Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 6871298, 17 pages
https://doi.org/10.1155/2019/6871298

Research Article

A Modified Dragonfly Optimization Algorithm for Single- and Multiobjective Problems Using Brownian Motion

Çiğdem İnan Acı[1] **and Hakan Gülcan**[2]

[1]*Mersin University, Department of Computer Engineering, Mersin 33343, Turkey*
[2]*Mersin University, Department of Electrical-Electronics Engineering, Mersin 33343, Turkey*

Correspondence should be addressed to Çiğdem İnan Acı; caci@mersin.edu.tr

Received 6 February 2019; Accepted 6 May 2019; Published 2 June 2019

Academic Editor: Roman Bartak

The dragonfly algorithm (DA) is one of the optimization techniques developed in recent years. The random flying behavior of dragonflies in nature is modeled in the DA using the Levy flight mechanism (LFM). However, LFM has disadvantages such as the overflowing of the search area and interruption of random flights due to its big searching steps. In this study, an algorithm, known as the Brownian motion, is used to improve the randomization stage of the DA. The modified DA was applied to 15 single-objective and 6 multiobjective problems and then compared with the original algorithm. The modified DA provided up to 90% improvement compared to the original algorithm's minimum point access. The modified algorithm was also applied to welded beam design, a well-known benchmark problem, and thus was able to calculate the optimum cost 20% lower.

1. Introduction

The fact that the real-life problems have increased from the past to the present led the scientists to produce more effective solutions by using optimization algorithms. The search for these effective solutions brought about the understanding of the behaviors of swarms in nature. Scientists have developed various algorithms by following the behavior, experiences, and reactions of swarms in nature. These algorithms are known as swarm-inspired optimization algorithms.

So far, swarm-inspired optimization algorithms have successfully solved a lot of real-world problems: In 2013, the artificial bee colony (ABC) algorithm was used for data collection in wireless sensor networks [1], and the ant colony optimization (ACO) algorithm was used for multi-compartment vehicle routing problems [2]. In 2015, the ABC algorithm was used for brain tumor segmentation in MRI images [3], and the ACO algorithm was used in job scheduling [4], in economic dispatch problems [5] and for task-scheduling problems in cloud computing [6]. In 2017, the multilevel image thresholding problem was solved with the elephant herding optimization (EHO) algorithm [7], and the ACO algorithm was used in estimating transportation energy demand in Turkey [8]. The dragonfly algorithm (DA) was used in the synthesis of concentric circular antenna arrays by Babayigit [9], and the EHO algorithm was used in support vector machine parameter tuning [10]. Lastly, Debnath et al. [11] made an important study on access point planning for disaster scenarios by using the DA in 2018.

As a result of the modification of nature-inspired optimization algorithms and combining them with other optimization algorithms or methods, hybrid optimization algorithms, which provide better results than the original ones, were developed. Hybrid optimization algorithms also solved many problems in previous studies: In 2010, the ACO algorithm for solving a complex combinatorial optimization problem was modified by Yang and Zhuang [12], and the particle swarm optimization (PSO) for nonconvex economic dispatch problems was improved by Roh et al. [13]. In 2011, Yu et al. [14] improved the ACO algorithm for the multi-depot vehicle routing problem, and the ACO algorithm for constrained optimization problems was modified [15]. In 2012, the ACO algorithm on real-parameter optimization was improved [16], and Ishaque et al. [17] hybridized the PSO with the maximum power point tracking method for the photovoltaic system. In 2015, Forsati et al. [18] modified the ACO algorithm for document clustering. In 2016, Salam et al. [19] proposed a hybrid DA with an extreme learning machine for prediction, and lastly, a memory-based DA for numerical optimization problems was proposed in 2017 [20].

Random motion (randomization) is one of the most fundamental features in optimization algorithms to solve problems effectively. Random mobility ensures that there is no single way to solve the problem. The solution found during optimization suggests that even if it is the closest to the optimum and even optimal, random behavior can always be a better solution. This recommendation often prevents the best solution from getting stuck in the local best of problems. Another important benefit of random motion is the success of leaving no scanned space in the search space. If there is no random action, the optimization can only be installed in one region of the designated search space and may never see the results in other regions. Random motion increases the capacity of the algorithm to reach every field in the search space.

The classical random motion which is randomization based on a random number to be generated by the processor is commonly used in optimization algorithms. However, the occasional inadequacy of this classic randomness solution has led researchers to find new solutions. One of the new solutions to hybridize optimization algorithms with a random flight method is the Levy flight mechanism (LFM). The LFM also has a random number to be generated by the processor, but the mechanism is based on a statistical mathematical formula. There are many examples in the literature using LFM for randomization: In 2007, Pavlyukevich [21] used LFM in his research to theoretically validate and justify a new stochastic algorithm for global optimization. In 2008, Barthelemy et al. [22] used the LFM to optimize transmission and transmission of light. In 2009, Yang and Deb [23] implemented the LFM into the cuckoo search algorithm which is based on the obligate brood parasitic behavior of some cuckoo species in combination with the LFM behavior. Yang [24] worked on the firefly algorithm to adapt LFM, and the numerical results of his study proved that the proposed algorithm is superior to existing metaheuristic algorithms. In 2010, Lin et al. [25] proposed a bat algorithm with LFM for parameter estimation in nonlinear dynamic biological systems. Hakli and Uğuz [26] implemented the PSO algorithm with LFM in 2014. In that study, an improvement was achieved with LFM, and successful results were obtained due to the problem of early convergence of the agents during the optimization and localization of the agents. In 2017, Heidari and Pahlavani [27] adapted LFM to the gray wolf optimization. Similarly to the problem in PSO, they predicted that the lack of location of the wolves caused local minimization and solved this problem with the LFM.

The DA developed by Mirjalili [28] with the LFM was used to model the search process for the optimal solution of dragonflies when there is no neighborhood solution. Nevertheless, the random motion of dragonflies is intermittently interrupted by LFM and the step control mechanism within the algorithm. The LFM's very large searching steps caused interruption, and dragonflies could extend beyond the search space. In order to prevent overflowing, a step control mechanism was applied to the original algorithm. However, the step control mechanism is contrary to the original movement of the dragonflies and disrupts the nature of swarm behavior.

The main objective of this study is an adaptation of the Brownian motion to DA instead of LFM and its application to benchmark functions as available in the literature [29]. The goal of our study is to improve the performance of the DA and overcome the interruption problem caused by LFM. The reason for choosing the Brownian motion method is that its isotropic approach (completely independent of direction) increases the discovery capability. On the contrary, the sizes of the steps, both controllable size and time-based random form, prevent the outgoing of the search space, providing continuity of motion. In addition, there is only one study in which the Brownian motion has been used so far in the area of optimization: An optimization method was developed by using the Brownian motion of gas molecules in nature and very successful results were obtained in the study [29]. These results were compared with those of well-known heuristic algorithms such as PSO and genetic algorithm (GA). Within the scope of Abdechiri et al.'s [29] study, there are two aims to be achieved: (i) to increase the effect of random motion in metaheuristic algorithms and (ii) to present the contribution of the Brownian motion to swarm intelligence algorithms. Given the ease of implementation and the results of the previous study, the Brownian motion has a high potential to improve the performance of swarm-inspired optimization algorithms.

In this study, the randomization stage of DA is improved by means of the Brownian motion. The modified DA was compared with the original DA and tested in the optimization of single-objective and multiobjective benchmark functions. The results obtained from single-objective optimization functions were compared to the minimum point found, and the average values were calculated from 200 separate solutions of the benchmark functions. As a result of these comparisons, 11 of 15 benchmark functions managed to find better minimum points than the original DA. In multiobjective optimization, 5 of 6 benchmark functions achieved better results than the original DA in graphical results obtained from 100 iterations. The modified DA was finally applied to the welded beam design problem, which is a well-known real-life problem in the optimization field. According to the results, the modified DA found 20% better optimal cost than the original one. The rest of the paper is organized as follows: Section 2 presents detailed information about the materials and methods used in the study, Section 3 outlines the test methods and results of the study, and Section 4 concludes the paper.

2. Materials and Methods

2.1. Dragonfly Algorithm (DA). The DA was developed by Mirjalili at Griffith University in 2016 [28]. This technique, which is a metaheuristic algorithm based on swarm intelligence, is inspired by the static and dynamic behaviors of dragonflies in nature. There are two main stages of optimization: exploration and exploitation. These two phases were modeled by dragonflies, either dynamically or statically searching for food or avoiding the enemy.

There are two cases where swarm intelligence emerges in dragonflies: feeding and migration. Feeding is modeled as a static swarm in optimization; migration is modeled as a dynamic swarm. According to Craig and Hart [30], the swarms have three specific behaviors: separation, alignment, and cohesion. Here, the concept of separation means that an individual in the swarm avoids static collision with his neighbor (equation (1)). Alignment refers to the speed at which the agents are matched with the neighboring individuals (equation (2)). Finally, the concept of cohesion shows the tendency of individuals towards the centre of the herd (equation (3)).

Two additional behaviors are added to these three basic behaviors in DA: moving towards food and avoiding the enemy. The reason for adding these behaviors to the algorithm is that the main purpose of each swarm is to survive. Therefore, when all individuals are moving towards food sources (equation (4)), they must avoid the enemy in the same time period (equation (5)). Each of these behaviors is mathematically modeled as follows:

$$S_i = -\sum_{j=1}^{N} X - X_j, \tag{1}$$

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N}, \tag{2}$$

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X, \tag{3}$$

$$F_i = X^{+} - X, \tag{4}$$

$$E_i = X^{-} + X. \tag{5}$$

In the above equations, X represents the instantaneous position of the individual, while X_j represents the instantaneous position of the j^{th} individual. N represents the number of neighboring individuals, while V_j represents the speed of the j^{th} neighboring individual. X^{+} and X^{-} represent the location of the food source and enemy source, respectively.

In order to update the position of artificial dragonflies in the search space and simulate their motions, two vectors are considered: step (ΔX) and position (X). The step vector, which can also be considered as speed, indicates the direction of dragonfly motions (equation (6)). After calculating the step vector, the position vector is updated (equation (7)):

$$\nabla X_{t+1} = \left(sS_i + aA_i + cC_i + fF_i + eE_i\right) + w\nabla X_t, \tag{6}$$

$$X_{t+1} = X_t + \nabla X_{t+1}, \tag{7}$$

where the values of s, a, and c in equation (6) represent separation, alignment, and cohesion coefficients, respectively, and f, e, w, and t values represent the food factor, enemy factor, inertia coefficient, and iteration number, respectively. This coefficient and the mentioned factors enable to perform exploratory and exploitative behaviors during optimization. In the dynamic swarm, dragonflies tend to align their flight. In the static motion, the alignment is very low, while the fit to attack the enemy is very high. Therefore, the coefficient of alignment is high and the cohesion coefficient is low in the exploration process; in the exploitation process, the coefficient of alignment is low and the coefficient of cohesion is high.

