Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Advance Bat Algorithm Inspired Algorithm for Fast Convergence and High Accuracy in Solving Numerical Optimisation Problems

M. R. Ramli

raziff@uum.edu.my

Universiti Utara Malaysia

Z. A. Abas

Technical University of Malaysia Malacca

F. K. Ahmad

Universiti Utara Malaysia

K. M. Zaini

Universiti Utara Malaysia

Article

Keywords: Metaheuristic Optimization, Swarm Intelligence, Predictive Maintenance, Convergence Accuracy

Posted Date: March 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9045548/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Advance Bat Algorithm Inspired Algorithm for Fast Convergence and High Accuracy in Solving Numerical Optimisation Problems

M. R. Ramli[1, *], **Z. A. Abas**[2], **F. K. Ahmad**[3], **K. M. Zaini**[4]
[1,3,4] Universiti Utara Malaysia,
School of Computing, Malaysia
[2]Universiti Teknikal Malaysia Melaka,
Malaysia.
Faculty of Information Technology and
Communication, Malaysia
[*]raziff@uum.edu.my
[+]these authors contributed equally to this work

ABSTRACT

This paper presents the Advanced Bat Algorithm (ABA), a novel enhancement to the standard Bat Algorithm (BA) designed to address persistent challenges in numerical optimisation, notably slow convergence and susceptibility to local minima in high-dimensional search spaces. ABA introduces three key innovations: physics-based inertia weight modeling, a boundary-aware position update using sine functions, and adaptive boundary control. These modifications collectively improve the balance between exploration and exploitation, ensuring robust performance across a variety of benchmark functions. Extensive computational experiments demonstrate that ABA achieves faster convergence and higher accuracy than standard BA and leading population-based algorithms such as Particle Swarm Optimisation (PSO), Genetic Algorithm (GA), and Harmony Search (HS). Comparative analysis with recent BA variants, including Improved Bat Algorithm (IBA) and Hybrid Self-Adaptive Bat Algorithm (HSABA), further confirms ABA's superior performance, especially in high-dimensional and complex optimisation scenarios. The findings suggest that ABA is a promising tool for real-world optimisation tasks in 2026, with potential applications in AI model tuning, predictive maintenance, and smart manufacturing.

Keywords Metaheuristic Optimization, Swarm Intelligence, Predictive Maintenance, Convergence Accuracy.

Introduction

The optimisation process is an important social activity for human beings even in industrial engineering, finance, sport, healthcare, crime, and so forth. The raise optimisation problem from human activities can be interpreted into complex nonlinear constraints and known as optimisation model. Development of optimisation model give the opportunity to solve optimisation problem in term of minimum the cost and material use. Hence, increase performance and lifetime services. Since today, the variance of optimisation algorithm m has been published. However, it is important to match the suitable optimisation algorithm to the specific optimisation problem[1], [2]. Severe optimisation issue in particular real-world engineering problems [3], [4] had been solved by metaheuristic algorithm included evolutionary and swarm intelligence algorithms [5], [6] in which most of the metaheuristic algorithm inspired by nature living things that involve biological and/or physical ecosystem behaviour.

The current advanced and popular new optimisation algorithm is modern metaheuristic optimisation. Most of the optimisation algorithms are inspired by nature [5], [7], from simulated annealing [8] to ant colony optimisation [7], and from particle swarm optimization [6] to bat algorithm[9], [10]. Almost all area optimisation, scheduling and planning, data mining, machine intelligence and many other problems have been solved by metaheuristic algorithms.

The various optimisation problems can be successfully solved by a metaheuristic if there is a balance between the exploration of the search space and the exploitation of the accumulated search experience to identify the best solution to solve the problem in a near-optimal way. The way of achieving an optimal solution is the main difference between the existing metaheuristics. The metaheuristic method can be differentiated by their behaviour on how the algorithm plays its rule to search the solution or how memory is exploited.

S. Bandaru and K. Deb had described that the classifications of the metaheuristic are made according to the deterministic versus stochastic methods, greedy versus non-greedy methods, memory usage versus memoryless methods, one versus various neighborhood methods, dynamic versus static objective function and least nature-inspired versus non-nature inspired method [11]. The classification of metaheuristic techniques has been illustrated in Figure 1. The components in the blue-coloured boxes represent the work related to this paper.

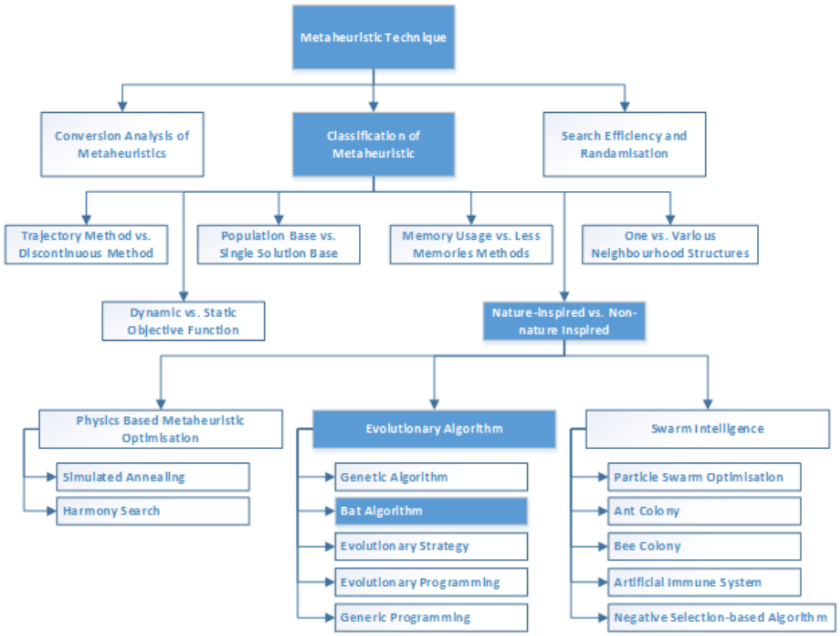


Figure 1. Classification of metaheuristic technique

One of the metaheuristic algorithm is Bat Algorithm (BA). The Bat Algorithm (BA), introduced by Xin-She Yang in 2010, is inspired by the echolocation behaviour of microbats. BA has been widely adopted for its simplicity, adaptability, and ability to balance exploration and exploitation in the search for optimal solutions. However, despite its strengths, standard BA exhibits notable limitations: it can become trapped in local minima, especially in high-dimensional or multimodal landscapes, and may converge slowly when the search space is vast or the objective function is complex.
To address these challenges, this research introduces the Advanced Bat Algorithm (ABA), which incorporates three major enhancements. First, Physics-based inertia weight modelling to better control the balance between exploration and exploitation. Second, Boundary-aware position update using sine functions to ensure feasible and diverse solution generation within the search space. Third, Adaptive boundary control to dynamically focus the search and prevent premature convergence.
This paper systematically details the design, implementation, and evaluation of ABA, situating it within the broader context of recent advances in BA variants and metaheuristic optimisation. The results demonstrate that ABA not only outperforms standard BA but also achieves superior results compared to leading alternatives, making it highly relevant for contemporary and future optimisation challenges.

