AI

MDPI

Article

Adaptive Exploration Artificial Bee Colony for Mathematical Optimization

Shaymaa Alsamia [1,2,*], Edina Koch [1], Hazim Albedran [2] and Richard Ray [1]

[1] Department of Structural and Geotechnical Engineering, Széchenyi István University, 9026 Győr, Hungary; koche@sze.hu (E.K.); ray@sze.hu (R.R.)
[2] Faculty of Engineering, University of Kufa, Najaf P.O. Box 21, Iraq; hazimn.bedran@uokufa.edu.iq
* Correspondence: alsamia.shaymaa.mahmood.badr@sze.hu

Abstract: The artificial bee colony (ABC) algorithm is a famous swarm intelligence method utilized across various disciplines due to its robustness. However, it exhibits limitations in exploration mechanisms, particularly in high-dimensional or complex landscapes. This article introduces the adaptive exploration artificial bee colony (AEABC), a novel variant that reinspires the ABC algorithm based on real-world phenomena. AEABC incorporates new distance-based parameters and mechanisms to correct the original design, enhancing its robustness. The performance of AEABC was evaluated against 33 state-of-the-art metaheuristics across twenty-five benchmark functions and an engineering application. AEABC consistently outperformed its counterparts, demonstrating superior efficiency and accuracy. In a variable-sized problem (n = 10), the traditional ABC algorithm converged to 3.086×10^{6}, while AEABC achieved a convergence of 2.0596×10^{-255}, highlighting its robust performance. By addressing the shortcomings of the traditional ABC algorithm, AEABC significantly advances mathematical optimization, especially in engineering applications. This work underscores the significance of the inspiration of the traditional ABC algorithm in enhancing the capabilities of swarm intelligence.

Keywords: artificial bee colony; optimization; swarm intelligence; metaheuristics; optimal design

Citation: Alsamia, S.; Koch, E.; Albedran, H.; Ray, R. Adaptive Exploration Artificial Bee Colony for Mathematical Optimization. *AI* **2024**, *5*, 2218–2236. https://doi.org/10.3390/ai5040109

Academic Editor: Giovanni Diraco

Received: 3 October 2024
Revised: 25 October 2024
Accepted: 1 November 2024
Published: 5 November 2024

1. Introduction

Optimization algorithms [1] use a technique to find the best solution in a space of candidate solutions. Since its inspiration by Dervis Karaboga in 2005 [2], the artificial bee colony (ABC) algorithm has been considered a cornerstone in swarm intelligence (SI). The ABC algorithm mimics the behavior of foraging honeybees, consisting of three categories: employed bees, onlookers, and scouts. The algorithm governs the exploration and exploitation processes, defining the search engine as globally optimal in a search landscape. Despite its proven efficiency, the ABC algorithm, like many contemporaries, presents some disadvantages [3]. Researchers have identified key areas, particularly in its exploration mechanisms, where it performs sub-optimally, particularly in high-dimensional or complex landscapes. This shortcoming has encouraged ongoing research on enhancing the algorithm's robustness through modifications and hybridizations with other metaheuristic approaches [4,5].

The inspiration engine of the artificial bee colony came with a significant issue: the concept did not consider that in real life, honeybees consider the distance between the exhausted source of food and the new proposed food source. Employed bees naturally tend to move to the closest food sources rather than with an equal probability of moving to the nearest or farthest ones. Ignoring this fact reduced the capability of ABC's inspiration engine and mathematical model (discussed in Section 3). Experiments discussed later also demonstrate this shortcoming.

Optimization has seen significant advancements in developing and applying various novel algorithms. These algorithms, which draw inspiration from natural phenomena,

biological processes, and physical laws, provide varied methods for addressing complex optimization problems in fields such as engineering [6], finance, optimal design [1], logistics, control [7], and artificial intelligence [8]. This paper provides an overview of some innovative optimization algorithms applied to the experiments while evaluating the adaptive exploration artificial bee colony (AEABC) algorithm. One of the critical algorithms is the Ant Lion Optimization (ALO) [9], developed based on the hunting behavior of antlions. On the other hand, the Bayesian Optimization Algorithm (BOA) [9] uses Bayesian techniques to guide the search for optimal solutions, providing a robust strategy for dealing with complex landscape explorations. The Gray Wolf Optimization (GWO) [10] imitates the hierarchy of leading and hunting behavior seen in a pack of grey wolves in the environment. The Particle Swarm Optimization (PSO) [11] is an algorithm inspired by social behavior in fish and birds. Many researchers have applied it due to its simplicity and effectiveness in navigating the search space.

Similarly, the Sine Cosine Algorithm (SCA) [12] utilizes mathematical functions to simulate the explorative moves of search agents. The Whale Optimization Algorithm (WOA) [13] was developed and inspired by the bubble-net hunting strategy of humpback whales. Another noteworthy algorithm is Dynamic Differential Annealed Optimization (DDAO) [14,15], which introduces dynamic elements into the differential optimization framework to enhance convergence speed and accuracy. The Bat Algorithm (BA) [16] models the echolocation behavior of bats, while the Firefly Algorithm (FF) [17,18] mimics the bioluminescent communication of fireflies. The Krill Herd (KH) algorithm [19] simulates the herding behavior of krill individuals in finding denser areas of food. The Multi-verse Optimizer (MVO) [20] originates from the theory of multi-verse in physics, representing multiple possible solutions through universes. The Squirrel Search Algorithm (SSA) [21] simulates the foraging behavior of squirrels. The gravitational search algorithm (GSA) [22] uses mass interactions and the law of gravity to perform the search, and the Dolphin Echolocation (DE) [23,24] simulates how dolphins identify and locate their prey through echolocation. Furthermore, the Flower Pollination Algorithm (FPA) [25] and the Fast Evolutionary Programming (FEP) [26] are influenced by the natural pollination process of flowers and evolutionary strategies, respectively. The State of Matter Search (SMS) [27], Moth-Flame Optimization (MFO) [28], and the Genetic Algorithm (GA) [29] draw from physics, the navigational method of moths in nature, and biological evolution principles, respectively. The Fertilization Algorithm (FO) [30] emulates the process of biological fertilization, and the Harmony Search (HS) is developed based on the improvisation process of musicians. Each of these algorithms has unique characteristics and has been applied successfully to solve specific optimization problems, demonstrating the richness and diversity of approaches in the optimization field.

This work introduces the adaptive exploration artificial bee colony (AEABC), a novel variant of the traditional ABC algorithm. Integrating new distance-based parameters into the bees' decision-making processes allows AEABC to navigate the search space more effectively, thereby improving the algorithm's efficiency and accuracy in finding a global optimal. This enhancement addresses the deficiencies observed in ABC's ability to handle complex mathematical optimization problems. This study performs a rigorous comparative analysis against 33 efficient metaheuristics across twenty-five benchmark functions and an engineering application. The results illustrate AEABC's superior performance, notably in scenarios involving large-scale optimization problems, where it dramatically outpaces the traditional ABC. Also, the experiments express the robustness and superiority of the AEABC algorithm on small-scale and constrained optimization problems. The following sections will guide the reader through the theoretical underpinnings of the AEABC, followed by detailed experimental results and analyses, highlighting the algorithm's versatility and robustness. By considering both the theoretical and practical aspects, this work contributes a substantial advancement to the field of mathematical optimization and demonstrates the applicability of AEABC in critical engineering contexts.

2. The ABC Algorithm

Dervis Karaboga [31] introduced an optimization method known as artificial bee colony, which utilizes the SI principles. The algorithm is a metaheuristic that successfully solves multidimensional optimization issues. The behavior of the honey bee colony's foraging mimics that based on a model presented by Tereshko and Loengarov [32]. The model of the ABC algorithm consists of the following components:

1. Employed bees: Each employed bee (proposed solution) is associated with a specific food source (solution), which it explores locally to find better positions (solutions). The employed bee randomly modifies the current solution to generate a new one in the neighborhood. If this new solution has a higher fitness value (indicating a more desirable solution), the food source's position is updated. Employed bees share information about the food source's quality with onlooker bees upon returning to the hive.
2. Onlooker bees: They remain in the hive and receive information from the employed bees; bees share their information using waggle dance [33]. In optimization, we can say that onlooker bees are proposed solutions that improve themselves based on the fitness values of the employed bee solutions.
3. Scout bees: When a food source (proposed solution) fails to improve after several attempts, it is abandoned. In other words, if a proposed solution does not improve over a specific number of iterations, then it should be deleted and replaced by a new randomly generated solution. The associated employed bee then becomes a scout and begins a random search for a new food source. This mechanism introduces randomness, helping the algorithm escape local optima and discover new, potentially more fruitful areas of the search space.
4. Food source: This is the location where nectar resides in multidimensional space; in optimization, it is a proposed solution that can be improved over iterations.