2.2. Levy Flight Mechanism (LFM) and Dragonfly Algorithm.

LFM derives its name from the French mathematician Paul Levy. Technically, this mechanism has an infinite variance (possible length). Figure 1 shows the simulation of LFM in the first 1000 steps.

In order to improve the randomness, the probabilistic behavior, and the discovery of artificial dragonflies, a random walk (LFM) solution is reached when there is no neighborhood solution. Accordingly, the position of artificial dragonflies is updated as follows:

$$X_{t+1} = X_t + \text{Levy}(d) \times X_t, \tag{8}$$

$$\text{Levy}(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\beta}}, \tag{9}$$

$$\sigma = \left(\frac{\tau(1+\beta) \times \sin(\pi\beta/2)}{\tau((1+\beta)/2) \times \beta \times 2^{((\beta-1)/2)}}\right)^{1/\beta}, \tag{10}$$

$$\tau(x) = (x-1)!, \tag{11}$$

where d in equation (8) indicates the size of the position vector, $r1$ and $r2$ in equation (9) are random numbers in the range [0, 1], and β is a constant value.

In LFM used in DA, a multiplication, not included in the original mathematical formula of the flight method, was taken. This multiplication was obtained by taking 1% of LFM size as seen in equation (9). The aim here was to control the step size. This multiplication defines the value of a solution, which states the amount of the best individual deviation after LFM (position of the best individual). The 1% deviation value can be set according to the range of variables in the application. For example, if the range of variables in the application is $[-10e6, 10e6]$, the multiplication value of 1% can be set to 1.

Although LFM raises the performance of DA to a certain extent, it is disadvantageous in that very long steps may occur depending on the characteristics of the mechanism (Figure 1). These major steps are tried to be controlled in two ways in the algorithm: The first one is that if a long step is taken, the agents have to go outside the search space and a new step vector is produced. However, it is not certain that this solution will always give correct results. The new step to

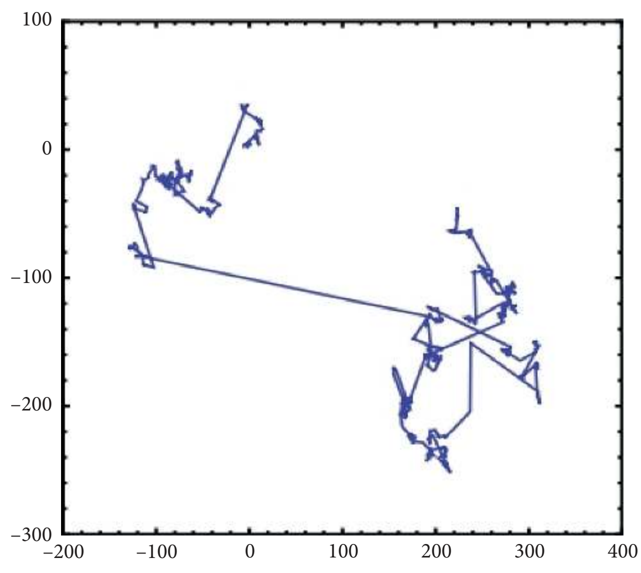


Figure 1: Simulation of the first 1000 steps of LFM.

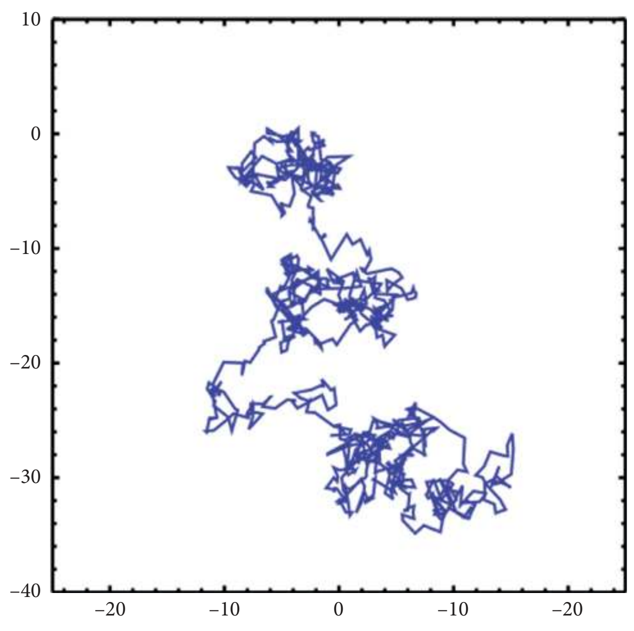


Figure 2: Simulation of the first 1000 steps of the Brownian motion.

be produced may take the general operation back. The second one is to take 1% of the step size as seen in equation (9) or a different percentage according to the variable range in the application. This solution method is better than the first solution. However, the step size control is contrary to the nature of LFM.

2.3. Brownian Motion.

2.3. Brownian Motion. Another one of the random motion mechanisms is the Brownian motion [31]. This method is inspired by the motion of free liquid/gas molecules. The introduction to the literature was done, thanks to Ingenhousz, observing the random motion of the coal and dust particles in 1779 while they were swimming in alcohol. The motion was discovered by the botanical scientist Robert Brown in 1828. The Brownian motion is defined as one of a variety of physical phenomena where a quantity continuously passes through small, random fluctuations. The Brownian motion is the random motion of particles suspended in a liquid (or a gas) resulting from collisions with fast moving molecules in the fluid. The Brownian motion is considered to be a Markov process with a Gaussian process and a continuous path that occurs continuously over time. Figure 2 shows an example of the Brownian motion in 1000 steps.

There are some basic differences between the LFM and the Brownian motion. Mathematically, a random walk can be defined as

$$X_{N+1} = X_N + W_N, \qquad (12)$$

where X_N is the solution that exists in step N and W_N is a random vector generated from a known probability distribution. If W_N is produced from the Gaussian distribution, random walking is isotropic. In this case, the motion takes the form of normal diffusion and is called the Brownian motion. The expected step size can be modeled with square root scaling as follows:

$$R(N) \propto \sqrt{N}. \qquad (13)$$

If the steps (W_N) are obtained from a predetermined tailed probability distribution, such as the Levy distribution or the Cauchy distribution, the diffusion becomes abnormal. In this case, the expected step size becomes

$$R(N) \propto N^q, \quad q > 0. \qquad (14)$$

If $q \geq 1/2$, the diffusion is called superdiffusion. Both the Levy distribution and the Cauchy distribution for step sizes may have some of the major steps leading to superdiffusion. This means that the average distance increases faster than that for normal diffusion.

2.4. Improvement of Dragonfly Algorithm with Brownian Motion. Modification of DA by the Brownian motion is expressed as follows:

$$X_{t+1} = X_t + h * \text{rand}\,() * P_\text{g}, \qquad (15)$$

$$h = \sqrt{\frac{T}{N}}, \qquad (16)$$

$$N = 100 * T, \qquad (17)$$

$$P_\text{g} = \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{(\text{dimension} - \text{agents})^2}{2h^2}\right). \qquad (18)$$

In equation (16), the term T represents the motion time period in seconds of an agent (dragonfly). In this study, the T value is taken as 0.01. The term N in equation (17) gives the number of sudden motions (change in the path) for the same agent in proportion to time. Unlike the LFM, the Brownian motion steps are chosen based on the normal (Gaussian) distribution instead of the dominant tailed distribution. The periodic motion of the dragonflies was spread over time with a normal distribution, and sudden jumps and random motions were made in the form of vibrations. The equation was modified by the size and number of agents in the algorithm, and the Brownian motion was finalized (equation (18)).

The modified DA has been adapted for both single- and multiobjective problems. Pseudocodes of the modified DA for single- and multiobjective problems are given in Figures 3 and 4. Figure 5 shows the flowchart of the modified DA for single-objective problems. The optimization is started by randomly placing dragonflies in the search space and identifying step vectors. Then, the current position is sent as a parameter to the comparison function. After that, the best and the worst solutions are determined. Following these solutions, the number of neighborhoods for each dragonfly is examined. If each dragonfly has at least one neighbor, the velocity vector is calculated with the coefficients determined at the beginning of the algorithm and the position vector is updated. If the dragonflies have no neighborhoods, the Brownian motion solution is used and the position vector is updated accordingly. Then, it is checked whether the dragonflies are in the search space. If the control is positive, it is checked whether the criterion is fulfilled. If the control is negative, then the neighborhood is solved. Then, again, it is checked whether the criterion of completion is achieved and the optimization is terminated.

The flowchart of the modified DA for multiobjective optimization is shown in Figure 6. Multiobjective optimization starts with placing dragonflies randomly in the search space and identifying step vectors. Then, the maximum archive size and the number of segments (hyperspheres) are defined. The instantaneous locations of the dragonflies are sent as a parameter to the comparison function, and the solutions that cannot be dominated by each other are produced. If the archive is full, some solutions are eliminated with the roulette wheel mechanism and new solutions are added to the archive. If any of the added solutions are left out of the hypersphere, the hypersphere is updated to cover all solutions. Following these processes, the best and the worst solutions in the archive are assigned as the source of nutrients and enemies, respectively. The neighborhood control is performed, and from there on, the algorithm works in the same way as the single-objective problem optimization.

3. Test Results and Discussions

In this section, the experimental evaluation of the modified DA is presented. The MATLAB [32] software was used for the solution of single-objective and multiobjective problem optimizations and the application of the welded beam design problem. In all cases, the computer used in simulations is configured with 2.2 GHz Intel Core i7 CPU and 6 GB DDR3 RAM. 15 benchmark functions were used for the optimization of the single-objective problems and 6 for the multiobjective problems. The results are compared and discussed with those of the original DA in terms of minimum point achieved and average performance.

3.1. Benchmark Functions. Test functions are useful for evaluating convergence rate, precision, robustness, and overall performance characteristics of optimization algorithms. Here, some test functions are presented to give an idea of different situations in any optimization algorithm that can be encountered when dealing with such problems. Only a general form of equality, limits of object variables, and global minimum coordinates are given here. Tables 1 and 2 show the benchmark functions used for single-objective and multiobjective problems.

3.2. Single-Objective Problem Optimization Results. The modified DA was tested by using 15 benchmark functions for single-objective problem optimization, and the minimum value for each test was compared with LFM results in the DA (the original DA). The Brownian motion in the modified algorithm was calculated separately by taking 1% of the original form and the calculated step size expressed in Section 2.3. The average value that is taken from 200 separate iterations of the original DA was compared with that of the two different versions of the modified DA: the Brownian motion and step size-controlled Brownian motion (SSCBM). The numerical results are shown in Table 3, and the graphical representation of the results is presented in Figures 7 and 8. When the results are evaluated in terms of minimum and average values, the following inferences can be made:

(i) In terms of minimum values, when the Brownian motion and SSCBM (the modified DA) are considered at the same time, they both gave better results than LFM (the original DA) in 11 of 15 benchmark functions. While one function did not change (*F*14) and one function reduced (*F*12), the worst (0.63%) and the best (99.98%) improvements were taken from *F*8 and *F*11, respectively.

