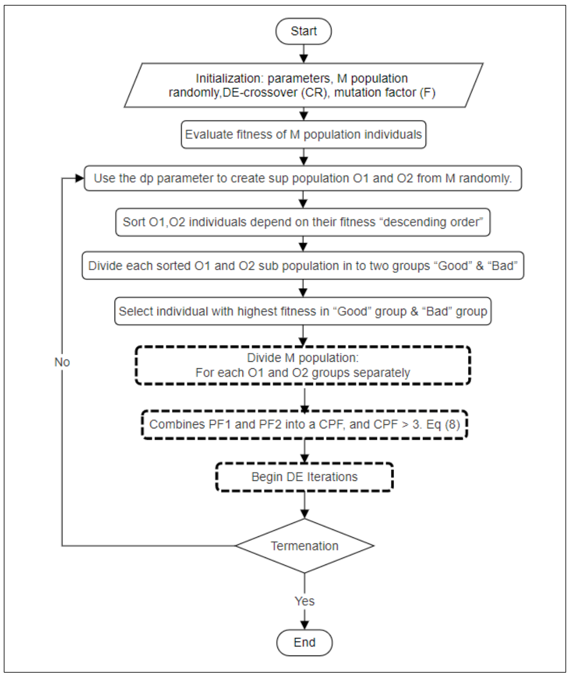
Start
Initialization: parameters, M population randomly,DE-crossover (CR), mutation factor (F)
Evaluate fitness of M population individuals
Use the dp parameter to create sup population O1 and O2 from M randomly.
Sort O1,O2 individuals depend on their fitness "descending order"
Divide each sorted O1 and O2 sub population in to two groups "Good" & "Bad"
Select individual with highest fitness in "Good" group & "Bad" group
Divide M population: For each O1 and O2 groups separately
Combines PF1 and PF2 into a CPF, and CPF > 3. Eq (8)
Begin DE Iterations
Termenation
No
Yes
End

Fig. 1 *Flowchart of the mLPB algorithm*

Using the approach above, we can randomly generate subpopulations O1 and O2 from a population M via the dp parameter. Assess the level of fitness that individuals from both O1 and O2 have Fig.2. Both subpopulations O1 and O2, are categorized into two groups, the good (highest fitness) and the bad (lowest fitness), to assess the level of fitness that each individual in the population M has, then each group in isolation, to assess which of the subpopulations, good1, bad1 for O1, and good2, bad2 for O2 populations has the highest level of fitness. The goal of this division is to reach the required values (the best individuals to use in the next phase) from the general populations in the shortest amount of time possible, as these two groups, O1, O2, will work in parallel, instead of having one group in the original algorithm. This means that I will have four groups through which the M populations will be evaluated based on the fitness in these four groups, according to the following check strategy (for each O group separately):

Concerning the subpopulations (O1), move an individual from the M population to the BP1 population of the bad1 population group, Eq. (5), if the individual's fitness is not higher than the highest fitness in the bad population. If a person from M has a fitness level that is not greater than the highest fitness in the good population, then move that person to the good1 population group called GP1 Eq. (6). Alternatively, it should be transferred to the ideal population, which is perfect PF1 Eq. (7). The above is a mathematical representation:

1. If the individual's fitness value from population M does not equal or surpass the best fitness score among the bad population "bad":

$$if\ f(M_i) \leq \max\big(f(bad)\big) \Rightarrow M_i \rightarrow BP1 \tag{5}$$

2. If the individual's fitness value from population M does not equal or surpass the best fitness score among the good population "good":

$$If\ f(M_i) \leq \max\big(f(good)\big) \Rightarrow M_i \rightarrow GP1 \tag{6}$$

3. If neither of the above conditions is met, the individual is transferred to the perfect population PF1:

$$Otherwise\ M_i \rightarrow PF1 \tag{7}$$

Where $f(M_i)$ represents the fitness value of the individual M_i , max $\big(f(bad)\big)$ and max $\big(f(good)\big)$ represent the highest fitness value in the bad group and the good group, respectively.

This process is also done similarly for the O2 subgroup. As a final result of this stage of the divisions of the random population M, they were divided into six population groups, respectively, bad1, good 1, perfect 1, bad2, good 2, and perfect 2. The next step is to combine the first perfect group, PF1, with the second perfect group, PF2, into one group named the combined perfect CPF. Ascending order is used to organize the group by the values and variables inside it. Then the group CPF is reduced to the size of the PF1 group before merging with the highest fitness selection from the integrated group, Eq. (9), provided that the number of individuals in CPF is more than three, to prepare it for the next phase of the modified algorithm. The LPB algorithm's exploratory capability was augmented at this stage by utilizing this method and the novel approach to division. The mathematical explanation of the above is as follows:

1. Dividing the M population into six subgroups based on fitness:
   $M \rightarrow \{bad1, good1, perfect1, bad2\ good2, perfect2\}$
2. The two "perfect "groups, PF1 and PF2, are combined into a single population group, CPF:
   $$CPF = PF1 \cup PF2$$

The method of selecting the population in the CPF group is done through the following formula:

$$k \le n$$
$$\Rightarrow \begin{cases} if\ CPF\ is\ not\ empty\ and\ individuals \ge 3 & select\ an\ individual\ from\ CPF \\ if\ CPF\ is\ not\ empty\ and\ individuals < 3, \begin{cases} if\ GP1\ not\ empty, & select\ remaining\ individual\ from\ GP1 \\ Else\quad if\ GP1\ empty, & select\ remaining\ individual\ from\ GP2 \end{cases} \end{cases} \tag{8}$$

Where:
k iterations' number, where n indicates the total number of permitted iterations.

3. The aggregated group CPF is sorted in increasing order according to fitness values.
4. The size of the CPF population is reduced to match the size of the PF1 population by selecting the individuals with the highest fitness:
   $$CPF_{reduced} = Select\ top\ individual\ from\ CPF\ such\ that\ |CPF_{reduced}| = |PF1|$$
   Provided that:
   $$|CPF| > 3 \tag{9}$$