Bees in the ABC algorithm communicate through shared information about food source quality (fitness value of the solutions), similar to the "waggle dance" observed in natural bee colonies. Employed bees returning to the hive signal their food source's quality, allowing onlooker bees to make informed choices. This communication enhances the colony's search efficiency. In the ABC algorithm, the employed bee solutions communicate with onlooker solutions via Equation (3). Additionally, bees transition between roles dynamically: employed bees can become scouts, and scouts can become employed when discovering new sources. ABC algorithm (Algorithm 1) can be summarized as follows:

Algorithm 1: Pseudocodeof the ABC algorithm

```
Scout Bee Phase
do
Employed bee Phase
Onlooker bee Phase
If (Employed bee Phase and Onlooker bee Phase) have no improvement
Then, call for Scout bee Phase
Memorize the best solution in the current trial
Until (stop condition)
```

The low-level structures of the algorithm's sections interact to influence the global level. All bees can be considered scouts at the initialization stage, randomly looking for new source food (solution). Let x be a solution vector:

$$x = (x_1, x_2, x_i \ldots, x_{m-1}, x_m) \tag{1}$$

where $m \in R^n$ $j = 1 \ldots m$.

The foraging behavior in the ABC algorithm utilizes the following equation to express finding new solutions in the search space [2]:

$$v_{ij} = x_{ij} + \varphi_i(x_{ij} - x_{kj}) \tag{2}$$

where v_i is a new solution vector; φ_i is a randomly generated number between -1 and 1; and i (i = 1 to n) is the solution index in the (n) number of populations. j is the index of a variable in a solution, and k is a random number representing different random indexes in the population of solutions.

The onlooker bees employ a probability function as a function of the fitness value to choose the optimal option. The roulette wheel selection method [34] provided a means to accomplish this. The probability of a solution (P_i) should be:

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \tag{3}$$

$$f_i = \begin{Bmatrix} \frac{1}{1+O_i} & O_i \geq 0 \\ 1 + abs(O_i) & O_i < 0 \end{Bmatrix} \tag{4}$$

where f_i is the fitness value, and O_i is the value of the objective function. Initially, all the bees in the algorithm are considered scouts. However, as the algorithm progresses, these bees exchange roles between employed and onlookers. The bees whose solution remains unchanged after several trials must relinquish their employment and convert into scouts. The abandonment criteria, also known as limit control, play a crucial role in escaping local minimums and enabling the search for the global minimum in an optimization problem. The MathWorks website [35] provided the ABC algorithm's code implementation using MATLAB R2021a.

3. The AEABC Algorithm

With its three segments, the ABC algorithm does not pay attention to the distance to the food source. In other words, new solutions emerge using the neighborhood structure expressed in Equation (2). The random index k in Equation (2) means that a randomly chosen solution from the population provides the basis for a new solution. Thus, the original ABC algorithm ignored the bees' preference for the nearest food source if they had multiple choices. The far-source food has a lower probability than the probability of the nearest source food, and the bee leaves its current position for another position with a higher probability. This probability is calculated based on the distance between the current position of a bee and the randomly chosen position of another bee in the swarm. This probability parameter enhances the shared knowledge among the agents of the swarm. In other words, bees in the swarm share their positions with a random set of other bees, and that enhances the swarm's intelligence. We can say the same thing for onlooker bees in the hive when extracting information from employed bees through waggle dance; they consider the nearest food source if different employed bees come from various sources. The proposed algorithm, adaptive exploration artificial bee colony (AEABC), comes to redesign the original ABC algorithm to include the distance criteria in the mathematical model. In machine learning, various distance metrics measure the similarity or dissimilarity between data points, such as the Manhattan or Hamming distance. Euclidean distance can be used for Cartesian spaces:

$$d = \left\| x_{ij} - x_{kj} \right\| \tag{5}$$

where d is the distance between any solution in the population and a solution of random index k in the population. The distance probability Pd, based on the distance parameter, can be calculated as follows:

$$Pd = e^{-\frac{1}{d}} \tag{6}$$

The distance probability significantly impacts the search engine, as revealed in Section 5. If the random number $r \in [0,1] > Pd$, then the candidate solution generated by Equation (2) can be shifted as follows:

$$S_{ij} = r\, v_{ij} \tag{7}$$

where S_{ij} is the shifted solution. Table 1 explains how the distance between two solutions can affect the probability of shifting a solution in a search landscape.

Table 1. Probabilities of shifting a new solution based on its distance from the current population.

d	*Pd*	Description	Condition: Is ($r \in [0,1] > P$)
Long distance	$Pd = e^{-\left(\frac{1}{\text{High value}}\right)}$	High value of *Pd* (close to 1)	There is a lower probability that the current solution shifts according to Equation (7).
Short distance	$Pd = e^{-\left(\frac{1}{\text{Low value}}\right)}$	Low value of *Pd* (close to 0)	There is a higher probability that the current solution shifts according to Equation (7).

4. Designing and Executing the Experiment

Several benchmarks with different complexities provide a basis for comprehensively examining the AEABC algorithm. Also, the benchmarks supported other metaheuristics, allowing for their comparisons to the AEABC on a specific benchmark. Table 2 shows seven unimodal test functions with their variable size (D), search space (Range), and global minimum (fmin).

Table 2. Unimodal benchmarks.

Symbol	Description	Dimension	Range	fmin
F1	$f = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
F2	$f = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-100, 100]$	0
F3	$f = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	$[-100, 100]$	0
F4	$f = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	30	$[-100, 100]$	0
F5	$f = \sum_{i=1}^{n-1} \left(100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right)$	30	$[-100, 100]$	0
F6	$f = \sum_{i=1}^{n} \left([x_i + 0.5] \right)^2$	30	$[-100, 100]$	0
F7	$f = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	$[-1.28, 1.28]$	0

Table 3 reveals six multimodal benchmarks with a more complex search landscape with many local minima/maxima. The benchmarks in Tables 4 and 5 are used to evaluate the AEABC algorithm on large-scale and small-scale variable sizes, respectively where F1 to F7 refer to the test function number.

Table 3. Multimodal benchmarks.

Symbol	Description	Dimension	Range	fmin
F8	$f = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$[-500, 500]$	-418.9829×5
F9	$f = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi\, x_i) + 10\right]$	30	$[-5.12, 5.12]$	0
F10	$\mathrm{f} = -20\exp\left(-20\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]$	0
F11	$f = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 \Pi_i^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$	0
F12	$f = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n}(y_i-1)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, u, k, m) = \begin{cases} k(x_i-a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i-a)^m x_i < -a \end{cases}$	30	$[-50, 50]$	0
F13	$f = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right] + (x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

Table 4. Large-scale benchmarks.

Symbol	Description	Dimension	Range	fmin
F14	$f = \sum_{i=1}^{n} i x_i^2$	1000	$[-100, 100]$	0
F15	$f = \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2 + (x_{1-1})^2$	1000	$[-65.535, 65.535]$	0
F16	$f = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi\, x_i) + 10)$	1000	$[-5.12, 5.12]$	0
F17	$f = \sum_{i=1}^{D} \frac{(x_i - 100)^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$	1000	$[-600, 600]$	0
F18	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{i}{D}\sum_{i=1}^{D}\cos(2\pi\, x_i)\right) + 20 + e$	1000	$[-32.768, 32.768]$	0

Table 5. Two-dimension benchmarks.