(ii) In terms of minimum values, the results of the Brownian motion showed performance between 0.63% and 86.05% in 11 of 15 benchmark functions. In total, three functions (*F*3, *F*11, and *F*12) failed, while there was no improvement in a function (*F*14). When two functions (*F*2 and *F*10) with the greatest success are observed, the Brownian motion is effective enough to exceed 80% success in problems with sharp boundaries. When the worst three functions are considered, the Brownian motion is not an appropriate solution for global or especially early convergent problems.

(iii) When the results obtained from the modified DA with SSCBM were examined, they were better than those of the Brownian motion in 9 of 15 benchmark functions in terms of minimum values. SSCBM achieved over 99% success (*F*6 and *F*11) in benchmark functions that were clearly affected by step size control.

(iv) In terms of average values, 10 of 15 benchmark functions showed better results than LFM for both versions of the modified DA. When the performance results were examined without step size

```
while the end condition is not satisfied
    Calculate the objective values of dragonflies
    Update the food source, enemy source, motion weights, and neighboring radius
    if there is at least one neighbor of a dragonfly
        Update velocity and position
        if there is at least one dragonfly in the search space
        finish
        else
        go to neighboring condition
        end if
    else
    Apply the Brownian motion
    Go to search space condition
    end if
end while
```

Figure 3: Pseudocode of the modified DA for single-objective problems.

```
Initialize the dragonfly population and step vectors
Define the number of segments and archive size
while the end condition is not satisfied
    Calculate the objective values of all dragonflies
    Find the pareto optimal nondominated solutions and update the archive
    while the archive is full
        Run roulette wheel and insert the new solution to the archive
    end while
    while any of the solutions are not in the segment
        Update the segments to cover all solutions
    end while
    Select food source and enemy source from archive
    if there is at least one neighbor of a dragonfly
        Update velocity and position
        if there is at least one dragonfly in the search space
        finish
        else
        go to neighboring condition
        end if
    else
    Apply the Brownian motion
    Go to search space condition
    end if
end while
```

Figure 4: Pseudocode of the modified DA for multiobjective problems.

control, progression was observed in 12 of 15 benchmark functions between 0.50% and 88.62%. The three functions ($F1$, $F6$, and $F10$) could not show the desired average success after 200 iterations.

(v) The biggest success was obtained from $F7$ with 88.8% in terms of average values using SSCBM. It can be said that the local minimum is quite high and SSCBM is more successful than LFM in this type of function. When the worst three functions ($F1$, $F3$, and $F4$) are examined in this manner, it can be seen that although the minimum point in global functions is more optimal than LFM, this optimality is caused by random motion because these three functions are similar and there was a decline according to the average of 200 separate results.

(vi) Finally, the Brownian motion randomization (with or without step size control), in accordance with its nature, proved its success by giving better results than LFM in most of the benchmark functions.

3.3. Multiobjective Problem Optimization Results. The modified DA was applied to multiobjective problems given in Table 2, and 12 minimum values were obtained from 6 well-known benchmark functions used in the comparison. In the modified DA, the results were obtained from 100 iterations over 100 solutions which did not dominate each other after applying the Brownian motion step. The numerical $f1$ and $f2$ results are shown in Table 4, and graphical results are shown through Figures 9–14.

When the results are analyzed numerically, $f1$ values are mostly better in the original DA. This is because $f1$ minimization corresponds to the $x1$ position of artificial dragonflies in 5 of 6 problems. In the $f2$ minimization, the expected difference is observed. This is the phase where the

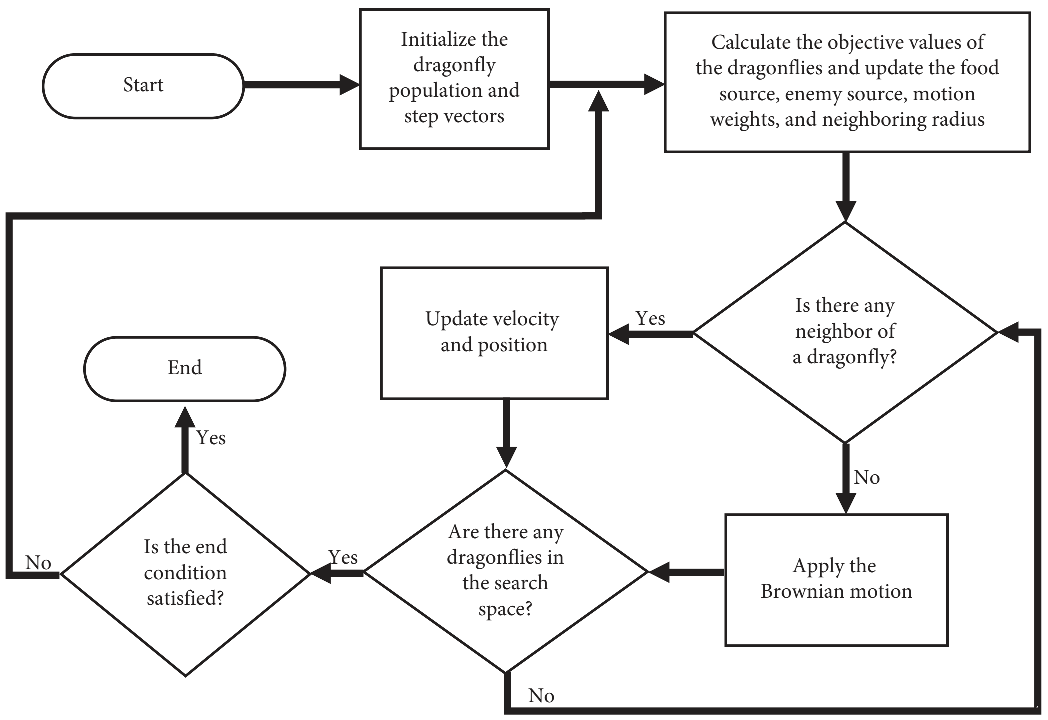

Start
Initialize the dragonfly population and step vectors
Calculate the objective values of the dragonflies and update the food source, enemy source, motion weights, and neighboring radius
Is there any neighbor of a dragonfly?
Yes
Update velocity and position
No
Apply the Brownian motion
Are there any dragonflies in the search space?
Yes
No
Is the end condition satisfied?
Yes
End
No

Figure 5: Flowchart of the modified DA for single-objective problems.

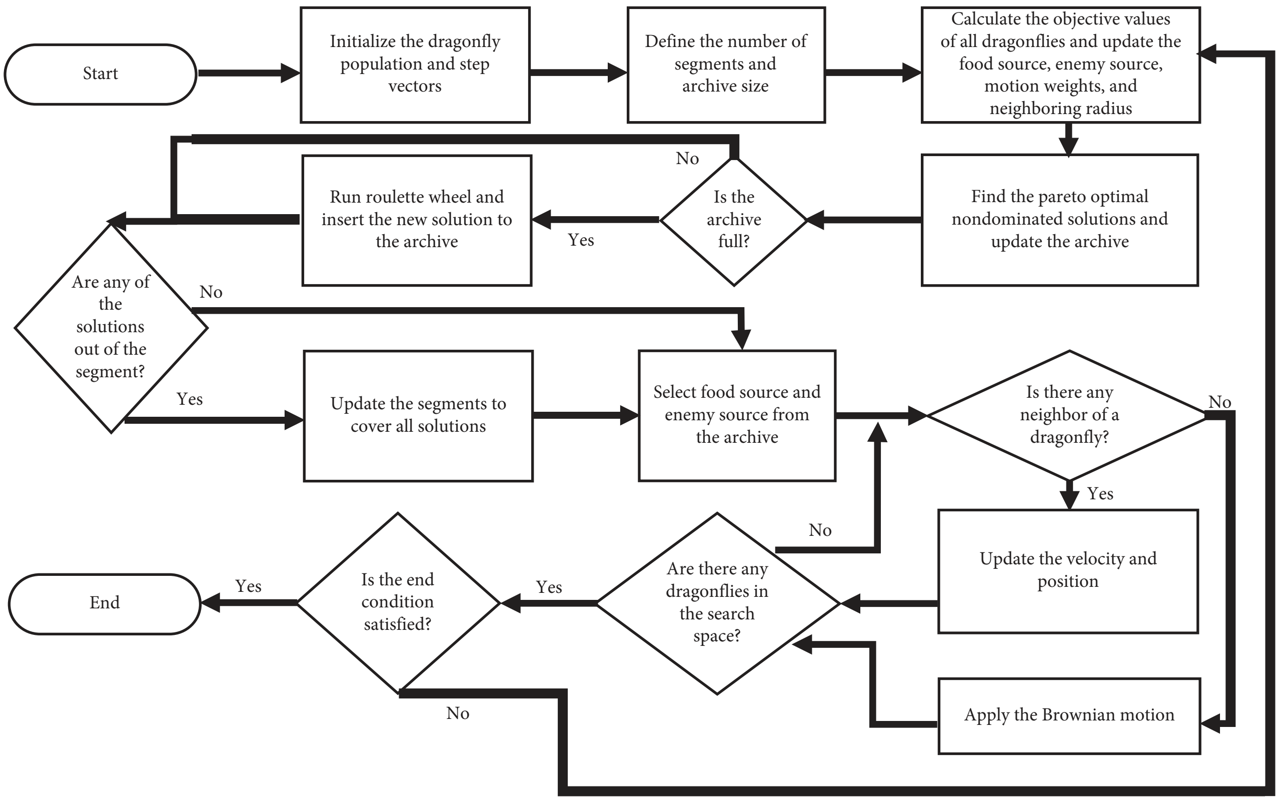

Start
Initialize the dragonfly population and step vectors
Define the number of segments and archive size
Calculate the objective values of all dragonflies and update the food source, enemy source, motion weights, and neighboring radius
Find the pareto optimal nondominated solutions and update the archive
Is the archive full?
Yes
No
Run roulette wheel and insert the new solution to the archive
Are any of the solutions out of the segment?
No
Yes
Update the segments to cover all solutions
Select food source and enemy source from the archive
Is there any neighbor of a dragonfly?
No
Yes
Update the velocity and position
Apply the Brownian motion
Are there any dragonflies in the search space?
No
Yes
Is the end condition satisfied?
Yes
End
No

Figure 6: Flowchart of the modified DA for multiobjective problems.

Table 1: Benchmark functions for single-objective problem optimization.