Related Works

Since its inception, the Bat Algorithm (BA) has inspired a prolific body of research, leading to numerous variants and hybridisations designed to overcome its inherent limitations. Over the past five years, several key developments have emerged. The Improved Bat Algorithm (IBA) enhances population initialization and modifies update rules for frequency, velocity, and solution, while incorporating adaptive loudness and pulse emission rates, thereby improving convergence

and robustness in high-dimensional problems [12], [13]. The Hybrid Self-Adaptive Bat Algorithm (HSABA) integrates self-adaptation of control parameters and combines BA with differential evolution strategies for local search, resulting in stronger global search capability and reduced susceptibility to local optima [14]. The Binary Bat Algorithm (BBA) extends BA to discrete and combinatorial optimisation, particularly in feature selection and classification, with enhanced versions employing chaos, Lévy flights, and elitist strategies to improve diversity and search efficiency[15], [16]. The Modified Bat Algorithm (MBA) leverages the frequency and velocity of the current best solution to accelerate convergence and avoid entrapment in local optima, demonstrating superior performance compared to standard BA, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and the Dragonfly Algorithm (DA) in both benchmark tests and real-world applications such as call centre optimisation [13].

In addition, multi-objective and hybrid variants (e.g., MOBA) have been developed to address multi-objective optimisation and exploit the complementary strengths of other metaheuristics. For example, Differential Evolution (DE) contributes strong mutation and crossover mechanisms that enhance exploration [17], Harmony Search (HS) provides improvisation-based strategies that balance diversification and intensification [18], and the Artificial Bee Colony (ABC) algorithm introduces swarm intelligence principles inspired by bee foraging behaviour to improve solution diversity and convergence [19].

Recent systematic reviews and meta-analyses confirm that BA and its variants remain highly competitive, particularly when enhanced with adaptive mechanisms, hybrid local search, and dynamic parameter control [10], [17]. Their versatility has enabled successful applications across diverse domains, including engineering design (structural optimisation, power system dispatch, scheduling), machine learning (feature selection, classifier parameter tuning, clustering), and industrial contexts (production scheduling, supply chain management, predictive maintenance, smart manufacturing). Real-world case studies, such as call center optimisation, vehicle routing, and resource allocation, further underscore the adaptability of BA to both continuous and discrete domains, as well as its ease of hybridisation, which has contributed to its sustained relevance in academic research and industrial practice [20], [21].

The Bat Algorithm (BA) and its variants have been successfully applied across a wide spectrum of optimisation problems in diverse domains. In engineering design, BA has been utilised for structural optimisation, power system dispatch, and scheduling tasks, demonstrating its ability to handle complex technical constraints [20]. Within machine learning, BA has proven effective in feature selection, parameter tuning for classifiers such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN), as well as clustering, thereby enhancing model accuracy and efficiency [2], [22]. In industrial applications, BA has been employed for production scheduling, supply chain management, predictive maintenance, and smart manufacturing, offering practical solutions to operational challenges [23]. Furthermore, real-world case studies highlight its utility in call center optimisation, vehicle routing, and resource allocation, underscoring its adaptability to practical scenarios [4], [20]. The algorithm's capacity to operate in both continuous and discrete domains, coupled with its ease of hybridisation with other metaheuristics, has contributed to its sustained relevance and competitiveness in both academic research and industrial practice [20].

Bat Algorithm Fundamental

Echolocation of Microbats

Bats can be categorised into 1000 species. Bat have a variance of body size form small size (microbats) at range 1.5 to 2 grams and giant species (megabats) can have body size with wings about 2 m and weigh up to 1 kg. Most of the microbats use echolocation but not for megabats. Microbats are capable to produce a very loud sound pulse and use the reflected sound from the surrounding object for listening [9]. Meanwhile, the pulse rate can be varied depending on microbats species and their hunting strategy.

Most bats use modulated and low frequencies to sweep about one octave. Each pulse will last several thousand of a second (up to about 8 to 10 ms) in the frequency, f is typically at range 25kHz to 150 kHz. Usually, microbats can produce sound bursts for every second about 10 to 20 times, and when the bat hurry on their prey the emission of pulse rate can be raced up to about 200 pulses per second. While the speed of sound in the air is about v = 340 m/s. Thus, the wavelength of the ultrasonic sound bursts with a constant frequency f can be defined by $\lambda = v/f$. As a remark, their prey sizes are the same order with these wavelengths. In addition, microbats doing time delay between loudness vibration and their ears to sense three-dimensional surroundings. The ability of echolocation give the

opportunities for BA to be improved. Xin-She Yang (2010) has introduced three idealised rule as guidance to construct the improvement of BA.

Three rule was stated as follow:

1. Bats have the ability to sense the distance of an obstacle and can distinguish between food/prey using echolocation. Even in the dark.
2. Bat does searching for prey by fly randomly with the velocity v_i at the position x_i. The wavelength λ and loudness *A* are vary automatically through their pulse emission *r*. The pulse emission value is at range 0 to 1 and the adjustment is depending on their target istance.

Loudness can vary in any way because the loudness varies from a large (positive) A_{max} to a minimum constant value A_{min}.

Bat Motion

At the beginning of the simulation, most of the metaheuristic algorithms go through randomisation processes. The same goes for bats behaviour that moves randomly to an unknown location for searching their prey. The random search is limited to the size of the boundary search. The random process is a random vector that effect by the number of dimensions that denoted as D.

Each generated random value, the flying bats will move to new positions and will produce a new solution. Each moving bat makes their frequency and velocity value are updated. The process produces a new solution is presented in the following equation.

$$f_i = f_{\min} + \beta(f_{\max} - f_{\min}) \quad (1)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*) f_i \quad (2)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (3)$$

Based on the equation, the frequency value is denoted as f_i. The frequency depends on minimum frequency denoted as f_{min} and maximum frequency denoted as f_{max}. The symbol β denoted as a random number. The value of velocity v_i^t is denoted as . The simulation repeat at constant time *t* and constant number of bat *i*. The new bat position as a new solution is denoted as x_i. The generated new solution for each iteration time *t* will be sorted and compared to each other to select the best solution. The best solution will be represented as global best denoted as global best denoted as x_*.

Variations of Loudness and Pulse Rates

The variation of loudness (*A*) and pulse emission rate (*r*) happened went the bat becomes closer to the target (i.e. prey). These phenomena cause the value of loudness will be decreased and the value of pulse emission will be increased along with the time step *t*. The equation of this phenomena is presented as follow:

$$A_i^{t+1} = \alpha A_i^t \quad (4)$$

$$r_i^{t+1} = r_i^0 (1 - e^{\gamma t}) \quad (5)$$

Challenges in Optimisation Algorithm

It is not easy to solve optimisation problems either single-objective or multi-objective at high dimensions. To solve real-world optimization problem is very complex. Even the optimisation algorithm can't solve the problem at a given time. This combination of problems is

very complicated and impractical. Despite its versatility, the standard Bat Algorithm (BA) continues to face several critical challenges. A major limitation is premature convergence, whereby the algorithm often becomes trapped in local minima, particularly within multimodal or high-dimensional search landscapes. This is compounded by slow convergence rates, which reduce efficiency when navigating large or complex objective functions. Moreover, BA exhibits pronounced parameter sensitivity, as its performance is highly dependent on the careful calibration of parameters such as loudness, pulse rate, and frequency range. Equally significant is the algorithm's limited theoretical foundation, with insufficient rigorous convergence analysis and a lack of generalizable guidelines for parameter tuning. To address these shortcomings, recent research has increasingly emphasized adaptive parameter control, hybridisation with complementary metaheuristics, and the integration of physics-based and mathematical modelling techniques, all of which aim to enhance robustness and scalability in diverse optimization contexts.