Symbol	Description	Dimension	Range	fmin
F19	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi.x_1) - 0.4\cos(4\pi.x_2) + 0.7$	2	$[-100, 100]$	0
F20	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	2	$[-10, 10]$	0
F21	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1.x_2$	2	$[-10, 10]$	0
F22	$f(x) = -\cos(x_1).\cos(x_2).\exp\left(-(x_1-\pi)^2 - (x_2-\pi)\right)$	2	$[-100, 100]$	-1
F23	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	$[-4.5, 4.5]$	0
F24	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right) - 0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2}$	2	$[-100, 100]$	0
F25	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$	2	$[-2, 2]$	-1.03163

5. Contribution of the Distance Parameter

Applying distance parameters to a specific section of the ABC algorithm is crucial. Table 6 reveals that applying distance criteria to both employed and onlooker sections produces the best result versus applying it to one section. The experimental setup for results in Table 6 was performed for 30 independent runs, with a population size of 50 and

a maximum number of iterations of 500, while the variable size was changed to 100, 500, and 10, respectively. The best results in Table 6 are highlighted in bold.

Table 6. Effect of applying distance parameters to a specific section on the ABC algorithm.

Function	Variable Size	Mean	SD	*d* on Employed	*d* on Onlooker
F1	100	$\mathbf{1.0723 \times 10^{-239}}$	**0**	✓	✓
		1.516×10^{-115}	8.308×10^{-115}	✓	X
		6.1473×10^{-2}	1.0334×10^{-1}	X	✓
	500	$\mathbf{8.6509 \times 10^{-97}}$	$\mathbf{4.7383 \times 10^{-96}}$	✓	✓
		1.4291×10^{-47}	7.8273×10^{-47}	✓	X
		4.575×10^{-1}	8.540×10^{-1}	X	✓
	10	$\mathbf{7.9746 \times 10^{-83}}$	$\mathbf{4.3679 \times 10^{-82}}$	✓	✓
		7.487×10^{-30}	4.101×10^{-29}	✓	X
		7.552×10^{-1}	1.698	X	✓

The results indicate that applying the distance parameter in the ABC algorithm's employed and onlooker bee phases significantly enhances its performance, leading to lower mean values and minimal variability (as shown by the standard deviation). This effect is consistent across different variable sizes, with the best results occurring when the distance parameter is applied in both phases. When the distance parameter is applied only in the employed phase, the algorithm's performance is still relatively strong, though not as robust as when used in both phases. Conversely, applying the distance parameter only in the onlooker phase results in the worst performance, with higher mean values and greater variability, indicating that the onlooker phase alone is insufficient to capitalize on the benefits of the distance parameter. These findings underscore the importance of integrating the distance parameter across multiple phases of the ABC algorithm to achieve optimal performance, particularly in handling complex optimization tasks.

6. Experimental Results

This section presents how the new optimization algorithm AEABC is examined comprehensively by comparing its results with 33 other metaheuristic algorithms. The competitive internal parameters and their source codes in final versions can be free downloaded at [36,37], and this work is not responsible for tuning these competitive algorithms.

6.1. Time-Based Experiment

AEABC was evaluated on the seven unimodal benchmarks in Table 2 and compared with ABC, dynamic differential annealed optimization (DDAO) [14], Whale Optimization Algorithm (WOA), Sine Cosine Algorithm (SCA), Particle Swarm Optimization (PSO), Gray Wolf Optimization (GWO), Butterfly optimization algorithm (BOA), and Ant Lion Optimizer (ALO). All the algorithms used 30 independent runs, a population size of 50, and a maximum runtime of one second for each independent run. The results are shown in

Tables 7 and 8 express how the AEABC algorithm significantly improved its origin ABC and superior performance with other metaheuristics in the experiment. The AEABC algorithm consistently outperforms the other metaheuristics on most functions, as shown in Table 7, especially in terms of best values and standard deviations, indicating both high accuracy and consistency. Its performance is particularly notable on functions F1, F2, F3, and F4, where it significantly outclasses all other algorithms. On functions F5, F6, and F7, while AEABC is still highly competitive, a few other algorithms like GWO, WOA, and BOA show comparable or slightly better performance in certain cases. Consistency: The low standard deviations across all functions for AEABC suggest that it is highly reliable and produces consistent results, which is critical in optimization problems. Competitors: GWO

and WOA are the closest competitors to AEABC, performing well across most functions but still generally lagging behind AEABC in terms of best values. AEABC demonstrates superior performance across a wide range of test functions compared to other metaheuristic algorithms, making it a highly effective choice for optimization tasks.

Table 7. Evaluating AEABC on unimodal benchmarks.

Algorithm		F1	F2	F3	F4	F5	F6	F7
ALO	Best	6.151×10^{3}	3.210×10^{1}	9.584×10^{3}	2.654×10^{1}	1.245×10^{6}	5.550×10^{3}	1.317
	SD	2.958×10^{3}	1.113×10^{10}	7.199×10^{3}	4.653	2.856×10^{6}	2.623×10^{3}	1.403
BOA	Best	2.146×10^{-16}	1.149×10^{-36}	1.341×10^{-7}	1.655×10^{-12}	2.860×10^{1}	7.309×10^{-1}	7.975×10^{-4}
	SD	5.378×10^{-15}	9.932×10^{-12}	9.753×10^{-7}	1.216×10^{-11}	3.869×10^{-2}	4.699×10^{-1}	8.289×10^{-4}
GWO	Best	8.996×10^{-83}	1.334×10^{-48}	5.896×10^{-15}	1.976×10^{-19}	2.496×10^{1}	2.355×10^{-5}	2.056×10^{-4}
	SD	7.811×10^{-79}	2.491×10^{-46}	4.247×10^{-10}	2.566×10^{-8}	8.810×10^{-1}	2.621×10^{-1}	4.478×10^{-4}
PSO	Best	3.880×10^{-10}	1.952×10^{-5}	1.057×10^{1}	3.171×10^{-1}	1.238×10^{1}	1.142×10^{-10}	3.356×10^{-2}
	SD	2.004×10^{-6}	2.955×10^{-3}	1.032×10^{1}	1.735×10^{-1}	5.347×10^{1}	9.456×10^{-7}	3.545×10^{-2}
SCA	Best	7.230×10^{-7}	2.613×10^{-10}	3.019×10^{2}	3.428	2.782×10^{1}	3.877	1.485×10^{-3}
	SD	2.565×10^{-2}	9.183×10^{-6}	2.114×10^{3}	8.788	6.807×10^{2}	7.101×10^{-1}	2.838×10^{-2}
WOA	Best	1.343×10^{-228}	1.828×10^{-142}	4.146×10^{2}	1.579×10^{-6}	1.261×10^{-2}	1.222×10^{-9}	2.770×10^{-5}
	SD	0.0	2.481×10^{-129}	5.808×10^{3}	2.607×10^{1}	6.192	2.958×10^{-9}	1.383×10^{-3}
DDAO	Best	4.980×10^{-5}	3.869×10^{-3}	1.104×10^{-3}	3.915×10^{-3}	2.901×10^{1}	6.439	3.980×10^{-4}
	SD	1.090×10^{1}	1.291	4.291×10^{1}	3.344×10^{-1}	1.267×10^{1}	3.534	4.219×10^{-2}
ABC	Best	7.255	3.950×10^{-1}	2.341×10^{4}	4.091×10^{1}	1.777×10^{4}	7.264	2.332×10^{-1}
	SD	2.119×10^{1}	2.410×10^{-1}	4.527×10^{3}	5.401	7.007×10^{4}	5.955	1.906×10^{-1}
AEABC	Best	1.988×10^{-173}	7.755×10^{-119}	1.060×10^{-34}	3.258×10^{-86}	2.784×10^{1}	2.985×10^{-1}	1.309×10^{-5}
	SD	1.271×10^{-85}	1.393×10^{-105}	7.106×10^{-10}	6.232×10^{-64}	2.309×10^{-1}	5.818×10^{-2}	8.189×10^{-5}

Table 8. Evaluating AEABC on multimodal benchmarks.