Function	Dimension	Range
$F1(x) = \sum_{i=1}^{n} x_i^2$	10	[−100, 100]
$F2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[−10, 10]
$F3(x) = \sum_{i=1}^{n} (\sum_{J-1}^{i} X_J)^2$	10	[−100, 100]
$F4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	10	[−100, 100]
$F5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	10	[−30, 30]
$F6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	10	[−100, 100]
$F7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	10	[−1.28, 1.28]
$F8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	10	[−500, 500]
$F9(x) = \sum_{i=0}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	10	[−5.12, 5.12]
$F10(x) = -20 \exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}) - (\exp(1/n)\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	10	[−32, 32]
$F11(x) = (1/4000)\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos((x_i/\sqrt{i}) + 1)$	10	[−600, 600]
$F12(x) = (\pi/n)\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+\sum_{i=1}^{n} u(x_i, 10, 100, 4) y_i = ((x_i + 1)/4) u(x_i, a, k, m) = \left\{\begin{array}{cc} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{array}\right\}$	10	[−50, 50]
$F13(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\}$ $+\sum_{i=1}^{n} u(x_i, 5, 100, 4)$	10	[−50, 50]
$F14(x) = \sum_{i=1}^{n} x_i^2$	10	[−5, 5]
$F15(x) = \sum_{i=1}^{n} ((x_i^2)/(4000)_i)^2 - \prod_{i=1}^{n} \cos((x_i/\sqrt{i}) + 1)$	10	[−5, 5]

Table 2: Benchmark functions for multiobjective problem optimization.

Problem	Definition
Zitzler–Deb–Thiele 1	Minimize $f_1(x) = x_1$ Minimize $f_2(x) = g(x) x h(f_1(x), g(x))$ where $g(x) = 1 + ((9/(N-1))\sum_{i=2}^{N} x_i)$ $h(f_1(x), g(x)) = 1 - \sqrt{((f_1(x))/(g(x)))}, 0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 2	Minimize $f_1(x) = x_1$ Minimize $f_2(x) = g(x) x h(f_1(x), g(x))$ where $g(x) = 1 + ((9/(N-1))\sum_{i=2}^{N} x_i)$ $h(f_1(x), g(x)) = 1 - ((f_1(x))/(g(x)))^2, 0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 1 (with linear pareto front)	Minimize $f_1(x) = x_1$ Minimize $f_2(x) = g(x) x h(f_1(x), g(x))$ where $g(x) = 1 + (9/(N-1))\sum_{i=2}^{N} x_i$ $h(f_1(x), g(x)) = 1 - ((f_1(x))/(g(x))), 0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 3	Minimize $f_1(x) = x_1$ Minimize $f_2(x) = g(x)\ x h(f_1(x), g(x))$ where $g(x) = 1 + (9/29)\sum_{i=2}^{N} x_i$ $h(f_1(x), g(x)) = 1 - \sqrt{(f_1(x))/(g(x))} - ((f_1(x))/(g(x)))\sin(10\pi f_1(x)), 0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 4	Minimize $f_1(x) = x_1$ Minimize $f_2(x) = g(x) x h(f_1(x), g(x))$ where $g(x) = 1 + 10(N-1) + \sum_{i=2}^{N}(x_i^2 - 10\ \sin(4\pi x_i))$ $h(f_1(x), g(x)) = 1 - \sqrt{(f_1(x))/(g(x))}, 0 \le x_i \le 1, 1 \le i \le 30$
Zitzler–Deb–Thiele 6	Minimize $f_1(x) = 1 - \exp(-4 * x_1) * \sin(6\pi x_1)^6$ Minimize $f_2(x) = g(x) x h(f_1(x), g(x))$ where $g(x) = 1 +\ 9((\sum_{i=2}^{N} x_i)/(N-1)^{0.25})$ $h(f_1(x), g(x)) = 1 - ((f_1(x))/(g(x)))^2, 0 \le x_i \le 1, 1 \le i \le 30$

Table 3: Comparative results of the single-objective benchmark functions.

Benchmark functions	Performance results			Percentage of success rate	
	LFM	SSCBM	Brownian motion	SSCBM	Brownian motion
*F*1 (min.)	5.56*E* − 06	4.89*E* − 07	2.29*E* − 06	91.20	58.72
*F*1 (avg.)	4.59*E* + 00	5.03*E* + 00	4.93*E* + 00	−9.71	−7.52
*F*2 (min.)	1.29*E* − 02	7.24*E* − 03	1.80*E* − 03	43.99	86.05
*F*2 (avg.)	1.46*E* + 00	1.26*E* + 00	8.68*E* − 01	13.59	40.69
*F*3 (min.)	8.39*E* − 02	3.03*E* − 02	1.02*E* − 01	63.88	−21.02
*F*3 (avg.)	1.38*E* + 02	1.63*E* + 02	8.40*E* + 01	−17.42	39.30
*F*4 (min.)	3.45*E* − 02	2.63*E* − 02	3.02*E* − 02	23.84	12.35
*F*4 (avg.)	1.91*E* + 00	2.08*E* + 00	1.78*E* + 00	−8.87	6.58
*F*5 (min.)	5.56*E* + 00	4.03*E* + 00	5.44*E* + 00	27.53	2.16
*F*5 (avg.)	1.74*E* + 03	1.45*E* + 03	5.74*E* + 02	16.97	67.06
*F*6 (min.)	4.10*E* − 07	3.04*E* − 09	1.65*E* − 07	99.26	59.69
*F*6 (avg.)	5.49*E* + 00	4.30*E* + 00	5.58*E* + 00	21.55	−1.65
*F*7 (min.)	1.41*E* − 03	1.01*E* − 03	1.09*E* − 03	28.48	23.27
*F*7 (avg.)	1.95*E* − 01	2.18*E* − 02	2.22*E* − 02	88.80	88.62
*F*8 (min.)	−3.89*E* + 03	−3.94*E* + 03	−3.92*E* + 03	1.26	0.63
*F*8 (avg.)	−2.82*E* + 03	−2.84*E* + 03	−2.93*E* + 03	1.05	4.22
*F*9 (min.)	2.99*E* + 00	2.61*E* + 00	2.46*E* + 00	12.83	17.89
*F*9 (avg.)	3.06*E* + 01	2.44*E* + 01	2.47*E* + 01	20.39	19.47
*F*10 (min.)	4.44*E* − 15	8.88*E* − 16	8.88*E* − 16	80.00	80.00
*F*10 (avg.)	2.28*E* + 00	2.26*E* + 00	2.32*E* + 00	0.63	−1.79
*F*11 (min.)	3.94*E* − 03	8.59*E* − 07	4.10*E* − 03	99.98	−4.07
*F*11 (avg.)	4.70*E* − 01	4.38*E* − 01	4.34*E* − 01	6.81	7.71
*F*12 (min.)	1.63*E* − 04	3.07*E* − 04	2.85*E* − 04	−88.69	−75.08
*F*12 (avg.)	1.29*E* + 00	1.20*E* + 00	1.29*E* + 00	7.11	0.50
*F*13 (min.)	6.70*E* − 05	4.23*E* − 05	3.08*E* − 05	36.80	54.05
*F*13 (avg.)	8.35*E* − 01	6.90*E* − 01	6.71*E* − 01	17.38	19.72
*F*14 (min.)	9.98*E* − 01	9.98*E* − 01	9.98*E* − 01	0.00	0.00
*F*14 (avg.)	1.25*E* + 00	1.23*E* + 00	1.20*E* + 00	1.97	3.97
*F*15 (min.)	3.41*E* − 04	3.08*E* − 04	3.13*E* − 04	9.83	8.09
*F*15 (avg.)	2.45*E* − 03	2.12*E* − 03	2.04*E* − 03	13.45	16.70

Note: min. = minimum; avg. = average.

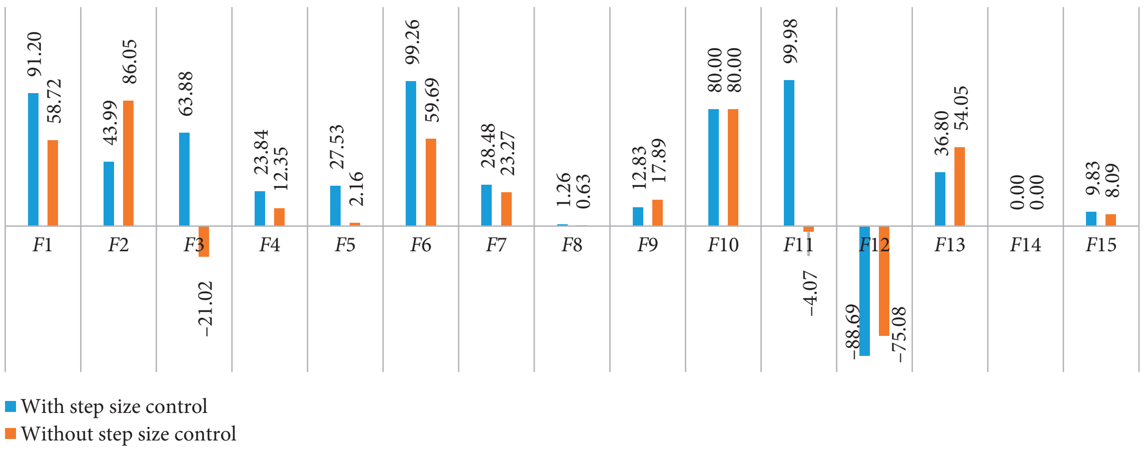


Figure 7: Percentage of the success rate of minimum values obtained from the modified DA.

real random motion of optimization is calculated to reach the minimum *f*2 value. Here, the Brownian motion achieved an average of 50% improvement in 5 out of 6 functions compared to LFM. If the statistical regression in a function (ZDT3) is examined, it is understood that the random motion should be applied stepwise in trigonometric rooted approaches.

When the results were analyzed graphically, the success of the Brownian motion in terms of convergence and coverage compared to LFM increased in proportion to the increase in size. While the DA with LFM was tested in 5 dimensions in the previous study [28], the search space was increased to 10 dimensions in order to increase the difficulty after it has been modified, and the Brownian motion clearly revealed its true success against the LFM.