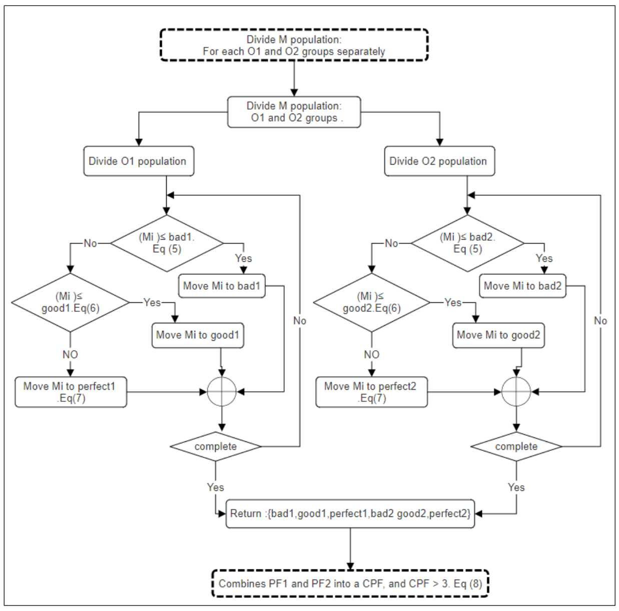
Divide M population: For each O1 and O2 groups separately
Divide M population: O1 and O2 groups .
Divide O1 population
Divide O2 population
(Mi)≤ bad1. Eq (5)
(Mi)≤ bad2. Eq (5)
No
Yes
Move Mi to bad1
Move Mi to bad2
(Mi)≤ good1.Eq(6)
(Mi)≤ good2.Eq(6)
Move Mi to good1
Move Mi to good2
NO
Move Mi to perfect1 .Eq(7)
Move Mi to perfect2 .Eq(7)
complete
Return :{bad1,good1,perfect1,bad2 good2,perfect2}
Combines PF1 and PF2 into a CPF, and CPF > 3. Eq (8)

Fig. 2 *Flowchart of dividing M population in mLPB*

Through the use of a crossover mechanism, an mLPB algorithm allows students to trade studying habits, thereby simulating interactions that occur in real life. This process results in the development of new, more effective behaviors. Mutation is associated with the introduction of random changes to the behavior of learners, which promotes diversity and has the potential to improve learning results. To enhance the exploitation performance of LPB, in this work, a new optimization algorithm, mLPB, is introduced by adding two DE operators to the original LPB. To strengthen the LPB algorithm's accuracy and convergence behavior, mLPB adapts the approach of the DE mutation and crossover strategies to the group collaboration (crossover stage) and metacognition (mutation stage) of LPB.

The DE uses a random population, and to use the properties and advantages of this algorithm, we use the group and the population in the CPF group as the initial population for the DE operations injected with this mLPB algorithm phase. As we have noted in Fig.2, the CPF group is considered the elite of the divided population, meaning that the best fitness in each iteration division can be seen in this group. Therefore, when this group or individuals of this group enter the DE operations as an initial population to improve it and pass through the mutation and crossover steps of this algorithm until reaching the selection part, the improved population will be improved and thus the best results will be reached in the fewest possible iterations of the proposed algorithm.

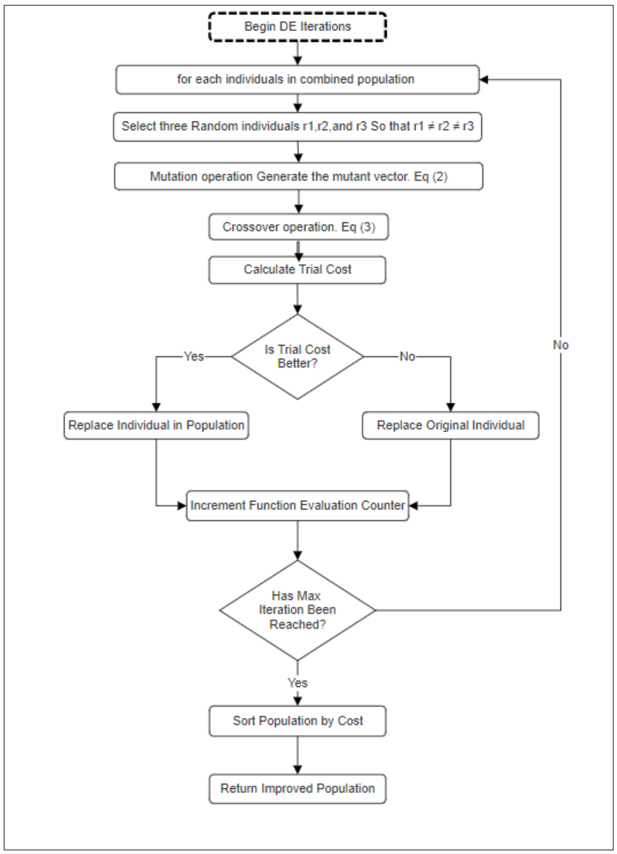
Begin DE Iterations
for each individuals in combined population
Select three Random individuals r1,r2,and r3 So that r1 ≠ r2 ≠ r3
Mutation operation Generate the mutant vector. Eq (2)
Crossover operation. Eq (3)
Calculate Trial Cost
Is Trial Cost Better?
Yes
No
Replace Individual in Population
Replace Original Individual
Increment Function Evaluation Counter
Has Max Iteration Been Reached?
No
Yes
Sort Population by Cost
Return Improved Population

Fig. 3 *Flowchart of DE operations in mLPB*

In Fig.3, we can see that mLPB's search strategy for all elements in a CPF population at any given time starts with a population of random solutions, changes them by mutation and crossover, picks the best solution accordance to the optimization function, and keeps doing this until the best solution is found. The mutation operator includes modifying its target vector by adopting another vector from the population to create what is referred to as the donor vector. The mutation strategy can be expressed in Eq. (2). To preserve the diversity that exists within the population, the crossover rate (CR) is utilized. To carry out this procedure, a trial vector is created by mixing the values of a donor and target individuals. A comparison is made between a randomly generated number and a certain probability of picking each value that will be included in the trial vector. If the stochastic value is less than the likelihood or if a particular location is selected randomly, the value is taken from the individual who participated in the donation. If that is not the case, it is taken from the person who is the target, Eq. (3). This random selection assures that the trial vector is distinct from the individual being targeted in certain ways, which facilitates the development of a diverse range of solutions. The selection's outcome determines whether to use the target vector or the trial vector of the current population to produce a new vector for the next generation. It compares the new solution and the obtained solution in the last iteration, replacing it if it is better. Alternatively, holding it as the perfect solution for the next generation and then improving the solution iteratively to find the best solution. The convergence rate of the mLPB exhibits a remarkably faster speed when compared with that of the LPB algorithms. However, the fast convergence property does give rise to a more considerable chance of searching close to a local optimum that entails good exploitation and thus avoids premature convergence.

5. Results and Discussion

To analyze the mLPB algorithm, this section makes use of several typical benchmark functions that may be found in the literature. Four well-known algorithms are compared with the mLPB algorithms' results in 19 classical benchmark tests, such as DA, PSO, and GA In addition to the original LPB algorithm, however, we studied the CEC-C06 2019 test functions (CEC'19 functions) and BTFs to demonstrate the ability of the algorithm to solve scalable optimization problems. Then, the Wilcoxon rank-sum test is employed for statistical comparison of these results against each other to determine their significance.