Algorithm		F8	F9	F10	F11	F12	F13
ALO	Best	-5.537×10^{3}	2.337×10^{2}	1.381×10^{1}	5.618×10^{1}	9.461×10^{3}	1.381×10^{6}
	SD	4.409×10^{1}	2.110×10^{1}	9.181×10^{-1}	2.300×10^{1}	2.336×10^{6}	8.849×10^{6}
BOA	Best	-3.597×10^{3}	0.0	4.751×10^{-12}	0.0	1.675×10^{-1}	1.536
	SD	4.677×10^{2}	0.0	5.433×10^{-12}	0.0	6.462×10^{-2}	3.853×10^{-1}
GWO	Best	-7.369×10^{3}	0.0	7.994×10^{-15}	0.0	4.014×10^{-6}	1.017×10^{-1}
	SD	5.302×10^{2}	1.751×10^{1}	4.580×10^{-15}	1.061×10^{-2}	1.755×10^{-2}	1.664×10^{-1}
PSO	Best	-8.850×10^{3}	2.494×10^{1}	2.806×10^{-6}	2.452×10^{-11}	5.164×10^{-13}	1.584×10^{-11}
	SD	7.533×10^{2}	1.147×10^{1}	1.686×10^{-1}	9.501×10^{-3}	1.178×10^{-8}	4.987×10^{-3}
SCA	Best	-4.757×10^{3}	2.793×10^{-7}	7.387×10^{-4}	7.744×10^{-7}	4.028×10^{-1}	2.656
	SD	2.082×10^{2}	1.505×10^{1}	9.586	2.702×10^{-1}	3.481×10^{1}	4.884×10^{4}
WOA	Best	-1.257×10^{4}	0.0	8.882×10^{-16}	0.0	8.898×10^{-4}	3.633×10^{-2}
	SD	5.622×10^{2}	0.0	2.312×10^{-15}	2.177×10^{-3}	8.655×10^{-3}	9.132×10^{-2}
DDAO	Best	-5.214×10^{3}	1.089×10^{-4}	5.817×10^{-2}	6.800×10^{-4}	1.147	3.015
	SD	3.388×10^{2}	4.450	9.099×10^{-1}	3.915×10^{-1}	3.337×10^{-1}	1.463
ABC	Best	-5.700×10^{3}	1.664×10^{2}	3.005	1.341	5.297×10^{4}	9.635×10^{5}
	SD	2.439×10^{2}	1.776×10^{1}	5.221×10^{-1}	1.804×10^{-1}	1.025×10^{6}	2.727×10^{6}
AEABC	Best	-6.428×10^{3}	0.0	8.882×10^{-16}	0.0	4.855×10^{-2}	8.790×10^{-1}
	SD	4.076×10^{2}	0.0	0.0	0.0	2.540×10^{-2}	4.212×10^{-1}

Table 8 reveals that AEABC is among the top performers, particularly on functions F9, F10, and F11, where it achieves near-perfect scores with zero variation. This indicates that

AEABC is highly effective for certain types of multimodal functions. The zero standard deviation in functions F9, F10, and F11 highlights AEABC's consistency and reliability in optimization. For other functions like F8 and F13, its standard deviation is moderate, suggesting that while its performance is good, there is some variability. While AEABC is competitive, especially regarding consistency, algorithms like WOA and GWO sometimes achieve slightly better best values, particularly for F8 and F12. However, AEABC's balance of good performance across functions and low standard deviation makes it a robust choice. Overall, AEABC demonstrates strong and consistent performance across a range of multimodal benchmark functions, making it a competitive algorithm. Its particular strength lies in its ability to find optimal or near-optimal solutions consistently, even though it might not always achieve the absolute best performance on every function compared to other top algorithms like WOA or GWO.

6.2. Comparison Against ABC

This section demonstrates how the AEABC algorithm can outperform its previous form by examining it in large-scale optimization. The variable size for the functions in Table 9 is 1000, the run condition was 30 independent runs, the population size is 100, and 1000 is the maximum number of iterations.

Table 9. AEABC against its ancestor on large-scale optimization.

Function	ABC		AEABC	
	Mean	**SD**	**Mean**	**SD**
F1	3.086×10^{6}	4.975×10^{4}	2.0596×10^{-255}	0
F14	4.616×10^{-16}	2.063×10^{-16}	4.091×10^{-50}	1.9×10^{-49}
F15	3.542×10^{9}	7.708×10^{7}	6.76×10^{-1}	4.081×10^{-3}
F16	1.68×10^{4}	2.132×10^{2}	0	0
F17	2.758×10^{4}	6.663×10^{2}	3.526×10^{-1}	1.024×10^{-1}
F18	8.881×10^{-16}	0	8.881×10^{-16}	0

AEABC shows a dramatic improvement in the mean values across all functions compared to the original ABC algorithm. For functions F1, F14, F15, F16, and F17, the reductions in error are several orders of magnitude, indicating that AEABC is far more effective in solving these large-scale optimization problems. AEABC consistently shows a lower standard deviation across all functions, often reaching zero, suggesting that it performs better on average and does so with much greater reliability and stability. This consistency is essential in optimization tasks where predictable performance is crucial. In function F18, both algorithms perform equally well, suggesting that AEABC does not lose any effectiveness compared to ABC on simpler tasks, even though it shows significant improvements in more complex ones. AEABC outperforms its predecessor, ABC, across almost all tested functions, particularly in accuracy and consistency. The improvements are most notable in more challenging functions, making AEABC a more powerful tool for large-scale optimization problems.

6.3. AEABC on F1 and F16

Comparison results were found in the literature [14,21] among a set of metaheuristics on F1 and F16 benchmarks and used to evaluate the performance of the AEABC algorithm as shown in Table 10. We have followed the same running conditions for a fair comparison with 25,000 function evaluations, 30 independent runs, and a variable size of 30. Table 10 shows that AEABC outperforms its competitors, which are Bat Algorithm (BA), Firefly (FF) optimization, Multi-Verse Optimizer (MVO), Krill Herd (KH) optimization, and Squirrel Search Algorithm (SSA). The optimal solutions of the test functions F1 and F16 are zero,

and AEABC converged perfectly to zero, while other algorithms in this experiment could not converge well.

Table 10. Statistical results of different metaheuristics and AEABC on F1 and F16 benchmarks.

Function	Metric	PSO	BA	FF	MVO	KH	SSA	DDAO	AEABC
F1	Mean	1.35×10^{3}	3.93×10^{4}	1.15×10^{-2}	7.85×10^{-1}	5.75×10^{-2}	4.16×10^{-8}	0.086	0
	SD	6.42×10^{2}	1.07×10^{4}	4.32×10^{-3}	2.47×10^{-1}	5.03×10^{-2}	1.43×10^{-7}	0.087	0
F16	Mean	1.03×10^{2}	1.21×10^{2}	2.50×10^{1}	1.18×10^{2}	1.23×10^{1}	4.90×10^{-7}	0.180	0
	SD	2.47×10^{1}	3.93×10^{1}	6.95	3.39×10^{1}	5.41	1.50×10^{-6}	0.097	0

The AEABC consistently achieves a mean value of 0 for both F1 and F16, meaning it optimally solves these benchmark functions without any error, which none of the other algorithms achieve. The standard deviation of 0 for both functions indicates that AEABC's performance is perfectly consistent across all runs, a significant advantage over the different algorithms that exhibit varying degrees of variability. Among the other algorithms, SSA shows the closest performance to AEABC in terms of low mean values and relatively low standard deviations. However, it still cannot match AEABC's perfect results. Algorithms like PSO, BA, and MVO show significantly worse performance in accuracy (higher mean values) and consistency (higher standard deviations). AEABC performs vastly superior to other metaheuristics on the F1 and F16 benchmark functions. Its ability to achieve perfect optimization with zero error and zero variability makes it the most reliable and effective algorithm among those compared.

6.4. Results of Large-Scale Benchmarks

In this experiment, AEABC is evaluated on the benchmarks shown in Table 11 and compared with five metaheuristics: GWO, PSO, GSA, DE, and FEP. The statistical results of these algorithms were found in the literature [10], while AEABC results were obtained by following the same run conditions, which are 30 independent runs, a maximum number of iterations of 500, and a population size equal to 30.

Table 11. AEABC on large-scale optimization problems.