The results of the modified DA were compared not only with those of the original algorithm but also with those of

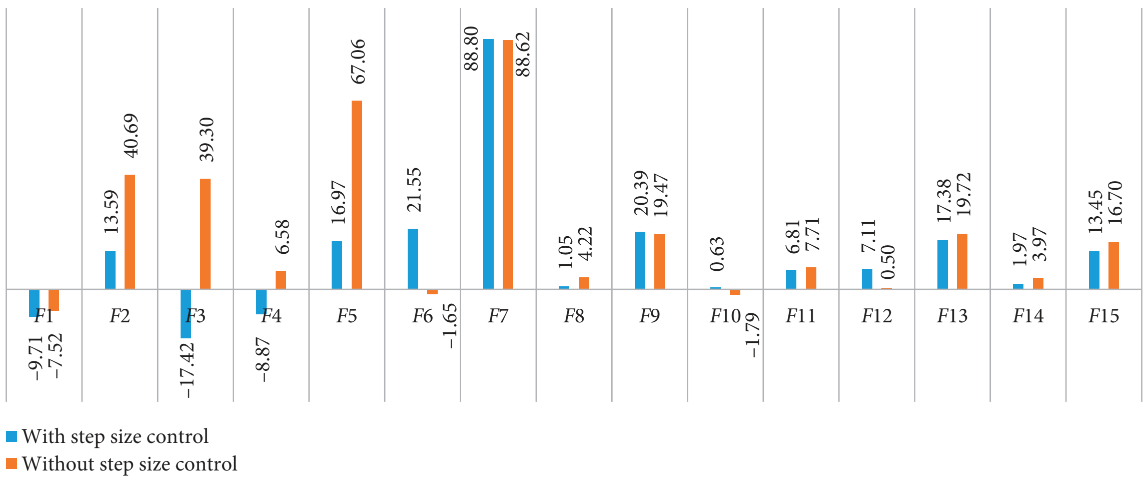


Figure 8: Percentage of the success rate of average values obtained from the modified DA.

Table 4: Results of the multiobjective benchmark functions.

Algorithm	ZDT1		ZDT2		ZDT1 linear		ZDT3		ZDT4		ZDT6	
	*f*1	*f*2	*f*1	*f*2	*f*1	*f*2	*f*1	*f*2	*f*1	*f*2	*f*1	*f*2
The original DA	0.20	2.00	0.15	1.57	0.07	1.04	0.40	0.28	0.14	2.24	0.19	1.71
The modified DA	0.21	0.50	0.50	0.72	0.55	0.45	0.22	0.43	0.20	0.60	0.61	0.91

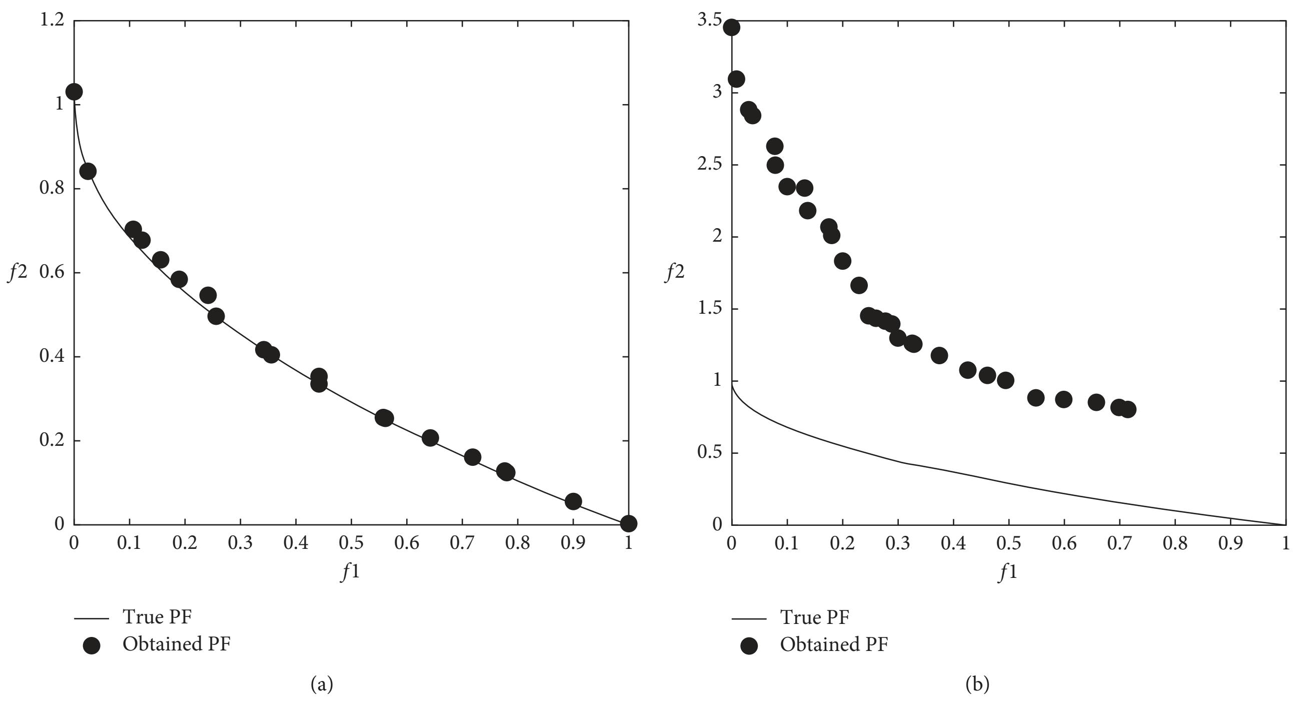


Figure 9: Pareto-optimal front comparison of ZDT1 optimization (PF = pareto front): (a) Brownian motion; (b) LFM.

some very important optimization algorithms such as GA, PSO, and ACO by means of basic statistics (i.e., mean and standard deviation).

Table 5 shows the results of 4 algorithms on 15 different benchmark functions. These results were carried out with a total of 500 iterations with 40 agents for each optimization method.

When the results are analyzed, the proposed method for solving the benchmark functions with an early convergence problem has produced similar and successful results with those of the ACO algorithm according to other algorithms. This means that the pheromone solution used by the ants is similar to the short-step solution of the

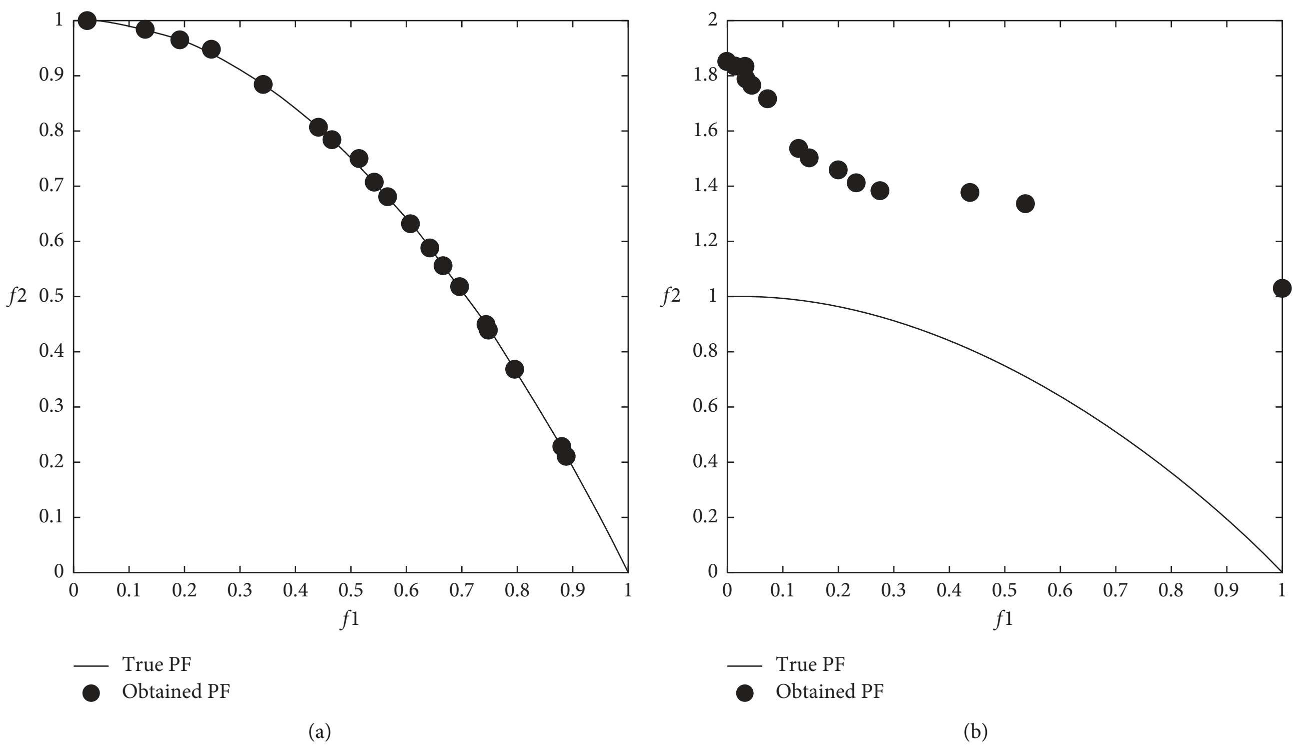


Figure 10: Pareto-optimal front comparison of ZDT2 optimization (PF = pareto front): (a) Brownian motion; (b) LFM.

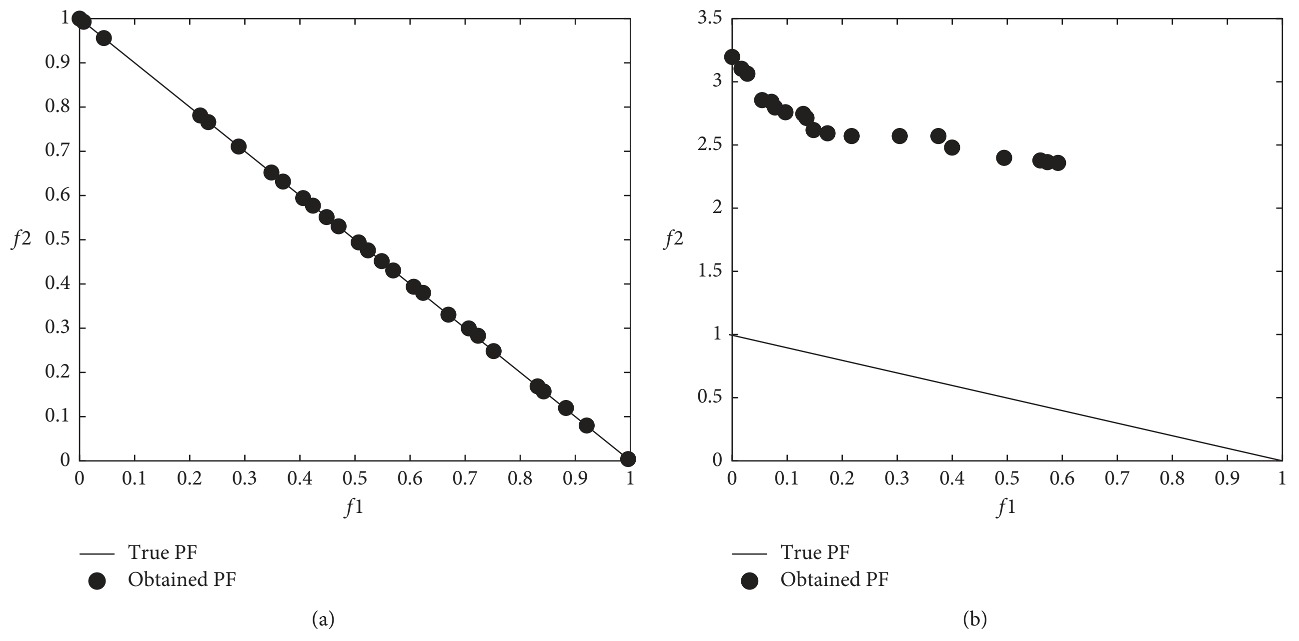


Figure 11: Pareto-optimal front comparison of ZDT1 linear optimization (PF = pareto front): (a) Brownian motion; (b) LFM.