Propose Design and Prototype

The design and prototype construction to improve the existing optimisation algorithm are guided from the analysis of the literature review. In the real-world, the optimisation problem is very complex. Even the decision variable increase, then there is not an optimal choice. This lets the exponentially increase the number of evaluations of the recursive function. Although the standard BA has exceptional properties compared to some traditional optimisation methods, the phenomena if low accuracy and slow convergence rate still exist. Therefore, the Advanced Bat Algorithm (ABA) builds upon the standard BA by introducing three principal enhancements: physics-based inertia weight modelling, boundary-aware position update using sine functions, and adaptive boundary control to improve the convergence rate and increase the accuracy to the optimum solution. Figure 1 shows the three propose additional steps to improve the standard BA.

Physics-Based Inertia Weight Modelling

Bat Algorithm is nature-inspired algorithms that consider the influence of surrounding conditions such as inertia. Like others metaheuristic, inertia is denoted as w. It's used to control the exploration and exploitation of the solution. The issue that has been raised in standard BA is during the iteration proceed, the tendency to explore the solution has gradually decreased. With the addition of the inertia weight factor will reduce this issue [10]. This approach has been practiced by previous researchers. The result has been established that the implementation of inertia weight will increase the performance of BA as an optimisation technique [24], [25], [26], [27], [28], [29].

However, it must be noted that the previous inertia weight factor in existing research do not apply the real concept of physic representative inertia behaviour. Inertia is the resistance, of any physical object with mass, to any change in its velocity. This includes changes to the object's speed, or direction of motion. Thus, an object will continue moving at its current velocity until some force causes its speed or direction to change. Therefore, in this research, the effect of inertia weight to the velocity equation in the real physics concept has been one of the enhancements. This in turn affects the whole process in existing BA.

The changes in movement or position of an object will produce the inertia factor. The phenomena of inertia are related to object mass and its velocity of object movement. Based on Newton's first law of motion, an object continues to move with constant velocity without the net of external force unless some force causes the speed or changing direction of an object. The same situation when the immovable object stays at its position until the net of external force causes it to move.

Unlike heuristic or linearly decreasing inertia weights commonly used in swarm intelligence, ABA models inertia as a function of the bat's mass and the square of the distance between its current and new positions, expressed as *rotational inertia (w)* = $m \cdot r^2$. Here, *m* represents the bat's mass (randomly assigned within a biologically plausible range, e.g., 0.0015–1 kg), and *r* denotes the Euclidean distance between the current best x_* and new positions x_t. This inertia weight is incorporated into the velocity update equation, ensuring that larger movements correspond to higher inertia (exploration) and smaller movements to lower inertia (exploitation), thereby maintaining a dynamic balance and reducing premature convergence [23], [30]. The calculation moment of inertia of bat was presented at Equation (6).

$$w = mr^2 \tag{6}$$

The procedure to implement the inertia phenomena inside standard BA is similar to procedure implementation of the inertia inside standard particle swarm optimisation (PSO). PSO consists of three parts velocity equation which is the

previous velocity, changes velocity affected from changes bat position, and changes velocity affected from previous velocity (current velocity). The previous velocity produces the effect of exploration. These phenomena give chances of the particle to explore in another area [10]. These phenomena give chances of the particle to explore in another area. In this process where the inertia will be implemented. The exploration and exploitation will be controlled by the phenomena of inertia. Equation (7) is a modified equation from standard BA at Equation (2).

$$v_i^t = v_i^{t-1} w_i + (x_i^t - x_*) f_i \quad (7)$$

It must be noted that the Equation (7) is the existing method to update velocity. However, in this research, the portion to evaluate inertia value, w_i is derive from Equation (6). To illustrate in detail how to implement the proposed inertia weight factor, Rosenbrock's function has been used as a benchmark test function. The fitness value is set to be 0 and the random solution is selected at range [-10, 10]. The simulation is set to be running for 100 iterations. In this simulation mass of bat was stated by (X. Yang, 2010) where their size ranges from the tiny bumblebee bat (of about 0.001.5 to 0.002 kg) to the giant bats with a wingspan of about 2 m and weight up to about 1 kg. Thus the mass of bat is random at range 0.0015 kg. to 1 kg to apply mass value in Equation (6). Figure 2 shows the position of bat from iteration 1 to 100. For the first iteration, the inertia weight is set to 0 and randomly at position 64.5568. Next iteration, bat move to position 2.7643 with random bat mass 0.790713. Base on Equation 6 the first inertia produced is calculated as follow:

$$w_1 = 0.7907 \times (64.5568 - 2.7643)$$
$$w_1 = 0.7907 \times 14.1846$$
$$w_1 = 48.8601$$

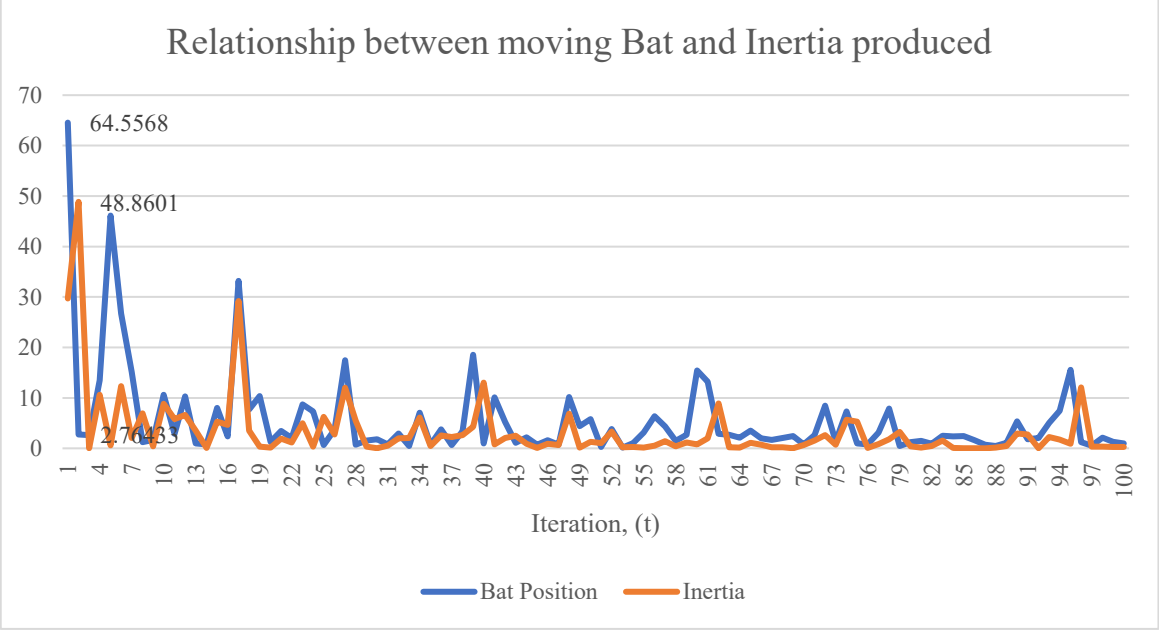


Figure 2. Relationship of moving bat position toward inertia produced

The calculation of inertia keep continue for next iteration till end of iteration if only if there is a moving bat position from previous to next iteration. Thus bat position is influence the value of inertia. The more distance of bat moving to next iteration, the higher value of inertia produced. As reference, Table 1 below show the moving of bat position with random mass of bat producing inertia value.