5.1 BTFs

Evaluation and comparison of the performance of improved algorithms can be accomplished through the utilization of test functions, which are divided into three main groups in locating the optimal solution within a specified search space. Unimodal functions are characterized by having only one local minimum or maximum, which makes them rather simple to compute. This simplistic approach is explicitly intended to assess the ability of an algorithm or an enhanced algorithm to converge on a single global optimum. An example of this category is the quadratic function. Multi-modal functions have many local minima and maxima, which makes it hard for algorithms to find all the optima. An illustration of this category is the Rastrigin function. Composite functions consist of both unimodal and multimodal components, which are created by concatenating simpler functions, usually via summing or multiplication. The reason for designing these functions this way is to reflect real-world problems that have both basic and advanced features, which are made up of quadratic and sinusoidal functions; for additional details regarding the test functions and their parameters, please consult [22],[23]. an extensive range of BTFs is used to assess the efficacy of the mLPB. These functions have been meticulously selected to demonstrate different facets of optimization problems. Composite functions consist of both unimodal and multimodal components, which are created by concatenating simpler functions, usually via summing or multiplication. These functions are intentionally built to model realistic problems that could demonstrate simple and sophisticated characteristics. The performance of the mLPB is evaluated using a wide variety of BTFs. Those functions are carefully selected to show the diversity of properties of optimization problems.

The evaluation suite consists of unimodal functions that measure the level of convergence and exploitation by the algorithm, as there is only one optimal solution. Another important part of the test set is multimodal functions, having several optima, such as global and local ones. This group meticulously evaluates the algorithm's capacity to explore, guaranteeing that it does not identify local optimal solutions. Furthermore, compound test functions comprising elements of both unimodal and multimodal cases provide a comprehensive evaluation. We evaluated mLPB using a number of functions to ensure that we get a well-rounded understanding of how it performs in the different optimization problems. The proposed mLPB was evaluated using standard algorithms like DA, PSO, GA, and LPB across 19 benchmark functions. The test functions of each algorithm in Table 1 were executed thirty times using 500 iterations, and then three results ("standard deviation (Std Dev)," "averages (Ave.)," and "runtime RT(s)") were obtained, with parameter settings adapted from the literature [24], and [4] In the end, both the "Ave." and "Std Dev." were calculated to check how stable and effective the algorithm was. mLPB performed better than other methods in most unimodal functions, except for F3, where DA did slightly better because of its strong exploitation. In these multi-function settings (F8–F13), mLPB always performed better than all other algorithms, including the original LPB. Generally, mLPB was able to explore well and avoided getting stuck in local optima for all test functions (F1–F19). The way it converges is illustrated in Fig. 4. One function is chosen from each category of test functions (the same functions used in the original algorithm, ref [4] to ensure accurate comparison. The modified algorithm consistently achieves less execution time than the LPB. Within certain functions, the disparity in implementation is more extreme. It also handles the balance between searching the whole space and improving a solution better than the tested optimization algorithms.

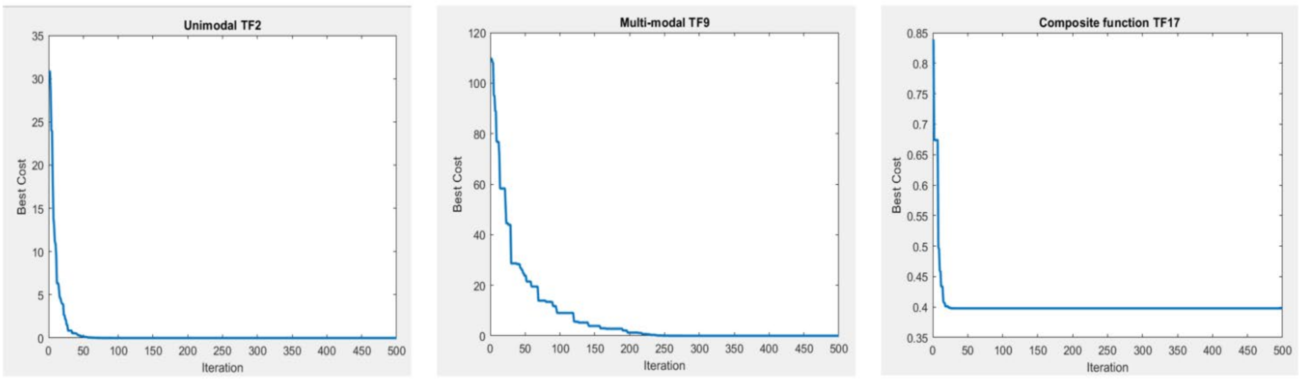

Fig. 4 *Convergence profile of mLPB for unimodal, multimodal, and composite* BTFs.

Table 1 *Results of 19* BTFs to *compare mLPB with LPB, DA, PSO, and GA algorithms. Significant values are in value [shaded]*