Brownian motion. The pheromone, which keeps ants close to each other and increases their nutrient concentration, has shown the effect of the neighborhood radius on the Brownian motion.

PSO was more successful than other algorithms in terms of standard deviations. The reason for this is that as in the original dragonfly algorithm, the long steps extend the search space and rarely reach the result in a shorter time. On the contrary, the proposed method has been more successful than PSO on benchmark functions with local minima. The reason for this is that the short steps to different directions in the Brownian motion method allow the particles to discover different aspects and produce better solutions at the relevant time interval.

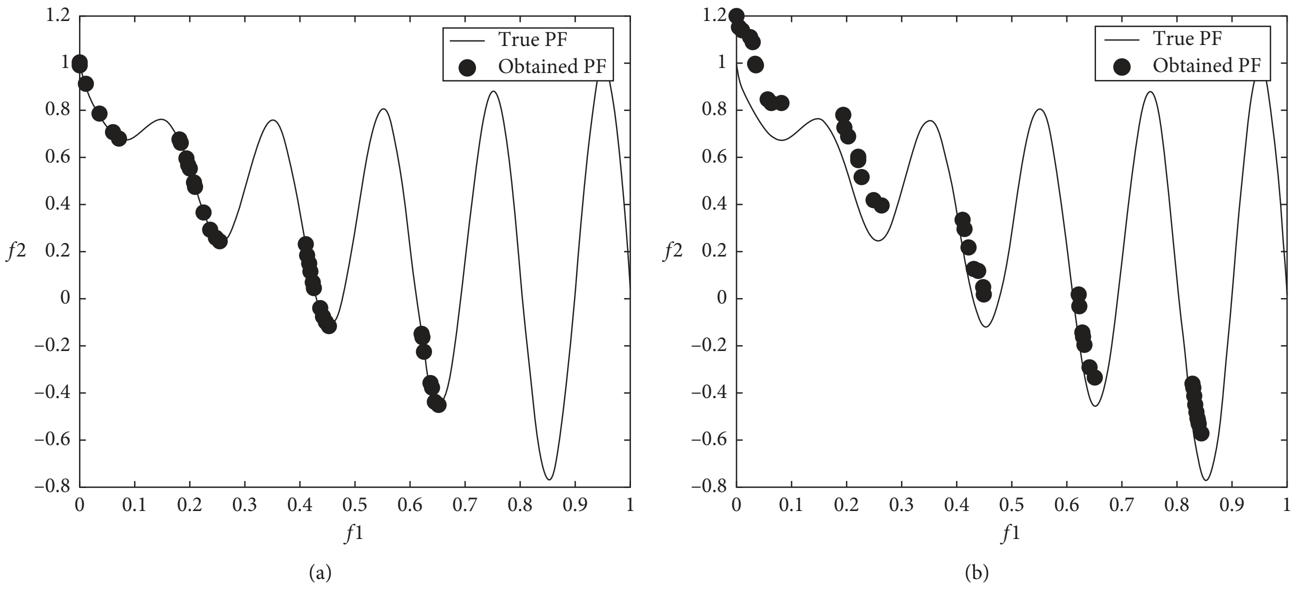


Figure 12: Pareto-optimal front comparison of ZDT3 optimization (PF = pareto front): (a) Brownian motion; (b) LFM.

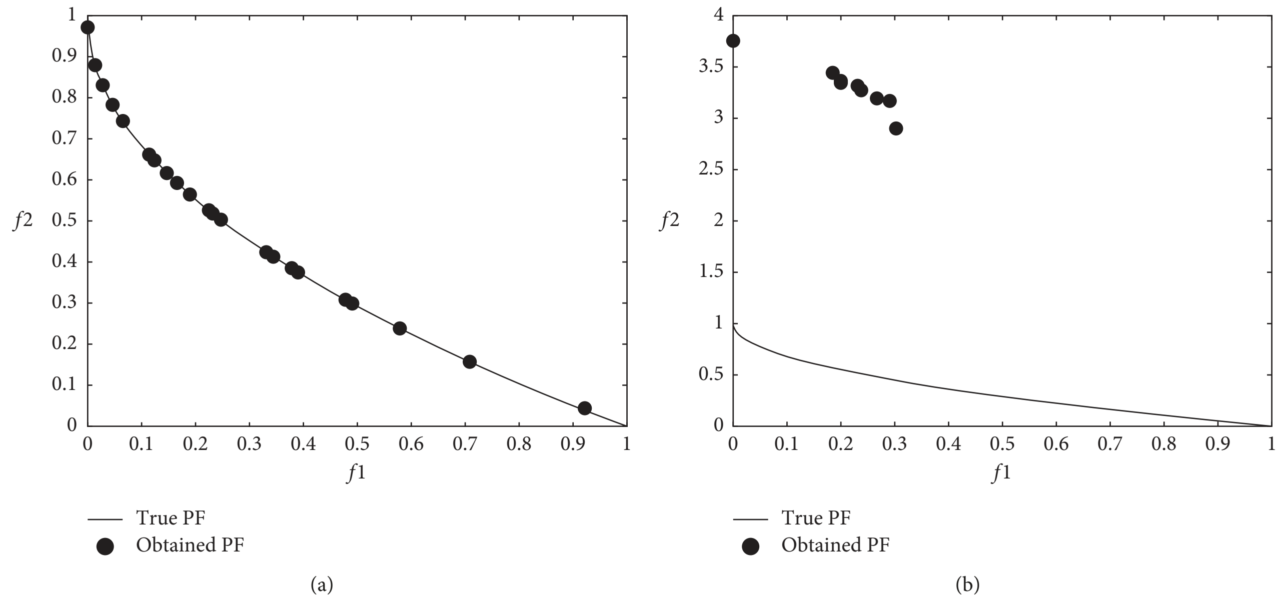


Figure 13: Pareto-optimal front comparison of ZDT4 optimization (PF = pareto front): (a) Brownian motion; (b) LFM.

3.4. A Solution of the Welded Beam Design Problem with the Modified Dragonfly Algorithm. The welded beam design is a practical design problem that is often used as a benchmark in testing different optimization techniques. The problem is an example of structural optimization problems, which consists of a nonlinear objective function and five nonlinear constraints [33]. The welded beam design problem was solved by many algorithms such as GA [34], simulated annealing [35], evolutionary strategy [36], and gravitational search algorithm [37].

In this study, the welded beam design problem was implemented in order to show the effectiveness of the modified DA. The reason for choosing this problem is that it was used many times as an application of hybrid swarm-inspired optimization techniques in the past. One of these is Kaveh and Talatahari's study [38] which hybridizes PSO and ACO. Another study is the application of the upgraded ACO [39], and Brajevic and Tuba [40] proposed a solution for limited engineering problems. Liao et al. [41] used it in the application of mixed-variable optimization problems, and finally, it is used by Ranjini and Murugan [20] for the memory-based modification of DA.

The welded beam design problem aims to minimize the manufacturing cost of the welded beam by finding a suitable

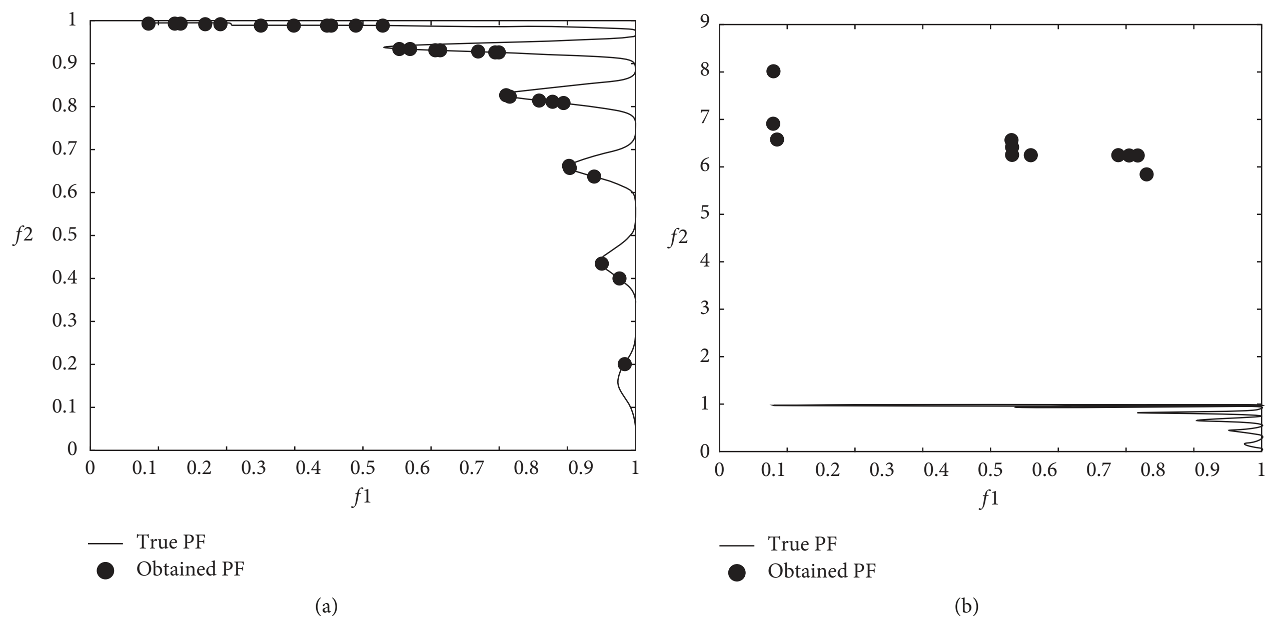


Figure 14: Pareto-optimal front comparison of ZDT6 optimization (PF = pareto front): (a) Brownian motion; (b) LFM.

Table 5: Comparison of the modified DA with well-known optimization algorithms.