F	Metric	GWO	PSO	GSA	DE	FEP	AEABC
F14	Mean	4.042	3.627	5.86	9.98×10^{-1}	1.22	9.98×10^{-1}
	SD	4.253	2.561	3.831	3.3×10^{-16}	5.6×10^{-1}	2.302×10^{-13}
F15	Mean	3.37×10^{-4}	5.77×10^{-4}	3.673×10^{-3}	4.5×10^{-14}	5.0×10^{-4}	5.793×10^{-4}
	SD	6.25×10^{-4}	2.22×10^{-4}	1.647×10^{-3}	3.3×10^{-4}	3.2×10^{-4}	7.948×10^{-5}
F16	Mean	−1.032	−1.032	−1.032	−1.032	−1.03	−1.032
	SD	−1.032	6.25×10^{-16}	4.88×10^{-16}	3.1×10^{-13}	4.9×10^{-7}	2.868×10^{-9}
F17	Mean	3.979×10^{-1}	3.979×10^{-1}	3.979×10^{-1}	3.979×10^{-1}	3.98×10^{-1}	3.979×10^{-1}
	SD	3.979×10^{-1}	0.0	0.0	9.9×10^{-9}	1.5×10^{-7}	1.453×10^{-11}
F18	Mean	3.0	3.0	3.0	3.0	3.02	3.0
	SD	3.0	1.33×10^{-15}	4.17×10^{-15}	2×10^{-15}	1.1×10^{-1}	4.134×10^{-4}

The AEABC achieves optimal or near-optimal mean values across all large-scale optimization functions, often matching the performance of other top algorithms like DE, GWO, PSO, and GSA. The standard deviation for AEABC is minimal across all functions, indicating that it reliably produces results close to the best-known solutions with slight variation across multiple runs. In functions like F14, AEABC shows impressive consistency, outperforming even the best alternatives. While DE slightly outperforms AEABC in one or

two cases (e.g., F15), it remains highly competitive and often superior to algorithms like PSO, GWO, and FEP, particularly regarding consistency and reliability. AEABC performs strongly on large-scale optimization problems, combining high accuracy with remarkable consistency. It is a robust and reliable algorithm that competes well with, and often surpasses, other state-of-the-art metaheuristics, making it a highly effective choice for complex optimization tasks.

6.5. Results of Small-Scale Benchmarks

In this experiment, AEABC is evaluated on the benchmarks shown in Table 12 and compared with five metaheuristics: GWO, PSO, gravitational search algorithm (GSA), differential evolution (DE), and Fast Evolutionary Programing (FEP). The statistical results of these algorithms were found in the literature [10], while AEABC results were obtained by following the same run conditions in Section 6.4.

Table 12. AEABC on small-scale optimization problems.

F	Metric	GWO	PSO	GSA	DE	FEP	AEABC
F19	Mean	−3.863	−3.863	−3.863	N/A	−3.86	−3.862
	SD	−3.863	2.58×10^{-15}	2.29×10^{-15}	N/A	1.4×10^{-5}	4.262×10^{-4}
F20	Mean	−3.287	−3.266	−3.318	N/A	−3.27	−3.318
	SD	−3.251	6.052×10^{-2}	2.308×10^{-2}	N/A	5.9×10^{-2}	5.025×10^{-3}
F21	Mean	-1.015×10^{1}	−6.865	−5.955	-1.015×10^{1}	−5.52	-1.007×10^{1}
	SD	−9.14	3.02	3.737	2.5×10^{-6}	1.59	3.132×10^{-1}
F22	Mean	-1.04×10^{1}	−8.457	−9.684	-1.04×10^{1}	−5.53	-1.025×10^{1}
	SD	−8.584	3.087	2.014	3.9×10^{-7}	2.12	6.643×10^{-1}
F23	Mean	-1.053×10^{1}	−9.953	-1.054×10^{1}	-1.054×10^{1}	−6.57	-1.021×10^{1}
	SD	−8.559	1.783	2.6×10^{-15}	1.9×10^{-7}	3.14	1.103

N/A: Not available.

AEABC consistently performs well across all functions, often achieving mean values close to the best-performing algorithms. While not the lowest in every case, its standard deviations are generally moderate, indicating that AEABC provides reliable and consistent results. Particularly in F20, AEABC matches the best-performing algorithm (GSA) in terms of mean value but consistently outperforms it. In some cases, particularly in functions F19 and F21, AEABC's mean values are slightly less optimal than GWO and DE, and its standard deviations are higher, suggesting room for improvement in achieving more consistent performance. While AEABC may not consistently achieve the absolute best performance, it remains highly competitive, outperforming algorithms like PSO, GSA, and FEP in most cases. Its overall performance across these small-scale optimization problems demonstrates its robustness and reliability as an optimization tool. In conclusion, AEABC is a strong contender in small-scale optimization problems, often producing results close to or better than many leading metaheuristics. Its performance is characterized by good accuracy and reasonable consistency, making it a valuable algorithm for solving complex optimization challenges.

6.6. AEABC on Multimodal Benchmarks

For comprehensive examination, the AEABC algorithm was compared with other optimization algorithms using their results from the literature [28] shown in Table 13. Using the same run condition, 30 independent runs, 1000 maximum number of iterations, and 30 population size, the AEABC was compared with six metaheuristics shown in Table 13.

Table 13. AEABC on multimodal benchmarks.

F	MFO		PSO		GSA		AEABC	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**Sd**
F8	-8.497×10^{3}	7.259×10^{2}	-3.571×10^{3}	4.308×10^{2}	-2.352×10^{3}	3.822×10^{2}	-5.527×10^{3}	4.323×10^{2}
F9	8.460×10^{1}	1.617×10^{1}	1.243×10^{2}	1.425×10^{1}	3.100×10^{1}	1.366×10^{1}	0.0	0.0
F10	1.260	7.296×10^{-1}	9.168	1.569	3.741	1.713×10^{-1}	8.882×10^{-16}	0.0
F11	1.908×10^{-2}	2.173×10^{-2}	1.242×10^{1}	4.166	4.868×10^{-1}	4.979×10^{-2}	0.0	0.0
F12	8.940×10^{-1}	8.813×10^{-1}	1.387×10^{1}	5.854	4.634×10^{-1}	1.376×10^{-1}	4.981×10^{-4}	9.375×10^{-5}
F13	1.158×10^{-1}	1.930×10^{-1}	1.181×10^{4}	3.070×10^{4}	7.617	1.225	2.245×10^{-1}	1.736×10^{-1}
	FPA		SMS		FA		GA	
F8	-8.087×10^{3}	1.553×10^{6}	-3.943×10^{3}	4.042×10^{6}	-3.662×10^{3}	2.142×10^{2}	-6.331×10^{3}	3.326×10^{2}
F9	9.269×10^{1}	1.422×10^{1}	1.528×10^{2}	1.855×10^{1}	2.149×10^{6}	1.722×10^{1}	2.368×10^{6}	1.903×10^{1}
F10	6.845	1.250	1.913×10^{1}	2.385×10^{-1}	1.457×10^{1}	4.675×10^{-1}	1.785×10^{1}	5.311×10^{-1}
F11	2.716	7.277×10^{-1}	4.205×10^{2}	2.526×10^{1}	6.966×10^{1}	1.211×10^{1}	1.799×10^{2}	3.244×10^{1}
F12	4.105	1.043	8.743×10^{6}	1.406×10^{6}	3,684,008	1.721×10^{5}	3.413×10^{7}	1.893×10^{6}
F13	6.240×10^{1}	9.484×10^{1}	1.0×10^{8}	0.0	5.558×10^{6}	1.690×10^{6}	1.080×10^{8}	3.850×10^{6}

AEABC consistently achieves the best or near-best average values, especially on functions F9, F10, F11, and F12, significantly outperforming all other algorithms. This demonstrates AEABC's superior ability to optimize complex multimodal functions. The AEABC often achieves a standard deviation of zero, indicating perfect consistency, particularly on F9, F10, and F11. This reliability is unmatched by other algorithms, which show varying degrees of performance. Even in cases where AEABC does not have the best average (e.g., F8 and F13), it remains highly competitive and offers strong performance with relatively low variability. The AEABC demonstrates excellent overall performance on multimodal benchmark functions, frequently outperforming other state-of-the-art metaheuristics. Its ability to achieve near-perfect optimization with minimal variability makes it a highly reliable and effective algorithm for complex optimization tasks.