F.no	Indicator	mLPB	LPB	DA	PSO	GA
F1	Ave.	7.08984E-34	0.001877545	2.85E-18	4.20E-18	748.5972
	Std Dev	6.0466E-34	0.002093616	7.16E-18	4.31E-18	324.9262
	RT (s)	106.4292	160.840946	1445.243327	249.665030	65.422226
F2	Ave.	3.5632E-21	0.005238111	1.49E-05	0.003154	5.971358
	Std Dev	1.52758E-21	0.003652512	3.76E-05	0.009811	1.533102
	RT (s)	108.6203	169.076368	1259.496468	3.826913	55.040008
F3	Ave.	2.29808E-06	36.4748883	1.29E-06	0.001891	1949.003
	Std Dev	1.79226E-06	29.22415523	2.10E-06	0.003311	994.2733
	RT (s)	111.87	202.408611	1216.762524	12.702411	80.126424
F4	Ave.	5.98399E-09	0.393866	0.000988	0.001748	21.16304
	Std Dev	3.55694E-09	0.135818	0.002776	0.002515	2.605406
	RT (s)	110.5087	191.301934	1399.014810	2.877756	63.099468
F5	Ave.	0.775528967	16.76919	7.600558	63.45331	133307.1
	Std Dev	0.721185745	22.19251	6.786473	80.12726	85007.62
	RT (s)	108.3106	130.846636	1707.285731	5.224432	55.818782
F6	Ave.	8.05439E-22	0.00203173	4.17E-16	4.36E-17	563.8889
	Std Dev	5.56752E-22	0.0027832	1.32E-15	1.38E-16	229.6997
	RT (s)	107.0503	157.547318	1550.130722	2.795879	51.284046
F7	Ave.	0.002066636	0.004975	0.010293	0.005973	0.166872
	Std Dev	0.000780291	0.002965	0.010293	0.003583	0.072571
	RT (s)	110.9875	158.642028	1593.877054	8.982616	56.555067
F8	Ave.	- 4189.8289	-3747.65	-2857.58	-7.10E+11	-3407.25
	Std Dev	3.63798E-12	189.0206	383.6466	1.2E+12	164.478
	RT (s)	111.8153	162.354305	1738.794894	8.266467	55.234252
F9	Ave.	1.15582E-12	0.001567	16.01883	10.44724	25.51886
	Std Dev	1.44882E-12	0.001842	9.479113	7.879807	6.66936
	RT (s)	108.9512	159.074029	1638.957037	4.816792	84.833759
F10	Ave.	2.36887E-11	0.017933	0.23103	0.280137	9.498785
	Std Dev	6.26602E-11	0.013532	0.487053	0.601817	1.271393
	RT (s)	112.0596	128.431567	1297.325669	8.013542	84.666823
F11	Ave.	4.43E-09	0.066355	0.193354	0.083463	7.719959
	Std Dev	9.80E-09	0.030973	0.073495	0.035067	3.62607
	RT (s)	1.12E+02	130.664299	1210.086084	9.429028	56.656545
F12	Ave.	2.84914E-11	2.78659E-05	0.031101	8.57E-11	1858.502
	Std Dev	1.55661E-11	3.83626E-05	0.098349	2.71E-10	5820.215
	RT (s)	123.9046	140.837076	1464.060419	22.898798	102.745164
F13	Ave.	0.000002	0.000309	0.002197	0.002197	68047.23
	Std Dev	1.403E-06	0.000512	0.004633	0.004633	87736.76
	RT (s)	127.3184	139.449467	1339.438272	16.752814	103.377836
F14	Ave.	0.998	0.998004	103.742	150	130.0991
	Std Dev	0	1.26E-11	91.24364	135.4006	21.32037
	RT (s)	163.8527	170.207352	1034.450489	86.298548	152.142368
F15	Ave.	0.000659933	0.002358	193.0171	188.1951	116.0554
	Std Dev	0.000214755	0.003757	80.6332	157.2834	19.19351
	RT (s)	110.5788	247.224271	1659.652400	8.250347	54.974533
F16	Ave.	-1.03160	-1.03163	458.2962	263.0948	383.9184
	Std Dev	6.66134E-16	2.46E-06	165.3724	187.1352	36.60532
	RT (s)	107.4854	181.858429	969.827007	4.247415	80.998874
F17	Ave.	0.39789	0.397888	596.6629	466.5429	503.0485
	Std Dev	1.66533E-16	3.16E-06	171.0631	180.9493	35.79406
	RT (s)	105.8968	141.213291	1018.757437	2.607163	50.990811
F18	Ave.	3	3.000142	229.9515	136.1759	118.438
	Std Dev	0	0.000283	184.6095	160.0187	51.00183
	RT (s)	106.6368	180.663489	1001.716543	2.718852	80.273981
F19	Ave.	-3.8628	-3.86278	679.588	741.6341	544.1018
	Std Dev	3.10862E-15	9.61E-07	199.4014	206.7296	13.30161
	RT (s)	120.2877	169.415055	1312.805448	8.952319	77.905123

5.2 CEC'19 Functions Results

The CEC'19 functions are applied to assess how well the mLPB algorithm performs on large-dimensional optimization problems, which matters greatly for the success of the annual competition [4]. Comparative analysis was conducted between the outcomes accomplished through the proposed algorithm, mLPB, and several established algorithms, including the LPB, dragonfly algorithm (DA), and PSO algorithm. mLPB and other algorithms under comparison have been repeatedly executed 30 times using around 80 search agents, and a 500-iteration selection of the mLPB parameters has been made to optimize the outcomes and operate the suggested algorithm with an equivalent number of objective function evaluations as comparative algorithms. The results of the mLPB and other algorithms used in this study are examined in Table 2. From the outcomes, the averages for the CEC01 to CEC5 in CEC'19 functions, and the (Std Dev) of mLPB, show the lowest result of all the other algorithms used for comparison in this paper. Furthermore, in the CEC07 to CEC09 test functions, mLPB had better results than LPB, DA, and PSO. It was lower in CEC06 and CEC010. The performance of the relative method achieved its best performance in seven standard functions, and proved the effectiveness of the mLPB algorithm. While it generally outperforms LPB, the difference under certain conditions is minimal, indicating that mLPB still has potential for further improvement. The outcomes of four other modern optimization algorithms, FOX, SSA, CLPB, and the whale optimization algorithm (WOA), were extracted from [15],[13], and [27] and are compared with those of the mLPB algorithm. The typical parameter, comprising 80 agents and 500 repetitions, is identical to those previously used. After 30 runs of the procedure, the Ave. and Std Dev are calculated for each test function (Table 4). For mLPB, they are lower than those for the modern algorithms of FOX, CLPB, WOA, and SSA in the vast majority of cases of CEC'19 functions. However, FOX showed its superiority in CEC01 and CLPB in CEC06. Furthermore, the Outcomes of the CEC'19 functions show that mLPB outperforms all algorithms in large-scale optimization problems.

Table 2 *Results on the CEC'19 functions test*

CEC Function	Indicator	mLPB	LPB	DA	PSO
CEC01	Ave.	30529827.432033	7494381363.657680	543 x108	1.47127E+12
	Std Dev	17899902.254976	8138223463.280230	669 x 108	1.32362E+12
	RT (s)	401.8522	377.373846	2034.958870	382.330436
CEC02	Ave.	17.3429	17.63898	78.0368	15183.91348
	Std Dev	0.00	0.31898	87.7888	3729.553229
	RT (s)	108.1111	140.912536	2122.108475	6.064791
CEC03	Ave.	12.7024	12.7024	13.7026	12.70240422
	Std Dev	0	0	0.0007	9.03E-15
	RT (s)	108.8118	144.194876	2223.799974	8.901970
CEC04	Ave.	7.02396	77.90824	344.3561	16.80077558
	Std Dev	4.14440	29.88519	414.0982	8.199076134
	RT (s)	113.038	137.305797	1720.974833	5.179151
CEC05	Ave.	1.04238	1.18822	2.5572	1.138264955
	Std Dev	0.03688	0.10945	0.3245	0.089389848
	RT (s)	117.4808	138.406681	1722.243949	5.370252
CEC06	Ave.	7.309273333	3.73895	9.8955	9.305312443
	Std Dev	0.41501	0.82305	1.6404	1.69E+00
	RT (s)	215.5229	142.041586	1401.682147	131.167162
CEC07	Ave.	8.543137	145.28775	578.9531	160.6863065
	Std Dev	111.772998	177.8949	329.3983	104.2035197
	RT (s)	113.58090	122.135692	1376.289834	5.436392
CEC08	Ave.	3.36130	4.88769	6.8734	5.224137165
	Std Dev	0.62003	0.67942	0.5015	0.786760649
	RT (s)	117.3091	138.20745	1802.883649	5.527832
CEC09	Ave.	2.35111	2.89429	6.0467	2.373279266
	Std Dev	0.00461	0.23138	2.871	0.018437068
	RT (s)	111.7715	141.699472	1365.799778	4.446880
CEC10	Ave.	20.05759	20.00179	21.2604	20.28063455
	Std Dev	0.02469	0.00233	0.1715	0.128530895
	RT (s)	115.3385	147.995515	1699.088096	9.462923

For mLPB to be compared against the 19 selected standard BTFs, the contemporary FDO, GOOSE, Leo, and IFDO algorithms will be designated as reference algorithms in the second group. The test outcomes of the FDO and GOOSE algorithms applied to the 19 selected standard BTFs will also be referred to in the references [25], [28], [29]. The test results have been presented in Table 3.