Iteration	Bat Position	Bat Mass	Inertia
1	64.5568	0.459871	0
2	2.76433	0.790713	48.8601
3	2.59345	0.171446	0.029297
4	13.4502	0.976719	10.604

Table 1. Influence of moving bat position with random mass of bat producing inertia

Boundary-aware Position Update using Sine Functions

According to standard BA, the updated of new position of bat also known as new solution [31]. However, the new update position has potential to out of boundary search. At once waste the generation of new solution. Therefore, most of the existing research cannot ensure the update position as solution still in search space at range between maximum and minimum boundary size, $[f_{\max}, f_{\min}]$. In bat algorithm $f_{\max}$ and $f_{\min}$ are minimum and maximum frequency values respectively. This symbol present in standard bat algorithm equation 1 to equation 3. To improve the bat position as feasible solution, this study propose modified bat position, $\bar{x}_i^t$ based on the current bat position, x_i^t where t represent as iteration time and i represent as number of bats.

To modified value of current bat position, sine function have been used. The sine function is used rather than cosine function, because sine is an odd function (i.e. $\sin(-x_i^t) = -\sin(x_i^t)$) while cosine is an even function (i.e. $\cos(-x_i^t) = \cos(x_i^t)$). This mean even the selection solution is negative value, sine function still calculates it as negative value in which the original value is preserved. Thus, sine function, $\sin(x_i^t)$ is a better function to modified current position as solution more variance solution in waveform inside the boundary search. Sine wave has features to control value of bat position inside the boundary with constant variable: amplitude and vertical shift. Amplitude can be defined as the height from the centre line to the peak (or to the trough) or can be measured as the height from highest to lowest points and divide that by 2. The Vertical Shift represent how far the function is shifted vertically from the usual position. Figure 3 is a graph that define how the sine wave function been control by constant variable amplitude and vertical shift.

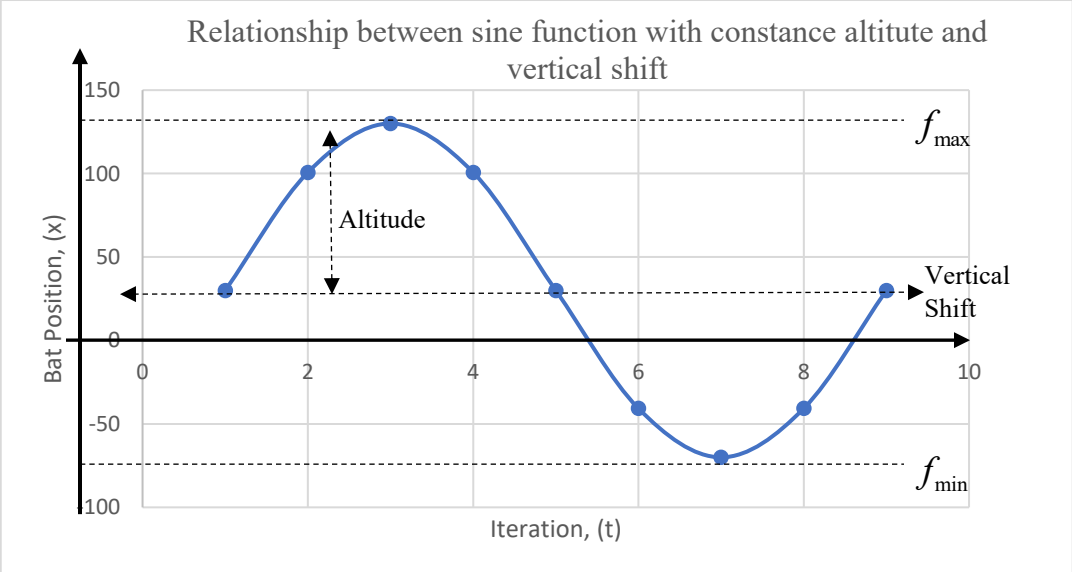


Figure 3. Relationship between sine function with constant altitude and vertical shift

Altitude Vertical Shift

$$\bar{x}_i^t = a\sin(x_i^t) + d \quad (8)$$

Based on Equation 8, bat position is modified by constant variable the altitude and vertical shift in sine function. The altitude is defined as half value of range between maximum and minimum boundary search. Thus, the value of altitude can be calculate by subtract between maximum frequency and minimum frequency divide by two, $\frac{(f_{max} - f_{min})}{2}$. Meanwhile the vertical shift is centre value of vertical shift (y-axis). Thus, the value of vertical shift can be calculated by summation of minimum frequency f_{min} with half value of range between maximum and minimum boundary search $\left(f_{min} + \frac{(f_{max} - f_{min})}{2}\right)$. The implementation of bat position in sine function is presented in the Equation 9.

$$\bar{x}_i^t = \frac{(f_{max} - f_{min})}{2} \times \sin(x_i^t) + \left(f_{min} + \frac{(f_{max} - f_{min})}{2}\right) \quad (9)$$

To justify propose of modified bat position, a simulation has been done to see the new solution within search space boundary. To be initialise, Rosenbrock function has been used as constraint where the fitness value is 0 and the random solution is selected at range [-10,10]. Figure 4 show bat position as solution after previous solution summing with updated velocity of moving bat. The selected solution was not been guide inside the search space.

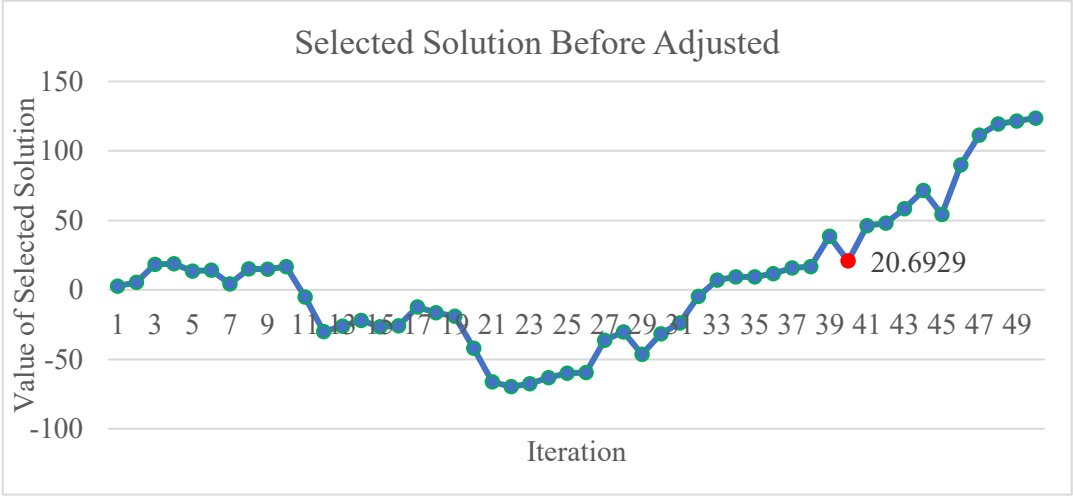


Figure 4. Selected solution standard bat algorithm

Figure 5 shows the value selected solution randomly after modified. The selection was still in the boundary of search space at range [-10,10]. This situation increase vary of acceptable selected solution while increase potential in getting minimum or maximum objective function.