Benchmark functions	The modified DA		ACO		GA		PSO	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
$F1$	$4.93E+00$	$7.16E-18$	$5.80E-19$	$9.85E-18$	$1.03E+03$	$4.47E+02$	$5.78E-18$	$1.80E-17$
$F2$	$8.68E-01$	$3.76E-05$	$3.09E-05$	$5.17E-05$	$8.21E+00$	$2.11E+00$	$4.34E-03$	$1.35E-02$
$F3$	$8.40E+01$	$2.10E-06$	$6.04E-06$	$2.89E-06$	$2.68E+03$	$1.37E+03$	$2.60E-03$	$4.55E-03$
$F4$	$1.78E+00$	$2.78E-03$	$2.15E-05$	$3.82E-03$	$2.91E+01$	$1.13E+01$	$2.40E-03$	$3.46E-03$
$F5$	$5.74E+02$	$6.79E+00$	$7.56E+00$	$9.33E+00$	$1.83E+05$	$1.17E+05$	$8.72E+01$	$1.10E+02$
$F6$	$5.58E+00$	$1.32E-15$	$5.73E-15$	$1.82E-15$	$7.75E+02$	$3.16E+02$	$6.00E-15$	$1.90E-16$
$F7$	$2.22E-02$	$4.69E-03$	$1.42E-02$	$6.45E-03$	$2.29E-01$	$9.98E-02$	$8.21E-03$	$4.93E-03$
$F8$	$-2.93E+03$	$3.84E+02$	$-3.93E+03$	$5.28E+02$	$-4.68E+03$	$1.11E+02$	$-9.76E+09$	$1.65E+12$
$F9$	$2.47E+01$	$9.48E+00$	$2.20E+01$	$1.30E+01$	$3.51E+01$	$9.17E+00$	$1.44E+01$	$1.08E+01$
$F10$	$2.32E+00$	$4.87E-01$	$3.18E-01$	$6.70E-01$	$1.31E+01$	$1.75E+00$	$3.85E-01$	$8.27E-01$
$F11$	$4.34E-01$	$7.35E-02$	$2.66E-02$	$1.01E-01$	$1.06E+01$	$4.99E+00$	$1.15E-01$	$4.82E-02$
$F12$	$1.29E+00$	$9.83E-02$	$4.28E-02$	$1.35E-01$	$2.56E+03$	$8.00E+03$	$1.18E-10$	$3.73E-10$
$F13$	$6.71E-01$	$4.63E-03$	$3.02E-03$	$6.37E-03$	$9.36E+04$	$1.21E+05$	$3.02E-03$	$6.37E-03$
$F14$	$1.20E+00$	$9.12E+01$	$1.42E+02$	$1.25E+02$	$1.79E+02$	$2.93E+01$	$2.06E+02$	$1.86E+02$
$F15$	$2.04E-03$	$8.06E+01$	$2.67E+01$	$1.11E+02$	$1.60E+02$	$2.64E+01$	$2.59E+02$	$2.16E+02$

Note: Std. = standard deviation.

set of four structural parameters of the beam. These four structural parameters are the thickness of the weld (x_1), the length of the clamped bar (x_2), the height of the bar (x_3), and the thickness of the bar (x_4). Relevant restrictions include shear stress (τ), bending stress (θ) in the beam, buckling load (P), and the last deflection of the beam (δ). Figure 15 shows the systematic design of the problem.

The total cost is equal to labor costs (a function of the source dimensions) and the cost of welding and beam material. The beam will be optimized for the minimum cost by changing the source and element dimensions ($x1$, $x2$, $x3$, and $x4$). The variables $x1$ and $x2$ are usually integer multiples of 0.0625 inches but are considered to be continuous for this application. The parameters and values of the problem are given in the following equations:

$$E = 30 \times 10^6 \text{ psi}, \quad (19)$$

$$G = 12 \times 10^6 \text{ psi}, \quad (20)$$

$$L = 14 \text{ inches}, \quad (21)$$

$$\tau_{\max} = 13600 \text{ psi}, \quad (22)$$

$$\sigma_{\max} = 30000 \text{ psi}, \quad (23)$$

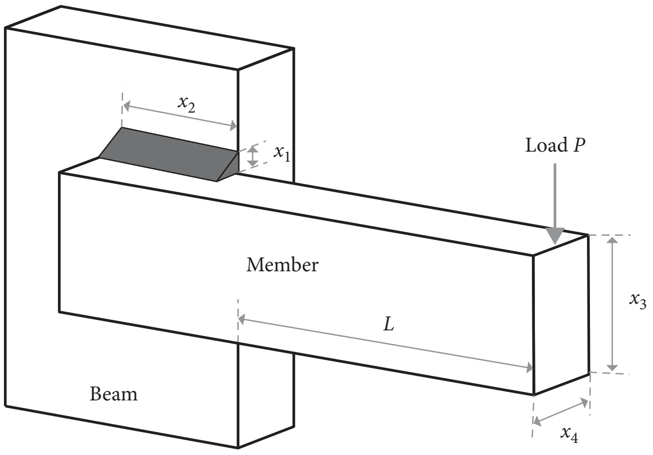


Figure 15: Welded beam design problem.

$$\delta_{\max} = 0.25\,\text{inches}, \quad (24)$$

$$P = 6000\,\text{lb}, \quad (25)$$

$$f(x) = C_0 + C_1 + C_2. \quad (26)$$

Young's modulus (psi) is given in equation (19), the shear modulus (psi) for the beam material in equation (20), the protrusion length (inches) of the member in equation (21), welding design stress (psi) in equation (22), normal design stress (psi) in equation (23) for the beam material, maximum deviation in equation (24), and load (lb) in equation (25). The cost function of the problem is shown in equation (26).

In equation (26), C_0 represents the initial cost, but it is assumed that the connections required for the installation and retention of the rod during welding are available. Therefore, the cost C_0 in the total cost model can be ignored. C_0 represents the cost of resources, and C_2 represents the cost of materials. The welded beam design problem was carried out with 40 dragonflies with 200 iterations. The average values were obtained after the optimization was performed 200 times. The results are shown in Table 6.

According to the results obtained, although the Brownian motion was less successful in terms of the minimum cost without step size controlling, it has almost the same result with LFM. The most significant success of the Brownian motion can be seen from the average values. When no step size controlling is applied to the algorithm, the Brownian motion is more successful than LFM. This means that the long premature jumps of LFM do not always have a positive effect. The step size controlling the movement of the Brownian motion increases the likelihood of reaching optimal results. On the contrary, the Brownian motion has yielded 20% more successful results than LFM in terms of minimum cost when 1% step size controlling has been applied. The success of the modified DA will be a guide for the solution of other real-world problems according to all these results.

3.5. Analysis of the Modified DA. In this study, long sudden jumps which are one of the main differences between LFM and Brownian motion were examined. As previously mentioned, long sudden steps are a solution that LFM produces to avoid early convergence. However, as a result of this solution, the step produced from time to time leads to the renewal of the step when it goes out of the search space. This causes time loss. At this point, the Brownian motion solution that we have implemented rescues the algorithm from these long steps.

The data obtained from Figure 16 show the long steps taken by 40 dragonflies through 1 iteration. Here, the long steps are taken as a result of the threshold value applied by taking the average distance from the steps taken for each method. As can be seen in the results, in the original DA with LFM, the long steps produced for each function are at least 5 and the average is 9.26. In the modified DA with Brownian motion, these numbers are at least 0 and average is 1.53. This is the main difference between the methods in this study, and our suggestion is that this difference often leads to success.

Even though the irregular jumps of LFM which is the motivation of the study are corrected by the Brownian motion, dragonflies may need sudden jumps to escape from the local minima. Although the proposed method greatly improves the irregular jumps caused by LFM from time to time, there is a rare possibility that it may get stuck in local minima. On the contrary, the number of neighbors in which the dragonflies have mass mobility has experienced a slight decrease with the Brownian motion. This, in a very rare way, can reduce communication groups and cause discovery to be restricted.

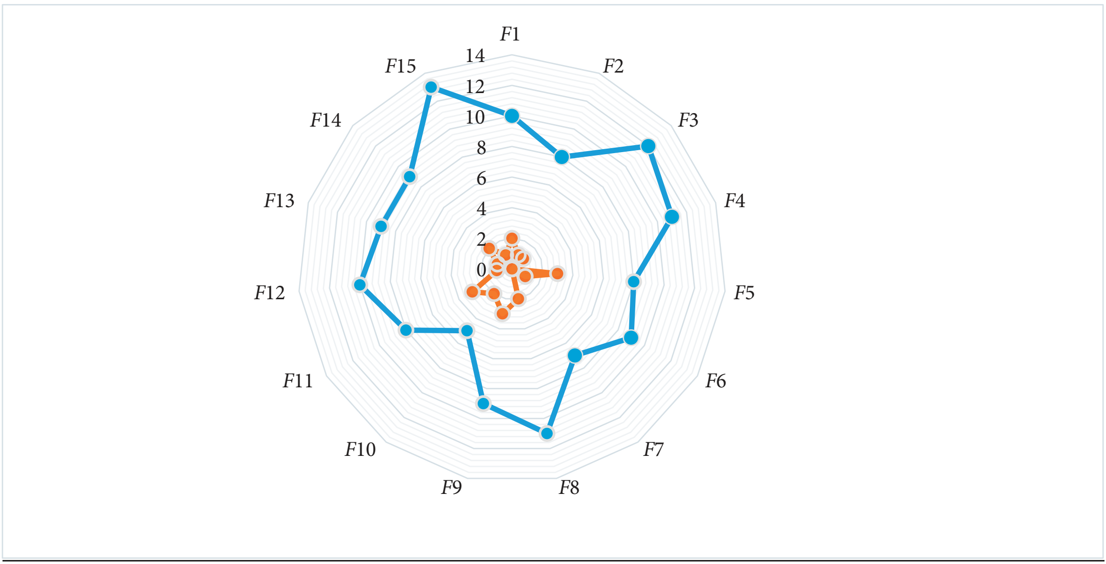
The modified DA has time complexity just like all other optimization algorithms. It depends on the population size and number of iterations. The overall complexity of the modified DA can be expressed as $O(\text{number of iterations} * \text{population size})$. Using Brownian motion instead of LFM did not affect the time complexity of the original algorithm. The goal here is to achieve the optimum result with the optimum number of dragonflies. Apart from the number of dragonflies, the dimension of the benchmark functions and the number of iterations are among the factors affecting the execution time. However, in contrast to the number of dragonflies, the concept of dimension has an inverse proportional effect with velocity. Table 7 compares the modified DA and the original DA in terms of execution time. The results of this comparison were obtained with 40 dragonfly agents, 15 different benchmark functions (all are in 10 dimensions), and 200 iterations.