Table 3 *mLPB, FDO, GOOSE, Leo, and IFDO test results on the standard BTFs*

F.no	Indicator	mLPB	FDO	GOOSE	Leo	IFDO
F1	Ave.	7.08984E-34	7.47E-22	1.15E-05	2.69874E-09	5.38E-24
	Std Dev	6.0466E-34	7.26E-19	1.84E-05	7.49992E-09	2.74E-23
F2	Ave.	3.5632E-21	9.388E-07	1.16E-02	3.7305E-06	0.534345844
	Std Dev	1.52758E-21	6.91E-06	7.93E-03	3.95635E-06	1.620259633
F3	Ave.	2.29808E-06	8.552E-08	0.0011	5.31468E-09	2.88E-07
	Std Dev	1.79226E-06	4.40E-06	1.50E-03	2.07901E-08	6.90E-07
F4	Ave.	5.98399E-09	6.688E-05	1.00E-03	3.60286E-05	2.60E-04
	Std Dev	3.55694E-09	2.49E-03	8.19E-04	3.22842E-05	9.11E-04
F5	Ave.	0.775528967	23.501	2.88E+01	10.60296667	1.94E+01
	Std Dev	0.721185745	5.98E+01	2.19E-02	13.93285916	3.31E+01
F6	Ave.	8.05439E-22	1.422E-19	0.0099	4.31581E-10	4.22E+06
	Std Dev	5.56752E-22	4.75E-18	3.32E-03	5.51803E-10	8.15E-09
F7	Ave.	0.002066636	0.544401	5.70E-03	0.001449721	5.68E+01
	Std Dev	0.000780291	3.15E-01	3.82E-03	0.002690575	3.14E+01
F8	Ave.	- 4189.8289	-2285207	-7187.6	- 2989.147333	-2.92E+06
	Std Dev	3.63798E-12	2.07E+05	6.59E+02	202.684514	2.24E+05
F9	Ave.	1.15582E-12	14.56544	0.0038	37.07867	1.35E+01
	Std Dev	1.44882E-12	5.20E+00	5.31E-03	12.2775166	6.66E+00
F10	Ave.	2.36887E-11	3.996E-16	0.002	4.8836E-05	5.18E-15
	Std Dev	6.26602E-11	6.38E-16	2.07E-03	2.89869E-05	1.67E-15
F11	Ave.	4.43E-09	0.568776	6.67E-07	2.7393E-08	0.525690405
	Std Dev	9.80E-09	1.04E-01	9.68E-07	5.51514E-08	8.90E-02
F12	Ave.	2.84914E-11	19.83835	0.00026	1.87667E-08	1.81E+01
	Std Dev	1.55661E-11	2.64E+01	1.18E-04	2.89749E-08	2.57E+01
F13	Ave.	0.000002	10.2783	0.0079	8.90491E-09	4.10E+09
	Std Dev	1.403E-06	7.42E+00	6.85E-03	1.88063E-08	1.50E-05
F14	Ave.	0.998	3.787E-08	9.9012	6.9979	2.68E-07
	Std Dev	0	6.32E-07	3.90E+00	5.833242622	4.68E-07
F15	Ave.	0.000659933	0.0015202	0.000315	0.001673093	4.03E-16
	Std Dev	0.000214755	1.24E-03	1.38E-05	0.003539145	9.25E-16
F16	Ave.	-1.03160	0.006375	-1.0316	- 0.622100333	9.14E-16
	Std Dev	6.66134E-16	1.06E-02	6.66E-16	0.396782974	3.61E-16
F17	Ave.	0.39789	23.82013	0.3979	1.788405333	2.38E+01
	Std Dev	1.66533E-16	2.15E-01	1.67E-16	2.237631581	1.24E-01
F18	Ave.	3	222.9682	3	3.590623333	2.24E+02
	Std Dev	0	9.96E-06	0	0.711917144	1.67E-05
F19	Ave.	-3.8628	22.7801	-3.8628	-2.670808	3.15E+01
	Std Dev	3.10862E-15	1.04E-02	3.11E-15	1.185307969	1.32E-03

As indicated in Table 3, mLPB surpassed Across nine test scenarios (F1–F2, F4–F6, F9, F11–F12, and F17), FDO and Leo all outperformed mLPB in two scenarios for each and were superior in all GOOSE and IFDO tests except the outcomes of F8 in unimodal and multimodal functions. This is demonstrated by the findings presented in the table. It tied with GOOSE results of functions F16, F18, and F19, outperforming FDO, Leo, and IFDO except for F4, F15, and F17, where mLPB does not perform poorly, but is just a little less effective than the other algorithms. on the whole, this suggests that mLPB is highly effective in evading local minima.

Table 4 *presents the experimental results of mLPB, SSA, CLPB, WOA, and FOX on the CEC'19 functions*

CEC	Indicator	mLPB	FOX	CLPB	WOA	SSA
CEC01	Ave.	30529827.432033	2.58E+04	1,151,565.313	4.11E+10	6.05E+09
	Std Dev	17899902.254976	22624.86	711,637.2073	5.42E+10	4.75E+09
CEC02	Ave.	17.3429	18.3442	17.34929	17.3495	18.3434
	Std Dev	0.00	0.000529	0.00574	0.0045	0.0005
CEC03	Ave.	12.7024	13.7025	12.7024	13.7024	13.7025
	Std Dev	0	0.000449	0	0	0.0003
CEC04	Ave.	7.02396	1.06E+03	3280.92051	394.6754	41.6936
	Std Dev	4.14440	501.8163	6.59052	248.5627	22.2191
CEC05	Ave.	1.04238	6.295	1.53148	2.7342	2.2084
	Std Dev	0.03688	1.27819	0.12728	0.2917	0.1064
CEC06	Ave.	7.309273333	5.0325	5.03164	10.7085	6.0798
	Std Dev	0.41501	1.285264	0.93160	1.0325	1.4873
CEC07	Ave.	8.543137	456.3214	201.42422	490.6843	410.3964
	Std Dev	111.772998	189.4313	109.39402	194.8318	290.5562
CEC08	Ave.	3.36130	5.6778	5.26162	6.909	6.371723
	Std Dev	0.62003	0.52774	0.50994	0.4269	0.5862
CEC09	Ave.	2.35111	3.7959	145.28040	5.9371	3.6704
	Std Dev	0.00461	0.339462	0.00075	1.6566	0.2362
CEC10	Ave.	20.05759	20.9878	20.09491	21.2761	21.04
	Std Dev	0.02469	0.005376	0.21368	0.1111	0.078