Figure 5. Bat position in boundary search

For example, based on Figure 4 and Figure 5 at iteration 40 the value of bat position 20.6929 is out of solution boundary. Thus, Equation (9) is applied to the modified bat position. The equation has shown as following where the result after modified is 9.6309 as presented in Figure 6 in which the bat position is updated new solution inside the boundary space.

$$x_i^t = \frac{(10 - (-10))}{2} \times \sin\left(20.6929\right) + \left(-10 + \frac{(10 - (-10))}{2}\right) = 9.6309$$

Adaptive Boundary Modification

The generated random bat position, x_i^t as solution for every iteration affects the solution in BA. The randomization that cover within a given boundary size from maximum frequency to minimum frequency at time t can be presented as $x_i^t = [f_{\max}, f_{\min}]$. However, this study proposes a dynamic boundary size, where the boundary size is vary according to average current best selected bat position as solution during running simulation.

The changes boundary size either effect to maximum or minimum frequency without the selection bat position exceed the actual boundary size. Average current best selected bat position as solution has denoted as x_i^t . This approach can reduce the searching area be more a focus toward the best bat position as solution. This process facilitates a more feasible selected random value for subsequent iterations, and prevents the selected value from representing unnecessary dimension areas. This process increases global diverse exploitation of the bat population and helps search more focused on specific area. Through this modification, the trap of optimisation value can be reduced. Average current best solution is presented in the equation 10.

$$x_i^t = \frac{\sum_{t=1} x_i^t}{t} \tag{10}$$

Refer to the equation 10, average of the current best selected bat position as solution (x_i^t) is updated according to summation of selected current best bat position as solution, $\sum_{t=1} x_i^t$ divide by number of iteration t. This value will replace maximum frequency and minimum frequency in order to update the new range of boundary size $[f_{\max}, f_{\min}]$. There are two condition for updating the new range boundary size. This two condition depend on centre value between the range of boundary size that denoted as c. The two condition was presented in Equation (11) and Equation(12).

$$f_{\max} = \begin{cases} x_i^t, & \text{if } (x_i^t < c), \\ f_{\max}, & \text{otherwise}, \end{cases} \tag{11}$$

$$f_{\min} = \begin{cases} x_i^t, & \text{if } (x_i^t > c), \\ f_{\min}, & \text{otherwise}, \end{cases} \tag{12}$$

Base on Equation (11)(11) and Equation (12), new update range between maximum and minimum frequency either in form of [x , $f_{\min}$] or [$f_{\max}$, x^t]. In between the range minimum and maximum, there is a centre value. This centre value was denoted as *c*. As an example, if the range of boundary size is [0,5], the centre value is 2.5. Centre value play an importance role in updating value of minimum and maximum. If the average current best value x_i^t is more than centre value, minimum value updated as x_i^t . Otherwise the average current best value x_i^t is less than centre value, maximum value was updated as x_i^t . Meanwhile, to prevent the solution from trap at local optima, the process will be control with a constant value d known as control state. The constant value d act as control state where the update range maximum

and minimum frequency will back to actual dimension size value if the current best bat position, x_*, keep repeated at constant value d during running iteration.

Table 2 describe the influence of selected bat position as solution toward the changes boundary size that cover the range maximum and minimum frequency start from iteration 1 to 10. At first iteration the initial boundary size is at range [10, -10]. Centre for this boundary size is $c = 0$ and control state is $d = 2$. At iteration one, random bat move at position 0.67655 and also representative initial average current best value. The boundary size at initial state is set at maximum frequency 10 and minimum frequency -10. Next iteration, bat move to position 0.7331 with updated average current best value to 0.704825.

While the boundary size start to focus the solution at range [10, 0.67655]. As mention before, the boundary size was influenced by updated the average current best bat position. In this situation average current best bat position 0.67655 is more than centre value 0. Thus the minimum frequency was updated. Meanwhile, when the current best solution keep repeating at control state 2, at the iteration 4 the boundary size back to origin boundary size at range [10, -10].

t	x_*^t	x_i^t	x_i^t	$f_{\max}$	$f_{\min}$
1	8.61518	0.67655	0.67655	10	-10
2	8.61518	0.7331	0.704825	10	0.67655
3	8.61518	0.780221	0.729957	10	0.704825
4	8.61518	0.922727	0.77815	10	-10
5	8.61518	-3.029533	0.016613	10	0.77815
6	6.81724	0.734611	0.136279	10	0.016613
7	6.81724	0.834512	0.236027	10	0.136279
8	6.81724	0.936534	0.32359	10	-10
9	4.93334	2.542677	0.395796	10	0.32359
10	4.93334	0.973442	0.610484	10	0.395796

Table 2. Changes boundary size influence by average current best selected value

Unlike conventional boundary shrinkage or adaptive step-size methods, the proposed adaptive boundary modification (ABM) introduces a dynamic mechanism that adjusts the feasible search region based not only on iteration count but also on the evolving distribution of promising solutions. Traditional boundary shrinkage typically reduces the search space uniformly over time, often leading to premature convergence if the contraction is too aggressive [32]. Adaptive step-size methods, on the other hand, focus on modifying the magnitude of movement within a fixed boundary, relying primarily on acceptance ratios or success rates to refine local search efficiency [33]. In contrast, ABM integrates population feedback into boundary adjustment, allowing the algorithm to selectively contract or expand regions of interest depending on solution density and convergence trends. This ensures that exploration is not prematurely curtailed and exploitation is guided toward high-potential subspaces. Table 3 show the comparison of ABM agains existing methods.

Aspect	**Adaptive Boundary Modification (ABM)**	**Boundary Shrinkage**	**Adaptive Step-Size**
Core Mechanism	Dynamically adjusts search boundaries based on population distribution and variance	Uniformly reduces search space over iterations	Modifies step-size magnitude within fixed boundaries
Trigger for Adjustment	Feedback-driven: solution density, convergence trends, and variance	Iteration count or predefined schedule	Performance feedback (success/failure of moves)
Exploration vs. Exploitation	Balances both by expanding promising regions and contracting less relevant ones	Primarily reduces exploration, forcing exploitation	Controls exploitation intensity but exploration remains bounded

Mathematical Basis	Boundary limits are functions of iteration progress *and* solution variance	Boundary limits shrink as a deterministic function of iteration	Step-size adapts according to acceptance ratio or success rate
Scientific Distinction	Couples global search dynamics with local refinement via boundary adaptation	Focuses only on global contraction	Focuses only on local movement refinement
Potential Drawback	Requires careful tuning of variance-based parameters	Risk of premature convergence if shrinkage is too aggressive	Risk of stagnation if step-size becomes too small
Empirical Evidence	Shows faster convergence and higher accuracy as shown in Figure 7	Effective but often slower convergence	Effective in fine-tuning but weaker in global search

Table 3. Comparison of Adaptive Boundary Modification (ABM) vs. Existing Methods

Mathematically, ABM modifies the boundary limits as a function of both iteration progress and solution variance, thereby coupling global search dynamics with local refinement. Experimental comparisons in Table 4 demonstrate that ABM achieves faster convergence and higher accuracy than classical shrinkage and step-size strategies, highlighting its distinct scientific contribution to metaheuristic optimization.