6.7. Small-Scale Optimization

This section presents the evaluation of the AEABC on small-scale optimization problems expressed in Table 5, where all of them have variable size D = 2. This experiment was achieved by 30 independent runs and 25,000 function evaluations, compared with six optimization algorithms. The statistical results of the competitors in Table 14 were adopted from the literature [14], while the results of AEABC were obtained using the abovementioned run conditions.

Table 14. Statistical results of AEABC and other metaheuristics on two-dimensional benchmarks.

Function		PSO	BA	MVO	KH	DDAO	AEABC
F19	Best	4.4298×10^{-14}	1.6438	1.0021×10^{-5}	2.8890×10^{-8}	9.64105×10^{-7}	0
	SD	2.7559×10^{-10}	1.1194×10^{2}	3.7081×10^{-4}	2.9457×10^{-7}	0.018695917	0
F20	Best	1.3482×10^{-14}	3.0619×10^{-11}	4.2336×10^{-9}	5.9289×10^{-12}	4.4529×10^{-5}	9.431×10^{-7}
	SD	2.3629×10^{-10}	5.4689×10^{-10}	6.5125×10^{-7}	3.1189×10^{-10}	0.01952182	5.491×10^{-5}
F21	Best	8.6209×10^{-17}	1.4036×10^{-12}	2.0125×10^{-10}	2.1689×10^{-14}	5.66862×10^{-12}	0.0
	SD	1.5676×10^{-12}	4.8089×10^{-11}	1.1549×10^{-8}	1.2854×10^{-11}	9.8777×10^{-7}	0.0
F22	Best	-1	-1	-1	-1	-0.99843	-1.0
	SD	2.8316×10^{-11}	1.8257×10^{-1}	1.8257×10^{-1}	1.8257×10^{-1}	-0.80388	4.139×10^{-5}
F23	Best	1.3624×10^{-14}	2.8649×10^{-11}	9.2179×10^{-9}	4.4607×10^{-13}	9.06×10^{-5}	7.950×10^{-9}
	SD	1.6315×10^{-1}	3.1004×10^{-1}	1.9334×10^{-1}	3.0481×10^{-10}	0.003105	1.327×10^{-5}
F24	Best	0	9.7159×10^{-3}	2.1517×10^{-6}	1.2317×10^{-7}	1.7525×10^{-9}	0.0
	SD	4.7621×10^{-3}	1.0699×10^{-1}	3.3529×10^{-3}	4.1334×10^{-6}	1.6223×10^{-6}	0.0

Table 14. *Cont.*

Function		PSO	BA	MVO	KH	DDAO	AEABC
F25	Best	−1.03163	−1.03163	−1.03163	−1.03163	−1.03157	−1.032
	SD	8.2049×10^{-11}	2.4904×10^{-1}	1.6383×10^{-7}	6.3582×10^{-10}	−1.03066	2.120×10^{-8}

AEABC consistently achieves optimal or near-optimal best values across all functions in this experiment. In many cases (F19, F21, F24, and F25), it reaches the perfect solution (zero error) with no variability. Perfect Consistency: For several functions (F19, F21, and F24), AEABC demonstrates zero standard deviation, consistently finding the optimal solution in every run, outperforming all other reliable algorithms. Even in cases where AEABC does not achieve a perfect solution (e.g., F20 and F23), it still demonstrates strong performance, with low mean errors and relatively low standard deviations compared to other algorithms. In conclusion, AEABC performs exceptionally on small-scale, two-variable test functions, often outperforming other state-of-the-art metaheuristics in accuracy and consistency. This makes AEABC a highly reliable and effective algorithm for solving small-scale optimization problems.

7. Statistical Analysis

The Wilcoxon rank-sum test was used to evaluate the statistical significance of the performance differences between the adaptive exploration artificial bee colony algorithm and other metaheuristic algorithms in Tables 7 and 8 across multiple benchmark functions (F1 to F13). The purpose of this statistical analysis is to assess whether the observed performance differences are statistically significant or due to random variations. The results are summarized in three parts: Table 15 (for test functions F1–F5), Table 16 (for test functions F6–F10), and Table 17 (for test functions F10–F13). In each case, the Wilcoxon statistics and *p*-value are reported, which indicate the direction and significance of the performance differences.

Table 15. Wilcoxon test results for F1–F5.

Algorithm	Metric	F1	F2	F3	F4	F5
ALO	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}
BOA	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.638×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	3.175×10^{-11}
GWO	Statistic	-6.653×10^{0}	-6.653×10^{0}	-5.130×10^{0}	-6.653×10^{0}	5.736×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.894×10^{-7}	2.872×10^{-11}	9.673×10^{-9}
PSO	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-2.218×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.658×10^{-2}
SCA	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.017×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	1.774×10^{-9}
WOA	Statistic	6.653×10^{0}	6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}
DDAO	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}
ABC	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}

Table 16. Wilcoxon test results for F6–F10.

Algorithm	Metric	F6	F7	F8	F9	F10
ALO	Statistic	-6.653×10^{0}	-6.653×10^{0}	5.396315278	−6.652991439	−6.652991439
	p-value	2.872×10^{-11}	2.872×10^{-11}	6.80234×10^{-8}	2.87195×10^{-11}	2.87195×10^{-11}
BOA	Statistic	-6.653×10^{0}	-6.653×10^{0}	−6.652991439	0	−6.652991439
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.87195×10^{-11}	1	2.87195×10^{-11}
GWO	Statistic	1.493×10^{0}	-6.579×10^{0}	6.224243101	−4.435327626	−6.652991439
	p-value	1.354×10^{-1}	4.734×10^{-11}	4.83886×10^{-10}	9.19324×10^{-6}	2.87195×10^{-11}
PSO	Statistic	6.653×10^{0}	-6.653×10^{0}	6.224243101	−6.652991439	−6.652991439
	p-value	2.872×10^{-11}	2.872×10^{-11}	4.83886×10^{-10}	2.87195×10^{-11}	2.87195×10^{-11}
SCA	Statistic	-6.653×10^{0}	-6.653×10^{0}	−6.120752124	−6.652991439	−6.652991439
	p-value	2.872×10^{-11}	2.872×10^{-11}	9.31347×10^{-10}	2.87195×10^{-11}	2.87195×10^{-11}
WOA	Statistic	6.653×10^{0}	-5.219×10^{0}	6.652991439	0	−4.878860388
	p-value	2.872×10^{-11}	1.800×10^{-7}	2.87195×10^{-11}	1	1.06701×10^{-6}
DDAO	Statistic	-6.653×10^{0}	-6.653×10^{0}	−5.677219361	−6.652991439	−6.652991439
	p-value	2.872×10^{-11}	2.872×10^{-11}	1.36902×10^{-8}	2.87195×10^{-11}	2.87195×10^{-11}
ABC	Statistic	-6.653×10^{0}	-6.653×10^{0}	1.907190879	−6.652991439	−6.652991439
	p-value	2.872×10^{-11}	2.872×10^{-11}	0.056495874	2.87195×10^{-11}	2.87195×10^{-11}

Table 17. Wilcoxon test results for F11–F13.