Comparisons of the mLPB algorithm with the current 19 conventional BTFs and the ranking of mLPB, FDO, GOOSE, Leo, and IFDO algorithms are shown in Tables 5 and 6. Table 6 lists the top five rankings of all the algorithms, and in Table 8, the rankings of the mLPB, FOX, CLPB, WOA, and SSA algorithms over 10 modern CEC'19 functions are tabulated. The rankings of all the algorithms are shown in Table 8. Tables 5 and 6 show that the mLPB algorithm outperforms the algorithm, achieving the highest first-rank score (8) and completely missing the fifth-rank (0). Moreover, mLPB demonstrates its efficacy by surpassing the FDO, GOOSE, Leo, and IFDO algorithms, achieving the highest rank on eight out of nineteen common benchmark functions. In addition, Table 7 shows the superiority of the mLPB algorithm over other algorithms, achieving the best performance on 7 out of 10 CEC'19 functions. To conduct a thorough assessment of the algorithm, mLPB is juxtaposed with the FDO, GOOSE, Leo, and IFDO algorithms, considering the nature of the benchmark function and overall efficacy for standard BTFs. Table 5 displays the mLPB rankings for several benchmark functions, both alone and together. To test and verify the findings achieved through the implementation of the suggested algorithm, certain mechanisms based on techniques already applied by other researchers were utilized. The performance of the mLPB algorithm was measured, with a rank of 2.1052 against 19 functions used as benchmarks. Various types of problems displayed certain ranking levels; for instance, mLPB was given a rank of 2.7142 against a collection of other algorithms on unimodal functions. mLPB specializes in finding new types of solutions with a score of 1.6666 on multimodal benchmark tasks and 1.8333 with the composite tasks. This is significant when compared to previous results achieved. As a result, this demonstrates that the algorithm can avoid local minima by conducting thorough research into beneficial locations within the search space and utilizing the optimal solution. Recognizing that no one method can produce optimal results for every optimization problem is a critical step in the optimization process. Therefore, certain algorithms will perform noticeably better than others, while others may underperform [24].

Table 5 *Comparative analysis of the mLPB algorithm*

BTF	1st	2nd	3rd	4th	5th	Rank	Partial total
F1	IFDO	mLPB	Leo	GOOSE	FDO	2	
F2	FDO	Leo	IFDO	mLPB	GOOSE	4	
F3	IFDO	FDO	Leo	mLPB	GOOSE	4	
F4	FDO	Leo	GOOSE	mLPB	IFDO	4	19
F5	mLPB	FDO	IFDO	Leo	GOOSE	1	
F6	FDO	Leo	mLPB	GOOSE	IFDO	3	
F7	mLPB	Leo	GOOSE	FDO	IFDO	1	
F8	mLPB	GOOSE	Leo	FDO	IFDO	1	
F9	mLPB	IFDO	FDO	Leo	GOOSE	1	
F10	IFDO	FDO	GOOSE	mLPB	Leo	4	10
F11	mLPB	FDO	GOOSE	Leo	IFDO	1	
F12	mLPB	IFDO	Leo	GOOSE	FDO	1	
F13	IFDO	mLPB	GOOSE	Leo	FDO	2	
F14	IFDO	mLPB	Leo	FDO	GOOSE	2	
F15	IFDO	mLPB	GOOSE	FDO	Leo	2	
F16	IFDO	mLPB	Leo	FDO	GOOSE	2	11
F17	IFDO	FDO	mLPB	Leo	GOOSE	3	
F18	mLPB	GOOSE	FDO	Leo	IFDO	1	
F19	mLPB	IFDO	GOOSE	FDO	Leo	1	
Total:						40	
Overall Rank:						40 /19 = 2.1052	
Unimodal Test Functions (TF1–TF7):						19 / 7 = 2.7142	
Multi-modal Test Functions (TF8–TF13):						10 / 6 = 1.6666	
Composite Test Functions (TF14–TF19):						11 / 6 = 1.8333	

Table 6 *mLPB, FDO, GOOSE, Leo, and IFDO total rankings on the standard BTFs*

Ranking Position	mLPB	FDO	GOOSE	Leo	IFDO
1st Place	8	3	0	0	8
2nd Place	5	5	2	4	3
3rd Place	2	2	7	6	2
4th Place	4	6	3	6	0
5th Place	0	3	7	3	6

Table 7 *mLPB, FOX, CLPB, WOA, and SSA ranking on the CEC'19 functions test*

BTF	mLPB	FOX	CLPB	WOA	SSA
CEC01	5	1	3	4	2
CEC02	1	5	3	2	4
CEC03	1	5	2	3	4
CEC04	1	5	3	2	4
CEC05	1	4	2	3	5
CEC06	3	1	2	5	4
CEC07	3	5	2	4	1
CEC08	1	5	2	4	3
CEC09	1	2	5	4	3
CEC10	1	4	2	3	5

Table 8 *mLPB, FOX, CLPB, WOA, and SSA total rankings on the CEC'19 functions*

Ranking Position	mLPB	FOX	CLPB	WOA	SSA
1st Place	7	2	0	0	1
2nd Place	0	1	6	2	1
3rd Place	2	0	3	3	2
4th Place	0	2	0	4	4
5th Place	1	5	1	1	2

5.3 Statistical Techniques Used

Wilcoxon Signed-Rank (WST) Test is used to check whether two observations came from the same population It is a type of pairwise test that sets out to identify whether the means of two samples or if the behaviors observed from an algorithm are significantly different. It means finding the absolute difference in performance between two algorithms on one of the n problems and dividing this by the scores of each algorithm. If the difference is below or equal to the Wilcoxon distribution value where it has n degrees of freedom, we do not accept the null hypothesis, and this means that one algorithm does better than the other. The test is statistically more powerful than the t-test. Intended to test continuous differences, they should not be rounded to one or two decimals [27].