When the results were examined, 7 of the total 15 functions improved by averagely 5%. On the contrary, there is an average of 3% regression observed for the remaining 8 functions. Two of the functions provided for improvement are outstanding: When $F5$ is examined, it has been observed that improvement in processing time is due to Brownian motion's short steps in different directions. When LFM was used, the fact that the step had to be dismissed because of the long jumps caused the execution time to be prolonged. Additionally, when $F8$ is examined from these functions, it is seen how the problem of early convergence which is the main optimization problem is solved with the Brownian motion. LFM used to solve this problem in the original

Table 6: Comparative optimum cost results of the welded beam design problem.

Algorithm	Without SSC		10% SSC		1% SSC	
	Min.	Avg.	Min.	Avg.	Min.	Avg.
The original DA	1.302	4689.090	1.253	1.985	1.252	1.718
The modified DA	1.293	1.956	1.252	2.079	1.204	1.930

Note: SSC = step size control; Min. = minimum; Avg. = average.



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
The original DA	10	8	12	11	8	9	7	11	9	5	8	10	9	9	13
The modified DA	2	1	1	0	3	1	0	2	3	2	3	1	1	2	1

●The original DA

●The modified DA

Figure 16: Comparison of long sudden jump counts of the original and modified algorithms.

Table 7: Comparison of execution times.

Function	Processing time (seconds)		Decrease in execution time
	The original DA	The modified DA	
*F*1	16.055	15.252	5%
*F*2	15.543	16.104	−4%
*F*3	15.643	15.742	−1%
*F*4	15.390	15.710	−2%
*F*5	17.125	15.430	10%
*F*6	14.883	15.290	−3%
*F*7	15.800	15.396	3%
*F*8	17.594	15.886	10%
*F*9	15.008	15.658	−4%
*F*10	15.145	14.758	3%
*F*11	15.307	15.034	2%
*F*12	16.439	16.000	3%
*F*13	15.389	15.994	−4%
*F*14	13.254	13.478	−2%
*F*15	14.675	15.221	−4%

algorithm does not always provide the exact solution as mentioned in the study. The Brownian motion, on the contrary, can be more successful in preventing getting stuck in local minima as it aims to go in different directions at every step. When *F*2 is examined from the observed functions, it is realized that the long steps in LFM worked this time and the short steps in the Brownian motion failed. The result to be taken as neutral here is that the success of the method may change according to the characteristic of the function.

4. Conclusions

Randomization is one of the essential elements of optimization techniques based on swarm intelligence. It plays a very important role in both exploration and exploitation stages. In this study, the randomization stage of DA, which is one of the swarm-based algorithms, used effectively in recent years, is modified with the Brownian motion. The results of the single-objective problem optimization from the obtained results clearly show that when the Brownian motion is used instead of LFM, definite success is achieved in the context of the minimum values in the benchmark functions. On the contrary, the modified DA was tested on 6 multiobjective problems. When numerical and graphical results are examined, it can be seen that the Brownian motion has significant success in 10-dimensional space compared to LFM.

As a general evaluation of the results, the long sudden steps from LFM sometimes caused the search space to be out of the search space. Therefore, the random motion had to be reproduced from the beginning. This is a costly time-consuming situation and it changes the original movement. However, with the Brownian motion, there is a significant reduction in the number of long steps in each iteration (Figure 16), and the original randomness is retained without having to repeat the movement. This has increased the importance of the neighborhood radius used in the original algorithm, both while saving time and facilitating the movement of dragonflies together.

In addition, the problem of getting stuck in the local minima (fast convergence) and the problem of infinite looping in the search field, as in most optimization algorithms and the original DA, have been significantly exceeded by the Brownian movement's principle of random motion generation in a different direction.

In addition to the success of LFM in relation to the usual random motion, this success of the Brownian motion also highlights the importance of random flight mechanisms. Random motion algorithms have a high effect on performance of optimization techniques. Finally, this success of the Brownian motion has shown another way to improve the results of other optimization techniques as the future work.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This academic work was linguistically supported by the Mersin Technology Transfer Office Academic Writing Centre of Mersin University.

References

[1] W.-L. Chang, D. Zeng, R.-C. Chen, and S. Guo, "An artificial bee colony algorithm for data collection path planning in sparse wireless sensor networks," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 3, pp. 375–383, 2015.

[2] M. Reed, A. Yiannakou, and R. Evering, "An ant colony algorithm for the multi-compartment vehicle routing problem," *Applied Soft Computing*, vol. 15, pp. 169–176, 2014.

[3] N. Menon and R. Ramakrishnan, "Brain tumor segmentation in MRI images using unsupervised artificial bee colony algorithm and FCM clustering," in *Proceedings of the 2015 International Conference on Communications and Signal Processing (ICCSP)*, Chengdu, China, April 2015.

[4] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, and V. Mahmoodian, "An ant colony algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs," *Computers & Industrial Engineering*, vol. 86, pp. 2–13, 2015.

[5] D. C. Secui, "A new modified artificial bee colony algorithm for the economic dispatch problem," *Energy Conversion and Management*, vol. 89, pp. 43–62, 2015.

[6] L. Zou, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, no. 1, pp. 2687–2699, 2015.

[7] E. Tuba, A. Alihodzic, and M. Tuba, "Multilevel image thresholding using elephant herding optimization algorithm," in *Proceedings of the 2017 14th International Conference on Engineering of Modern Electric Systems (EMES 2017)*, pp. 240–243, Oradea, Romania, June 2017.

[8] M. Sonmez, A. P. Akgüngör, and S. Bektaş, "Estimating transportation energy demand in Turkey using the artificial bee colony algorithm," *Energy*, vol. 122, pp. 301–310, 2017.

[9] B. Babayigit, "Synthesis of concentric circular antenna arrays using dragonfly algorithm," *International Journal of Electronics*, vol. 105, no. 5, pp. 784–793, 2018.

[10] E. Tuba and Z. Stanimirovic, "Elephant herding optimization algorithm for support vector machine parameters tuning," in *Proceedings of the 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI 2017)*, pp. 1–4, Piscataway, NJ, USA, June 2017.

[11] S. Debnath, A. Jee, and S. Baishya, "Access Point Planning for Disaster Scenario using Dragonfly Algorithm," in *Proceedings of the 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 226–231, Noida, India, February 2018.

[12] J. Yang and Y. Zhuang, "An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem," *Applied Soft Computing*, vol. 10, no. 2, pp. 653–660, 2010.

[13] J. H. Roh, M. J. Kim, H. Y. Song, J. B. Park, S. U. Lee, and S. Y. Son, "An improved particle swarm optimization for nonconvex economic dispatch problems," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 156–166, 2010.

[14] B. Yu, Z.-Z. Yang, and J.-X. Xie, "A parallel improved ant colony optimization for multi-depot vehicle routing problem," *Journal of the Operational Research Society*, vol. 62, no. 1, pp. 183–188, 2011.

[15] D. Karaboga and B. Akay, "A modified artificial bee colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing*, vol. 11, no. 3, pp. 3021–3031, 2011.

[16] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[17] K. Ishaque, Z. Salam, M. Amjad, and S. Mekhilef, "An improved particle swarm optimization (PSO)-based MPPT for PV with reduced steady-state oscillation," *IEEE Transactions on Power Electronics*, vol. 27, no. 8, pp. 3627–3638, 2012.

[18] R. Forsati, A. Keikha, and M. Shamsfard, "An improved bee colony optimization algorithm with an application to document clustering," *Neurocomputing*, vol. 159, no. 1, pp. 9–26, 2015.

[19] M. A. Salam, H. M. Zawbaa, E. Emary, K. K. A. Ghany, and B. Parv, "A hybrid dragonfly algorithm with extreme learning machine for prediction," in *Proceedings of the 2016 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA 2016)*, Sinaia, Romania, August 2016.

[20] K. S. S. Ranjini and S. Murugan, "Memory based hybrid dragonfly algorithm for numerical optimization problems," *Expert Systems with Applications*, vol. 83, pp. 63–78, 2017.

[21] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

[22] P. Barthelemy, J. Bertolotti, and D. S. Wiersma, "A Lévy flight for light," *Nature*, vol. 453, no. 7194, pp. 495–498, 2008.

[23] X.-S. Yang and S. Deb, "Cuckoo search via Levy flights," in *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, New Delhi, India, December 2009.

[24] X. Yang, "Firefly algorithm, Levy flights and global optimization," in *Research and Development in Intelligent Systems XXVI*, pp. 1–10, Springer, Cambridge, UK, 2009.

[25] J. Lin, C. Chou, C. Yang, and H. Tsai, "A chaotic Levy flight Bat algorithm for parameter estimation in nonlinear dynamic biological systems," *Computer and Information Technology*, vol. 2, no. 2, pp. 56–63, 2012.

[26] H. Hakli and H. Uğuz, "A novel particle swarm optimization algorithm with Levy flight," *Applied Soft Computing Journal*, vol. 23, pp. 333–345, 2014.

[27] A. A. Heidari and P. Pahlavani, "An efficient modified grey wolf optimizer with Lévy flight for optimization tasks," *Applied Soft Computing*, vol. 60, pp. 115–134, 2017.

[28] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[29] M. Abdechiri, M. R. Meybodi, and H. Bahrami, "Gases Brownian Motion optimization: an algorithm for optimization (GBMO)," *Applied Soft Computing*, vol. 13, no. 5, pp. 2932–2946, 2013.

[30] W. Craig and R. Hart, "Flocks, herds and schools: a distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[31] T. Hida, *Brownian Motion*, Springer US, New York, NY, USA, 1980.

[32] MathWorks, *MATLAB and Statistics Toolbox*, MathWorks, Natick, MA, USA, 2014.

[33] K. Deb, "Optimal design of a welded beam via genetic algorithms," *AIAA Journal*, vol. 29, no. 11, pp. 2013–2015, 1991.

[34] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[35] A.-R. Hedar and M. Fukushima, "Derivative-free filter simulated annealing method for constrained continuous global optimization," *Journal of Global Optimization*, vol. 35, no. 4, pp. 521–549, 2006.

[36] E. Mezura-Montes and C. A. C. Coello, "An empirical study about the usefulness of evolution strategies to solve constrained optimization problems," *International Journal of General Systems*, vol. 37, no. 4, pp. 443–473, 2008.

[37] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[38] A. Kaveh and S. Talatahari, "Engineering optimization with hybrid particle swarm and ant colony optimization," *Asian Journal of Civil Engineering*, vol. 10, no. 6, pp. 611–628, 2009.

[39] A. Kaveh and S. Talatahari, "An improved ant colony optimization for constrained engineering design problems," *Engineering Computations*, vol. 27, no. 1, pp. 155–182, 2010.

[40] I. Brajevic and M. Tuba, "An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems," *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 729–740, 2013.

[41] T. Liao, K. Socha, M. A. Montes de Oca, T. Stutzle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 503–518, 2014.