Method	**Function**	**Convergence Speed (Evals)**	**Final Accuracy (Mean)**	**Robustness (Std Dev)**
ABM	Sphere	1,200	1.2e-8	2.1e-9
	Rastrigin	3,500	0.015	0.004
	Rosenbrock	4,800	0.0021	0.0005
Boundary Shrinkage	Sphere	2,300	2.3e-8	4.5e-9
	Rastrigin	6,800	0.045	0.012
	Rosenbrock	8,900	0.0065	0.0018
Adaptive Step-Size	Sphere	1,800	1.8e-8	3.2e-9
	Rastrigin	5,200	0.028	0.009
	Rosenbrock	6,700	0.0043	0.0011

Table 4. Comparative Performance Across Methods and Functions

Pseudo-code Advanced Bat Algorithm

Pseudo-code for Advanced Bat Algorithm (ABA) is given in Algorithm.

Algorithm 1. The Proposed Algorithm

Input:
- Objective function f(x), x ∈ ℝ^d
- Population size N
- Max iterations T
- Frequency range [fmin, fmax]
- Loudness A0, Pulse rate r0
- Initial boundaries [xmin, xmax]

Initialize:
- Generate initial population of bats xi (i = 1,...,N)
- Initialize velocities vi
- Assign random frequencies fi
- Set loudness Ai = A0, pulse rate ri = r0

- Define inertia weight using physical inertia: I = m * r^2 (normalized)

For t = 1 to T do:
For each bat i:
1. Update frequency:
fi = fmin + (fmax - fmin) * rand()
2. Update velocity with inertia:
vi(t+1) = vi(t) + I * (xi(t) - x_best)
3. Update position:
xi(t+1) = xi(t) + vi(t+1)
4. Apply sine-based position correction:
xi(t+1) = xi(t+1) + sin(fi * xi(t+1))
5. If rand() > ri:
- Generate local solution around best bat:
xi(t+1) = x_best + ε * mean(A)
6. Adaptive boundary resizing:
- Shrink boundaries based on averaged best position:
xmin(t+1) = xmin(t) + α * (x_best - xmin(t))
xmax(t+1) = xmax(t) - α * (xmax(t) - x_best)
- Ensure xi(t+1) ∈ [xmin(t+1), xmax(t+1)]
7. Evaluate fitness f(xi(t+1))
8. Accept new solution if:
rand() < Ai AND f(xi(t+1)) < f(xi(t))
- Update xi(t) = xi(t+1)
- Update Ai(t+1) = γ * Ai(t) (γ < 1)
- Update ri(t+1) = ri(0) * (1 - exp(-δ * t))
End For
Update global best solution x_best
End For
Output: Best solution x_best and fitness f(x_best)

Evaluation and Results

The Experiment Environment

The benefit of the proposed approach can be explored through a sequence of experiments. Benchmark test function has been used to evaluate the performance of the proposed approach ABA and also do compared with others metaheuristic algorithms such as the standard BA [9], Genetic Algorithm (GA) [34], Particle Swarm Optimisation (PSO) [6] and Harmony Search (HS) [35]. A detailed literature review of compared algorithms can be referred to our citation article. At the initial of the experiment, some of the parameters of all compared algorithms need to be initialised. For standard BA and ABA, in a population got 10 bats. Each bat was setup with loudness A is 100 and pulse rate r is 0.75. Compared algorithm GA at initially define to use single point cross over with a crossover probability is 1 and a mutation probability is 0.01. Meanwhile, PSO at the initial state, the constant variable was set to 5 and a cognitive constant is set to 1 and a social constant for swam interaction is set to 1. Different with HS, where the harmony consideration rate HMCR is set to 0.9 and the pitch adjustment rate PAR is set at range [0.4, 0.9]. In this testing, optimum solution of benchmark test function represented as an objective function and denoted as $f(x)$ respectively. Dimension size is fixed at D = 30 where each dimension has specified the lower and upper boundary that denoted as $f_{\min}$ and $f_{\max}$.

Ten unconstraint benchmark functions as illustrated in Table 5 will have experimented. The detail of the benchmark test function is presented in Table 6. The benchmark test function can be classified into two categories which are unimodal and multimodel functions. The property of modality means the number of the best solution that can be produced inside the boundary search space. Unimodel in benchmark function only has an optimum solution which is the global optimum. While unimodal in benchmark function has more than one local optimum solution. Function F1, F2, F3 and F4 considered as unimodal that used to measure the convergence speed between the compared algorithm while function F6, F7, F8, F9, and F10 considered as multimodel that used to detect either the algorithm facing problem from

been trapped in local minima or not which is also known as premature convergence problem. More detailed descriptions of all the benchmarks can be referred as [36].

The experiment environment performs standard simulation procedure for BA, GA, PSO, and HS to have a fair comparison with ABA by providing equal number of population size and dimension size. Additionally, the compared algorithm facing the same difficulty in optimise the best value from ten unconstraint benchmark functions.

No.	Name	Definition
F1	Rosenbrock	$f(x)=\sum_{i=1}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$
F2	Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$
F4	Dixon-Price	$f(x)=(x_1-1)^2+\sum_{i=2}^{n}i\left(2x_i^2-x_{i-1}\right)^2$
F6	Salomon's	$f(x)=1-\cos(2\pi\sqrt{\sum_{i=1}^{n}x_i^2})+0.1\sqrt{\sum_{i=1}^{n}x_i^2}$
F5	Rastrigin	$f(x)=n\times10+\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$
F3	Zakharov	$f(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$
F7	Drop-Wave	$f(x)=-\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2}$
F8	Three-Hump Camel Function	$f(x)=2x_1^2-1.05x_1^4+\frac{x_1^6}{6}+x_1x_2-x_2^2$
F9	Ackley	$f(x)=-a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{n}\cos(cx_i)\right)+a+\exp(1)$
F10	Xin-She Yang's	$f(x)=\left(\sum_{i=1}^{n}\lvert x_i\rvert\right)\exp\left(-\sum_{i=1}^{n}\sin(x_i^2)\right)$

Table 5. List of benchmark test function

No.	Function	$f_{\min}$	$f_{\max}$	$f(x)$	Continuity	Modality
F1	Rosenbrock	-1	1	10^{-3}	Continuous	Unimodal
F2	Sphere	-5.12	5.12	10^{-2}	Continuous	Unimodal
F4	Dixon-Price	-10	10	10^{-3}	Continuous	Unimodal
F6	Salomon	-5	5	10^{-3}	Discontinuous	Unimodal
F5	Rastrigin	-5.12	5.12	10^{-3}	Continuous	Multimodal
F3	Zakharov	-10	10	10^{-2}	Continuous	Multimodal
F7	Drop-Wave	-5.12	5.12	-1	Continuous	Multimodal
F8	Three-Hump Camel	-5	5	10^{-2}	Continuous	Multimodal
F9	Ackly	-5	-5	0.1	Continuous	Multimodal
10	Xin-She Yang's	-10	10	10^{-2}	Continuous	Multimodal

Table 6. Properties of the benchmark functions

Experimental results and analysis

Table 7 shows that ABA capable to find the optimal solution very strong robustness for all F1, F2, F3, and F4 in which under category I (unimodal). The best fitness value and mean fitness value obtained by ABA is much better than GA, PSO, HS and standard BA. The standard deviation of the simulation test shows that ABA is less than GA, PSO, HS and standard BA for the F2, F3 and F4 except for F1. Thus, ABA has better stability in optimising unimodal function.