Table 9 *Results of the statistical tests*

mLPB vs. LPB	
p_values	
Test Functions	**WST -sum test**
F1-F4	1.73E-06
F5	1.92E-06
F6	1.73E-06
F7	6.89E-05
F8-F13	1.73E-06
F14	1.00E+00
F15	3.18E-06
F16	1.00E+00
F17	5.00E-01
F18	1.28E-04
F19	1.00E+00

This can statistically determine the significance of results with the WST. The results from Table 9 for the classical test functions show that mLPB is notably better than LPB over most of this widely used test set. Its findings are also proven in reference [4]. Being much superior to PSO and GA, DA does not need a statistical comparison, as the advantage is obvious. All results were below 0.05, as shown in the table (except some cases of F3, F17, and F19 tests). It demonstrates why the results generated by the proposed algorithm matter.

5.4 Real-World Implementation

Like other metaheuristic algorithms, mLPB can tackle application-specific issues in practical situations. mLPB is employed in two distinct applications in this section:

5.4.1 Pressure Vessel Design Problem (PVDP)

PVDP is a frequently encountered engineering difficulty that many researchers have tackled using a number of different strategies. This problem seeks to minimize a pressure vessel's (the cylinder's) cost. The problem is to optimize the following four parameters: the shell thickness (Ts), inner radius (R), cylinder length without the head (L), and head thickness (Th). The details concerning this problem and constraints are presented below [30]:

$$n = 1,2,3,4$$

$$Min\ F1(x) = 0.6224_{x1x3x4} + 1.7781_{x2x_3^2} + 19.84_{x_1^2x3} + 3.1661_{x_1^2x4} \tag{10}$$

The variable limits are:

$$0 \le x_1 \le 99,$$

$$0 \le x_2 \le 99,$$

$$10 \le x_3 \le 200,$$

$$10 \le x_4 \le 200,$$

$$x = (x1, x2, x3, x4) = (T_s, T_h, R, L) \tag{11}$$

The constraints are applied as follows:

$$g_1(\vec{x}) = -x_1 + 0.0193_{x3} \le 0 \tag{12}$$

$$g_2(\vec{x}) = -x_3 + 0.00954_{x3} \le 0 \tag{13}$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^2 + 1{,}296{,}000 \le 0 \tag{14}$$

$$g_4(\vec{x}) = x_4 + 240 \ \le 0 \tag{15}$$

mLPB is employed to address this issue and attain the expected outcomes for its resolution. Consequently, three metaheuristic algorithms, namely WOA, FDO, and FOX, are employed to compare the results. Table 10 indicates that mLPB far surpasses WOA and FDO. mLPB produced adequate outcomes compared to the other algorithms. mLPB yielded the following results: 1.0625 forTs, 1.0453 forTh, 33.9608 for$R$, and 10 for$L$.

Table 10 *Comparison of WOA, FDO, and FOX with mLPB Performances for PVDP* [28]

Indicator	WOA	FDO	FOX	mLPB
Ave.	20070.54	63344.13	12026.01	12451.37
Std Dev	37401.74	90960.63	6544.212	4261.651

5.4.2 The Traveling Salesman Problem

Because it is challenging to solve the TSP in practice while it is simple to describe, the TSP is widely studied. It constitutes a complete NP problem. The problem is defined as follows: A salesman must visit several places, each with a certain distance or travel time between them. He seeks to determine the shortest possible path that allows him to visit every city only once and ultimately return to where he started. The subsequent mathematical formulation elucidates the Traveling Salesman Problem (TSP): Let us examine a graph in [31]G = (V, E), G is made up of two things: a set V of cities and a set E of edges with a weight (cost) assigned to each edge.

Minimize: $Cij(i,j) \in E$
Subject:

$$\sum_{n\neq 0} CijXij \tag{16.1}$$

$$\sum_{j=1}^{n} Xij = 1 \qquad (i \in V, i \neq j) \tag{16.2}$$

$$\sum_{i=1}^{n} Xij = 1 \qquad (ji \in V, j \neq i) \tag{16.3}$$

$$\sum_{i,j\,\in S} Xij \ \leq -1\ (\ S\ \subset V, 2 \leq |S| \leq n-2) \tag{16.4}$$

$$Xij = 0\ or\ 1\ (i,j) \leq A \tag{16.5}$$

According to equations (16.1) and (16.2)[5], every vertex j/i must be joined via an edge to another vertex i/j exactly once. However, these two needs do not guarantee that the model will give only one circuit. Consequently, equation (16.3) makes sure that the final solution does not consist of subsets of tours or sets of minor tours. It must make sure the route connects all the vertices. In equations (16.4) and (16.5), X shows if vertices (i,j) are linked by an edge in the final tour graph (1 if they are, 0 if they are not). The mLPB was tested against the LPB with 10 and 20 cities. The results in Table 11 show that mLPB outperformed the LPB in both cases. For 10 cities, mLPB found the best cost of 266.3594 vs. 328.9473 for LPB, and for 20 cities, mLPB found 379.9223 vs. 766.0628 for LPB. This gain is due to improvements in mLPB's search dynamics, especially in the exploitation mechanisms and diversity control, which are crucial in the TSP and help avoid premature convergence to suboptimal solutions.

Table 11 *LPB and mLPB applied to TSP for 10 and 20 cites [5]*

	Best Cost	
Algorithm	10 cities	20 cities
LPB	328.9473	766.0628
mLPB	266.3594	379.9223

6. Conclusion

The LPB algorithm shows an imbalance between exploration and exploitation. Alternatively stated, exploration is more desirable than exploitation. This work presents a new and adapted algorithm, namely mLPB. the modified mLPB algorithm improves several LPB directions by lowering the algorithm's processing time and refining the algorithm to strengthen the exploitation process, incorporating the crossover and mutation techniques of DE, and replacing the parameters in the GA that are built into LPB. The random population was divided into six groups representing different learner levels, then merged into a single group ordered in ascending order. The highest-performing individuals were selected to reduce the size, enhancing the algorithm's exploration phase. Thus, it can handle a wide range of optimization tasks because of its abilities in exploration and exploitation. BTF's results demonstrate that mLPB significantly enhances LPB's capabilities and competes effectively with established algorithms. Furthermore, mLPB advances the development of metaheuristic optimization algorithms by illustrating the efficacy of a hybrid approach. Accordingly, mLPB is better than LPB at exploring, avoiding local optima, execution time, and successfully establishing an effective balance between exploration and exploitation.