Algorithm	Metric	F11	F12	F13
ALO	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}
BOA	Statistic	0.000×10^{0}	-6.653×10^{0}	-2.927×10^{0}
	p-value	1.000×10^{0}	2.872×10^{-11}	3.419×10^{-3}
GWO	Statistic	-1.109×10^{0}	6.195×10^{0}	6.653×10^{0}
	p-value	2.675×10^{-1}	5.841×10^{-10}	2.872×10^{-11}
PSO	Statistic	-6.653×10^{0}	6.653×10^{0}	6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}
SCA	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.638×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	3.175×10^{-11}
WOA	Statistic	-4.435×10^{-1}	6.653×10^{0}	6.653×10^{0}
	p-value	6.574×10^{-1}	2.872×10^{-11}	2.872×10^{-11}
DDAO	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}
ABC	Statistic	-6.653×10^{0}	-6.653×10^{0}	-6.653×10^{0}
	p-value	2.872×10^{-11}	2.872×10^{-11}	2.872×10^{-11}

Across functions F1 to F5, AEABC outperforms most algorithms with highly significant *p*-values (2.872×10^{-11}) in all cases, indicating a robust difference in performance in favor of AEABC. Overall, the Wilcoxon rank-sum test results highlight that AEABC demonstrates superior performance compared to many of the competing metaheuristic algorithms across the majority of benchmark functions. However, specific cases such as GWO and WOA

on functions F5, F8, F9, and F13 suggest that these algorithms may perform better on certain types of optimization problems. These findings provide valuable insights into AEABC's strengths and weaknesses, offering direction for future improvements to enhance its robustness across a broader range of functions.

8. Practical Application

For extensive examination, this section employs statistical results from the literature [10] for the welded beam design problem. This work followed the same run condition mentioned in the original work to have a fair comparison and to give the reader a broader insight into the proposed algorithm. This engineering problem is represented in Figure 1, which has four variables, and the objective is reducing the fabrication cost. The problem is well expressed in [10]. The problem can be described as follows:

$$\text{Consider } \vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} = \begin{bmatrix} h & l & t & b \end{bmatrix}, \tag{8}$$

$$\text{Minimize } f\left(\vec{x}\right) = 1.10471\, x_1^2 x_2 + 0.04811\, x_3 x_4 (14.0 + x_2),$$

$$\text{subject to } g_1\left(\vec{x}\right) = \tau\left(\vec{x}\right) - \tau_{\max} \le 0,$$

$$g_2\left(\vec{x}\right) = \sigma\left(\vec{x}\right) - \sigma_{\max} \le 0,$$

$$g_3\left(\vec{x}\right) = \delta\left(\vec{x}\right) - \delta_{\max} \le 0,$$

$$g_4\left(\vec{x}\right) = x_1 - x_4 \le 0,$$

$$g_5\left(\vec{x}\right) = P - P_C\left(\vec{x}\right) \le 0,$$

$$g_6\left(\vec{x}\right) = 0.125 - x_1 \le 0,$$

$$g_7\left(\vec{x}\right) = 0.10471\, x_1^2 + 0.04811\, x_3 x_4 (14.0 + x_2) - 0.5,$$

where

$$\tau\left(\vec{x}\right)\sqrt{\tau'^2 + 2\tau'\tau''\frac{x_2}{2R} + \tau''^2},\ \tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},\ M = P\left(L + \frac{x_2}{2}\right),\ R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{9}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},\ \sigma\left(\vec{x}\right) = \frac{6PL}{x_4x_3^2},\ \delta\left(\vec{x}\right) = \frac{6PL^3}{Ex_3^2x_4},\ P_C\left(\vec{x}\right) = \frac{4.013\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000lb, L = 14in., \delta_{\max} = 0.25in., E = 30 * 10^6\ psi., G = 12 * 10^6\ psi., \tau_{\max} = 13600\ psi., \sigma_{\max} = 30000\ psi.,$$

$$0.1 \le x_1 \le 2$$

$$0.1 \le x_2 \le 10$$

$$0.1 \le x_3 \le 10$$

$$0.1 \le x_4 \le 2$$

Table 18 reveals the performance of the AEABC and the other ten algorithms in this study [10]. The adopted algorithms for this reference study are gray wolf optimizer GWO, gravitational search algorithm GSA [22], genetic algorithm GA [38,39], harmony search HS [40], random (Richardson's random method), Simplex method, Davidon–Fletcher–

Powell, APPROX [41]. The results of 30 independent runs and 500 iterations reveal that AEABC has a lower fitness value than other metaheuristics.

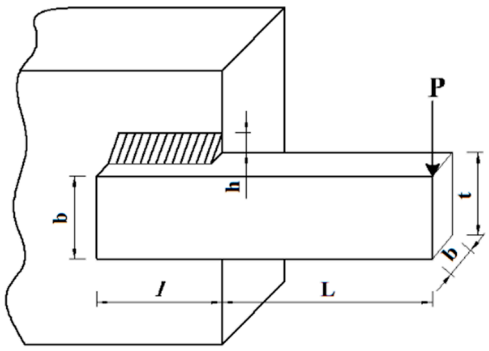

Figure 1. The welded beam design problem.

Table 18. Comparing the AEABC optimum solution with other methods from [10] on welded beam design.

Algorithm	Optimum Variables				Optimum Cost
	h	l	t	b	
AEABC	**0.1071**	**5.4402**	**8.8660**	**0.1**	**0.8981**
GWO	0.2056	3.4783	9.0368	0.2057	1.7262
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815
GSA	0.1821	3.8569	10.000	0.2023	1.8799
HS	0.2442	6.2231	8.2915	0.2443	2.3807
GA [38]	N/A	N/A	N/A	N/A	1.8245
Random	0.4575	4.7313	5.0853	0.66	4.1185
GA [42]	N/A	N/A	N/A	N/A	2.38
Simplex	0.2792	5.6256	7.7512	0.2796	2.5307
GA [39]	0.2489	6.1730	8.1789	0.2533	2.4331
David	0.2434	6.2552	8.2915	0.2444	2.3841

N/A: not available.

Also, AEABC's performance was compared with the results of CGWO and others [32], and it had the best performance, as shown in Table 19.

Table 19. Comparing the performance of AEABC with other algorithms adopted by [43].

Algorithm	Worst	Mean	Best	SD
AEABC	**1.1505**	**1.0394**	**0.8981**	**0.0642**
GWO	2.9136	2.8594	1.9421	2.6908
CPSO	1.7821	1.7488	1.7280	1.29×10^{-2}
CDe	N/A	1.7681	1.7334	N/A
GA4	1.9934	1.7926	1.7282	7.47×10^{-2}
CGWO	2.4357	2.4289	1.7254	1.3578
SC	6.3996	3.0025	2.3854	9.60×10^{-1}
UPSO	N/A	2.8372	1.9219	0.683
GA3	1.785835	1.7719	1.7483	1.12×10^{-2}

N/A: Not available.

AEABC demonstrates the best overall performance, achieving the lowest best cost and maintaining a low mean and worst cost, along with a very low standard deviation. This indicates that AEABC is the most cost-effective and reliable algorithm, consistently producing near-optimal solutions with minimal variation. The comparison underscores AEABC's superiority over other state-of-the-art algorithms in solving the welded beam design problem, making it an excellent choice for complex engineering optimization tasks.

9. Exploration and Exploitation in AEABC

The adaptive exploration artificial bee colony (AEABC) algorithm introduces a novel distance-based parameter to enhance the balance between exploration and exploitation, two crucial phases in optimization. Exploration involves searching for new regions of the solution space to avoid local optima, while exploitation refines existing solutions to achieve better convergence. In traditional ABC, the neighborhood search process does not consider the distance between solutions when generating new candidates. The AEABC algorithm addresses this limitation by introducing a distance-based probability, which governs the likelihood of shifting solutions based on their distance from each other. This distance is calculated using the Euclidean metric (Equation (5)), and the shift is determined (Equation (6)).

Exploitation in AEABC is enhanced by prioritizing solutions in close proximity. When the distance d between two solutions is small, the probability Pd approaches 0, leading to a higher chance of refining nearby solutions (Equation (7)). This increases the likelihood of local search intensification, helping the algorithm converge more effectively. Conversely, if d is large, the probability Pd is close to 1, and the current solution is less likely to shift, allowing broader exploration of the search space. Table 1 further explains how this probability mechanism functions, illustrating how long or short distances between solutions influence the likelihood of a solution shift. The results in Table 6 also demonstrate AEABC's enhanced exploitation capabilities. For example, in function F1 with a variable size of 100, applying the distance parameter to the onlooker bees leads to a substantial reduction in standard deviation (e.g., 8.31×10^{-115}). This indicates that the algorithm converges more consistently when prioritizing the exploitation phase, as onlooker bees focus on refining nearby food sources. The distance-based probability Pd enables AEABC to dynamically adjust its behavior, enhancing exploration during the early stages of optimization and shifting toward exploitation as the algorithm progresses. This adaptability is key to avoiding premature convergence while ensuring rapid convergence once high-quality solutions are identified.