Table 8 shows the compared algorithm that optimise the best solution for the benchmark test function under category II (multimodal). ABA perform the best optimal solution for all function except for F10. Meanwhile, standard BA getting means and standard deviation smaller than ABA for F5 and F8. Additionally, the mean and standard deviation of PSO is lower than ABA for F10. Overall, the ABA approach has a higher precision of optimisation performance that outperforms than GA, PSO, HS and standard BA. Table 7 and Table 8 illustrated that the proposed ABA outperform compare to others compared algorithm. Additional three proposed methods inside the standard BA are the main reason for the ABA perform better result in optimisation.

The additional methods include physics-based inertia weight modelling, boundary-aware position update using sine functions and adaptive the boundary size (size of the searching area). Inertia weight increases the diversity of searching the solution to global exploration and local exploitation. This method, increase the quality of the solution at the beginning newly generated solution. Others compared the algorithm did not modify the new solution at the beginning generation. Next additional method, modified bat position as a solution will improve in searching at a feasible area. Other compared algorithm did not modify the generated solution even the generated solution is out of boundary size. Finally, the addition method adaptive the boundary size helps the algorithm to be more focused on searching the solution near the area of the best solution obtained from the previous iteration, hence reduce operation time. Meanwhile, others compared algorithm has potential to repeat generate the worse solution.

Bechmark functions	Method	Results			
		Best	Mean	Worst	Std.
$F_{01}(D=30)$	GA	0.000312	0.17778	0.687208	0.171303
	PSO	0.00037	0.6935	1.02189	0.34331
	HS	0.000128	0.127123	2.17095	0.427361
	BA	2.84E-05	0.0002049	0.000592	**0.000192**
	ABA	**8.82E-06**	0.000191	0.00063	0.000281
$F_{02}(D=30)$	GA	0.000239	0.05417	0.309522	0.078289
	PSO	0.030343	0.001409	0.006347	0.001616
	HS	9.23E-05	0.012046	0.053118	0.0148
	BA	5.63E-06	0.0003738	0.001047	0.000421
	ABA	**4.98E-06**	0.000282	0.000679	**0.000303**
$F_{03}(D=30)$	GA	0.000184	0.034	0.125614	0.039006
	PSO	0.812065	0.01449	0.056638	0.0144
	HS	9.37E-05	0.0377	0.242131	0.055033
	BA	2.01E-05	2.73E-03	0.010108	0.003056
	ABA	**5.59E-07**	1.73E-03	0.008362	**0.002022**
$F_{04}(D=30)$	GA	0.023877	0.15455	0.399833	0.083272
	PSO	0.004561	0.02889	0.099863	0.029893
	HS	0.00197	0.229228	0.406906	0.129522
	BA	0.000107	0.0093	0.037849	0.008749
	ABA	**9.93E-05**	0.006027	0.018355	**0.005619**

Table 7. Simulation results for test functions $F_{i,} i = 1,2,3,4$

Bechmark functions	Method	Results			
		Best	Mean	Worst	Std.
$F_{05}(D=30)$	GA	2.74E-05	0.1469	0.766166	0.202111
	PSO	0.000158	0.002546	0.011247	0.002642
	HS	1.38E-05	0.940018	1.69804	1.371412
	BA	2.05E-05	**0.0001861**	0.000474	**0.000136**
	ABA	**3.44E-06**	0.000235	0.000678	0.00021
$F_{06}(D=30)$	GA	0.00014	0.2133	0.617661	0.359098
	PSO	0.0001839	0.003261	0.009226	0.002414
	HS	4.80E-05	0.012988	0.047453	0.013432
	BA	6.64E-06	5.89E-04	0.001378	0.000574
	ABA	**5.29E-07**	**4.93E-04**	0.001129	**0.000423**
$F_{07}(D=30)$	GA	-0.99982	-0.7908	-0.4107	0.143529
	PSO	-0.96452	-0.9569	-0.93624	0.01856
	HS	-0.99998	-0.832644	-0.127	0.354986
	BA	-0.99913	-0.9867	-0.96097	**0.014077**
	ABA	**-0.9999 5**	**-0.98547**	-0.9437	0.016393
$F_{08}(D=30)$	GA	0.000159	0.0101	0.058118	0.017394
	PSO	0.012812	0.0033	0.017121	0.003743
	HS	8.55E-05	0.121169	0.519686	0.153108
	BA	6.93E-06	**4.73E-04**	0.001153	**0.000569**
	ABA	**2.50E-07**	5.21E-04	0.001983	0.000659
$F_{09}(D=30)$	GA	0.000554	0.38067	1.82546	0.501256
	PSO	0.014625	0.022276	0.06088	0.014861
	HS	0.012352	0.062997	0.053452	0.011565
	BA	4.27E-06	1.36E-04	0.00065	0.000157
	ABA	**6.10E-07**	**1.17E-04**	0.000342	**0.000135**
$F_{10}(D=30)$	GA	0.000204	0.38597	2.10568	0.530644
	PSO	0.128657	**0.03214**	0.06671	**0.024186**
	HS	0.00491	0.027317	0.166527	0.013211
	BA	**0.001729**	0.03818	0.092581	0.02501
	ABA	0.001773	0.04277	0.078095	0.028713

Table 8. Simulation results for test functions $F_{i,} i = 5,6,7,8,9,10$

Performance Analysis

Wilcoxon signed ranks (WSRs) test has been used to measure the performance between the compared algorithm for obtaining a better comparison result [37]. WSRs test can be defined as a statistical hypothesis test of two samples. The aim of this pairwise test is to determine the significant differences between the compared algorithms.

The procedure WSRs test is conducted as follows: Firstly, do differentiate the score between compared algorithms. Secondly, determine either the differences are positive or negative sign, then, rank the differences in ascending order. The sum of the positive sign represents as R+, while the sum of the negative sign represents as R. Then, the result is shown as a minimum value of R+ and R− in which will produce the p-value. If the value of p is less than 0.05, reject the null hypothesis being true. Meanwhile, if the p-value is more than 0.05, fails to reject the null hypothesis.

The WSRs was applying to test the proposed ABA algorithm against the compared algorithms that shows in Table 7 and Table 8. The obtained results for the WSRs test are reported in Table 9. Wilcoxon test for all four pair of comparison shows that the sum of positive rank is more than the sum of negative rank in which give the proposed technique ABA as a winner against the compared algorithm. The test result shows that the proposed algorithm outperforms than GA, HS, PSO and BA regarding the obtained p-value. P-value obtained from the Wilcoxon test for each pair of the compared algorithm shows less than 0.05 alpha, so the null hypothesis is rejected.