Acknowledgment

The authors would like to acknowledge the University of Information Technology and Communications (UoITC) and the Information Institute for Postgraduate Studies, Baghdad, Iraq, for supporting this work.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

Authors have contributed significantly to the research and writing of this paper, where Abbas M. Ahmed ***conceived the presented idea, developed the LPB optimization algorithm, and conducted the experiments.*** *Tarik A. Rashid* ***conceptualized and verified the analytical methods and contributed to interpreting the results****. The authors reviewed the results and approved the final version of the manuscript.*

References

[1] S. R. Kumar and K. D. Singh, "Nature-Inspired Optimization Algorithms: Research Direction and Survey." 2021. [Online]. Available: http://arxiv.org/abs/2102.04013

[2] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges," *Artificial Intelligence Review*, vol. 56, no. 11. pp. 13187–13257, 2023. doi: 10.1007/s10462-023-10470-y.

[3] J. Luo and B. Shi, "A hybrid whale optimization algorithm based on modified differential evolution for global optimization problems," *Applied Intelligence*, vol. 49, no. 5. pp. 1982–2000, 2019. doi: 10.1007/s10489-018-1362-4.

[4] C. M. Rahman and T. A. Rashid, "A new evolutionary algorithm: Learner performance based behavior algorithm," *Egypt. Informatics J.*, vol. 22, no. 2, pp. 213–223, 2021.

[5] "A LEARNER PERFORMANCE-BASED BEHAVIOR ALGORITHM FOR SOLVING TRAVELLING SALESMAN PROBLEM." Journal of University of Duhok, p. Pp 291-304, 6262 2023.

[6] C. Hu, H. Zhou, S. Lv, and S. K. Saraswat, "LPB-SPP: Solving the internet of things service placement problem using the learner performance-based behavior algorithm," *Trans. Emerg. Telecommun. Technol.*, vol. 34, 2023, [Online]. Available: https://api.semanticscholar.org/CorpusID:259592090

[7] X.-S. Yang, "Metaheuristic optimization," *Scholarpedia*, vol. 6, no. 8, p. 11472, 2011.

[8] J. O. Agushaka, A. E.-S. Ezugwu, A. K. Saha, J. Pal, L. M. Abualigah, and S. Mirjalili, "Greater cane rat algorithm (GCRA): A nature-inspired metaheuristic for optimization problems," *Heliyon*, vol. 10, 2024.

[9] K. Vijayabhaskar, S. Ramesh, and P. Chandrasekar, "Evolutionary Based Optimal Power Flow Solution For Load Congestion Using PRNG," *Int. J. Eng. Trends Technol.*, 2021.

[10] C. M. Rahman, T. A. Rashid, A. M. Ahmed, and S. Mirjalili, "Multi-objective learner performance-based behavior algorithm with five multi-objective real-world engineering problems," *Neural Comput. Appl.*, vol. 34, pp. 6307–6329, 2022.

[11] D. A. Franci and R. K. Hamad, "CLPB: Chaotic Learner Performance Based Behaviour," *ArXiv*, vol. abs/2407.0, 2024.

[12] A. M. Aladdin and T. A. Rashid, "A New Lagrangian Problem Crossover: A Systematic Review and Meta-Analysis of Crossover Standards," *Syst.*, vol. 11, p. 144, 2022.

[13] D. Hamad and T. Rashid, "LPBSA: Enhancing Optimization Efficiency through Learner Performance-based Behavior and Simulated Annealing." 2024. doi: 10.21203/rs.3.rs-3964302/v1.

[14] W. Y. Lin, "Optimum design of rolling element bearings using a genetic algorithm-differential evolution (GA-DE) hybrid algorithm," *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 225, no. 3, pp. 714–721, 2011, doi: 10.1243/09544062JMES2389.

[15] J. Yang, W. T. Li, X. W. Shi, L. Xin, and J. F. Yu, "A hybrid ABC-DE algorithm and its application for time-modulated arrays pattern synthesis," *IEEE Trans. Antennas Propag.*, vol. 61, no. 11, pp. 5485–5495, 2013, doi: 10.1109/TAP.2013.2279093.

[16] S. Sayah and A. Hamouda, "A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems," *Appl. Soft Comput. J.*, vol. 13, no. 4, pp. 1608–1619, 2013, doi: 10.1016/j.asoc.2012.12.014.

[17] S. A. Salih and T. A. Rashid, "Learner Performance-Based Behavior Optimization Algorithm: A Functional Case Study," in *International Conference on Interactive Collaborative Robotics*, 2022.

[18] Bilal, M. Pant, H. Zaheer, L. García-Hernández, and A. Abraham, "Differential Evolution: A review of more than two decades of research," *Eng. Appl. Artif. Intell.*, vol. 90, p. 103479, 2020.

[19] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, "An improved differential evolution algorithm and its application in optimization problem," *Soft Comput.*, vol. 25, pp. 5277–5298, 2021.

[20] A. W. Mohamed, H. Z. Sabry, and M. Khorshid, "An alternative differential evolution algorithm for global optimization," *J. Adv. Res.*, vol. 3, pp. 149–165, 2012.

[21] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Trans. Evol. Comput.*, vol. 15, pp. 4–31, 2011.

[22] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, pp. 1053–1073, 2015, [Online]. Available: https://api.semanticscholar.org/CorpusID:207018626

[23] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," *Proc. 2005 IEEE Swarm Intell. Symp. 2005. SIS 2005.*, pp. 68–75, 2005.

[24] E. M. Cortés-Toro, B. Crawford, J. A. Gómez-Pulido, R. Soto, and J. M. Lanza-Gutiérrez, "A new metaheuristic inspired by the vapour-liquid equilibrium for continuous optimization," *Appl. Sci.*, vol. 8, no. 11, p. 2080, 2018.

[25] D. A. Muhammed, S. A. M. Saeed, and T. A. Rashid, "Improved fitness-dependent optimizer algorithm," *IEEE Access*, vol. 8, pp. 19074–19088, 2020.

[26] H. Mohammed and T. Rashid, "FOX: a FOX-inspired optimization algorithm," *Appl. Intell.*, vol. 53, no. 1, pp. 1030–1050, 2023.

[27] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011, doi: 10.1016/j.swevo.2011.02.002.

[28] R. K. Hamad and T. A. Rashid, "GOOSE algorithm: a powerful optimization tool for real-world engineering challenges and beyond," *Evol. Syst.*, vol. 15, no. 4, pp. 1249–1274, 2024, doi: 10.1007/s12530-023-09553-6.

[29] A. M. Aladdin and T. A. Rashid, "Leo: Lagrange Elementary Optimization," *arXiv Prepr. arXiv2304.05346*, 2023.

[30] H. Mohammed and T. Rashid, "A novel hybrid GWO with WOA for global numerical optimization and solving pressure vessel design," *Neural Comput. Appl.*, vol. 32, no. 18, pp. 14701–14718, 2020.

[31] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: an overview of applications, formulations, and solution approaches," *Travel. Salesm. Probl. theory Appl.*, vol. 1, no. 1, pp. 1–25, 2010.