10. Conclusions

The artificial bee colony (ABC) algorithm, renowned for its simplicity and efficiency, has been widely applied across various domains as a robust swarm intelligence approach. However, this article introduces a significant advancement to the original ABC through the development of the adaptive exploration artificial bee colony (AEABC) algorithm. Unlike mere hybridization or combination with other algorithms, AEABC reimagines the fundamental mechanisms of ABC by incorporating a crucial new parameter: the distance between food sources. This adjustment reflects a more realistic modeling of the foraging behavior observed in natural bee colonies, enhancing the algorithm's exploration and exploitation capabilities. The performance of AEABC was evaluated against 33 state-of-the-art metaheuristic algorithms across 25 benchmark functions, where it consistently demonstrated superior efficiency and accuracy. The results showed improvement over the traditional ABC algorithm, validating the effectiveness of this innovative approach. AEABC achieved remarkable consistency in producing optimal solutions, often outperforming its counterparts in mean performance, best and worst-case scenarios, and standard deviation, reflecting its reliability and robustness.

Furthermore, AEABC's application to a useful engineering problem underscored its practical utility, outperforming leading algorithms in producing cost-effective and reliable

solutions. This success highlights AEABC's potential as a powerful tool for solving real-world optimization problems. In conclusion, the AEABC algorithm represents a robust and straightforward approach to optimization. This work enhances the ABC algorithm and opens new avenues for future research, encouraging the exploration and reinvention of existing algorithms with innovative, nature-inspired ideas.

Author Contributions: Conceptualization, S.A. and H.A.; methodology, S.A.; software, H.A.; validation, S.A. and H.A.; formal analysis, S.A.; investigation, S.A.; resources, S.A.; data curation, H.A.; writing—original draft preparation, S.A.; writing—review and editing, R.R.; visualization, H.A.; supervision, E.K.; project administration, E.K.; funding acquisition, E.K. All authors have read and agreed to the published version of the manuscript.

Funding: This publication was financially supported by Széchenyi István University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ghafil, H.N.; Jármai, K. Optimum dynamic analysis of a robot arm using flower pollination algorithm. In *Advances and Trends in Engineering Sciences and Technologies III—Proceedings of the 3rd International Conference on Engineering Sciences and Technologies, ESaT 2018*; CRC Press: Boca Raton, FL, USA, 2019.
2. Karaboga, D. Artificial bee colony algorithm. *Scholarpedia* **2010**, *5*, 6915. [CrossRef]
3. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
4. Thammano, A.; Phu-ang, A. A hybrid artificial bee colony algorithm with local search for flexible job-shop scheduling problem. *Procedia Comput. Sci.* **2013**, *20*, 96–101. [CrossRef]
5. Choong, S.S.; Wong, L.-P.; Lim, C.P. An artificial bee colony algorithm with a modified choice function for the traveling salesman problem. *Swarm Evol. Comput.* **2019**, *44*, 622–635. [CrossRef]
6. Alsamia, S.; Albedran, H.; Mahmood, M.S. Contamination depth prediction in sandy soils using fuzzy rule-based expert system. *Int. Rev. Appl. Sci. Eng.* **2023**, *14*, 87–99. [CrossRef]
7. Albedran, H.; Jármai, K. Evolutionary control system of asymmetric quadcopter. *Int. Rev. Appl. Sci. Eng.* **2023**, *14*, 374–382. [CrossRef]
8. Ghafil, H.N.; László, K.; Jármai, K. Investigating three learning algorithms of a neural networks during inverse kinematics of robots. In *Solutions for Sustainable Development*; CRC Press: Boca Raton, FL, USA, 2019; pp. 33–40.
9. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
10. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
11. Alsamia, S.; Albedran, H.; Jármai, K. Comparative Study of Different Metaheuristics on CEC 2020 Benchmarks. In *Vehicle and Automotive Engineering 4: Select Proceedings of the 4th VAE2022, Miskolc, Hungary*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 709–719.
12. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
13. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
14. Ghafil, H.N.; Jármai, K. Dynamic differential annealed optimization: New metaheuristic optimization algorithm for engineering applications. *Appl. Soft Comput.* **2020**, *93*, 106392. [CrossRef]
15. Alsamia, S.M.; Mahmood, M.S.; Akhtarpour, A. Prediction of the contamination track in Al-Najaf city soil using numerical modelling. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *888*, 12050. [CrossRef]
16. Yang, X.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
17. Yang, X.-S.; He, X. Firefly algorithm: Recent advances and applications. *Int. J. Swarm Intell.* **2013**, *1*, 36. [CrossRef]
18. Alsamia, S.; Koch, E. Evaluation the behavior of pullout force and displacement for a single pile: Experimental validation with plaxis 3D. *Kufa J. Eng.* **2023**, *14*, 105–116. [CrossRef]
19. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
20. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
21. Jain, M.; Singh, V.; Rani, A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm Evol. Comput.* **2019**, *44*, 148–175. [CrossRef]
22. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]

23. Kaveh, A.; Farhoudi, N. A new optimization method: Dolphin echolocation. *Adv. Eng. Softw.* **2013**, *59*, 53–70. [CrossRef]
24. Alsamia, S.; Koch, E. Random forest regression on pullout resistance of a pile. *Pollack Period.* **2024**, *19*, 28–33. [CrossRef]
25. Yang, X.-S. Flower pollination algorithm for global optimization. In *International Conference on Unconventional Computing and Natural Computation*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 240–249.
26. Yao, X.; Liu, Y. Fast Evolutionary Programming. *Evol. Program.* **1996**, *3*, 451–460.
27. Cuevas, E.; Echavarría, A.; Ramírez-Ortegón, M.A. An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. *Appl. Intell.* **2014**, *40*, 256–272. [CrossRef]
28. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
29. Mirjalili, S.; Mirjalili, S. Genetic algorithm. *Evol. Algorithms Neural Netw. Theory Appl.* **2019**, *780*, 43–55.
30. Alsamia, S.; Koch, E.; Hamadi, H.S. Comparative study of metaheuristics on optimal design of gravity retaining wall. *Pollack Period.* **2023**, *18*, 35–40. [CrossRef]
31. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-tr06; Erciyes University, Engineering Faculty, Computer Engineering Department: Kayseri, Türkiye, 2005.
32. Tereshko, V.; Loengarov, A. Collective decision making in honey-bee foraging dynamics. *Comput. Inf. Syst.* **2005**, *9*, 1.
33. Georgia Tech College of Computing "youtub, 2011" The Waggle Dance of the Honeybee. Available online: https://www.youtube.com/watch?v=bFDGPgXtK-U&t=312s (accessed on 26 October 2024).
34. Goldberg, D.E.; Holland, J.H. Genetic algorithms and machine learning. *Mach. Learn.* **1988**, *3*, 95–99. [CrossRef]
35. Heris, M. Artificial Bee Colony. 2015. Available online: https://www.mathworks.com/matlabcentral/fileexchange/52966-artificial-bee-colony-abc-in-matlab (accessed on 26 October 2024).
36. MathWorks. Available online: https://www.mathworks.com/matlabcentral/ (accessed on 26 October 2024).
37. Yarpiz.Optimization Algorithms. Available online: https://yarpiz.com/ (accessed on 26 October 2024).
38. Coello, C.A.C. Constraint-handling using an evolutionary multiobjective optimization technique. *Civ. Eng. Syst.* **2000**, *17*, 319–346. [CrossRef]
39. Deb, K. Optimal design of a welded beam via genetic algorithms. *AIAA J.* **1991**, *29*, 2013–2015. [CrossRef]
40. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]
41. Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [CrossRef]
42. Ragsdell, K.M.; Phillips, D.T. Optimal design of a class of welded structures using geometric programming. *J. Eng. Ind.* **1976**, *98*, 1021–1025. [CrossRef]
43. Kohli, M.; Arora, S. Chaotic grey wolf optimization algorithm for constrained optimization problems. *J. Comput. Des. Eng.* **2018**, *5*, 458–472. [CrossRef]