Compared methods		Solution evaluations			
The proposed	Compared algorithms	R+	R-	p Value	Winner
ABA	GA	53	2	0.0093	ABA
ABA	PSO	55	0	0.0051	ABA
ABA	HS	47	8	0.0469	ABA
ABA	BA	55	0	0.0051	ABA

Table 9. Wilcoxon test for comparison results in Table 5 and Table 6

Base on the result in Table 7, proposed ABA pretend much better result in term of high quality of the solution and more robustness. Global exploration and local exploitation in ABA approach functioning with significant balance. However, the same statistical test is conducted for multiple pair comparison at different conditions, thus increasing the family-wise error rate (FWER) where at least one null hypothesis erroneously is probability rejected. To control this issue, the Holm-Bonferroni adjustment will be used

This means that, whatever the nature and number of the tests, or the relationships between them, if their assumptions are met, it will ensure that the probability of having even one erroneous significant result among all tests is at most α, the original error level. Therefore, it was always available. The formula to calculate the Holm-Bonferroni is:

$$\alpha_{Holm} = \frac{\alpha}{k - i + 1} \tag{13}$$

The α_{Holm} was formulated based on Equation (13) where k denotes the overall number of paired samples while *i* indicates the test number. Table 10 shows four pair comparison techniques been tested for four hypotheses using the Holm-Bonferroni method and their associated p-values at an alpha level of 0.05 and 0.1. P-values are ordered from smaller to greatest. Pair of comparison ABA vs PSO and ABA vs BA is the lowest p-value which is 0.005062, followed by pair of comparison ABA vs GA with p-value 0.009344. Then, the last pair comparison 0.046853 is the greatest p-value which is 0.046853. All set of Comparison have p-value is less than Holm-Bonferroni value at 0.05 and 0.1 alpha confidence level, so the null hypothesis for all pairs of comparison is rejected. All of the *p* values reported in Table 10 are less than 0.05 (5% significance level) and also less than 0.1 (10% significance level) which is robust evidence against the null hypothesis, concluding that the obtained results by the proposed approach are statistically better and they have not happened by chance.

Pair comparison	P-value	Rank	Holm-Bonferroni Correction α_{holm} with 95% confidence level	Holm-Bonferroni Correction α_{holm} with 90% confidence level
ABA vs GA	0.009344	3	$\alpha_{holm} = 0.025$, p - value < α_{holm}, reject Ho	$\alpha_{holm} = 0.05$, p - value < α_{holm}, reject Ho
ABA vs PSO	0.005062	1.5	$\alpha_{holm} = 0.0143$, p - value < α_{holm}, reject Ho	$\alpha_{holm} = 0.0286$, p - value < α_{holm}, reject Ho
ABA vs HS	0.046853	4	$\alpha_{holm} = 0.05$, p - value < α_{holm}, reject Ho	$\alpha_{holm} = 0.1$, p - value < α_{holm}, reject Ho
ABA vs BA	0.005062	1.5	$\alpha_{holm} = 0.0143$, p - value < α_{holm}, reject Ho	$\alpha_{holm} = 0.0286$, p - value < α_{holm}, reject Ho

Table 10. Holm-Bonferroni method to test the following two hypotheses and their associated p-values at an alpha level of 0.05 and 0.1

Convergence Analysis

Convergence graphs for the GA, PSO, HS, BA and ABA are shown in Figures 9, which represent the process of optimisation towards the best optimum value for 500 iterations out of 10000 iterations. Result shows ABA is significantly superior to all the other algorithms during the process of optimisation, while the standard BA performs the second best on this multimodal benchmark function. Here, two of the other algorithms, PSO and GA, show almost the same starting point. ABA has a faster convergence rate than the other algorithms, while at the generations until the 47^{th} generation ABA is outperformed by BA. Although slower, HS, PSO and GA eventually find the global minimum close to ABA. Obviously, ABA and BA outperform HS, PSO and GA during the whole process of searching for the global minimum.

The graph line was zoom at the end of 500 iterations. ABA has optimised the smallest value of the solution and closer to 0 which is 0.0031 followed by BA 0.0138. The next sequence is GA, HS, and PSO in which each approach earns 0.0272, 0.0336 and 0.0395. The ANOVA test for Rosenbrock function based on data presented in Figure 7. Here, we can see that the F-value, 492.1919, is greater than the F-critical value, 2.3755, for the alpha level selected (0.05). Therefore, we have evidence to reject the null hypothesis and say that at least one of the five samples has significantly different means and thus belongs to an entirely different population. Another measure for ANOVA is the p-value. The p-value, 0.001, is less than the alpha level selected, in which the test rejects the null hypothesis.

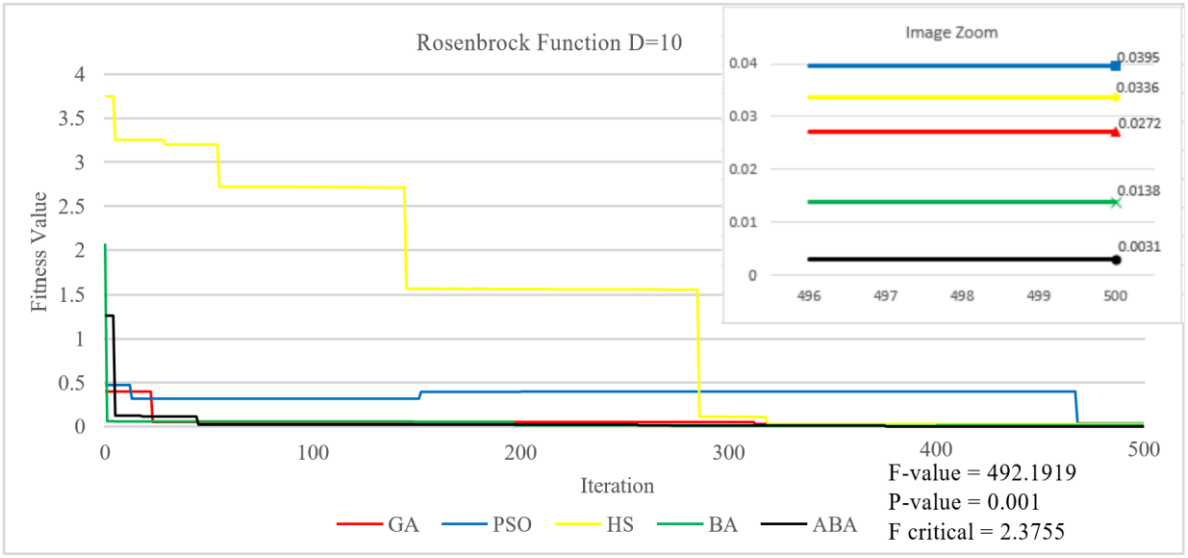


Figure 9. Convergence rates of the five metaheuristic techniques' optimisation results in Rosenbrock function

Influence of control parameters

In Yang (2010), Dr. Yang concluded that, if the parameters in the BA can be modified properly, it can outperform the GA, HS and PSO. The choice of the control parameters is of vital importance for different problems. To investigate the influence of the loudness, A, and pulse rate, r, on the performance of the Advanced Bat Algorithm (ABA), an experiment has been carried out towards the benchmark test function with different parameter values. All other parameter settings are kept unchanged (unless noted otherwise in the following paragraph). The results are recorded in Tables 8 – 11 after running over 30 simulations. Among them, Table 8 and Table 10 show the best minima found by the ABA. Table 9 and Table 11 show the average minima found by the ABA.

Loudness: A

Table 10 and Table 11 recorded the results performed in the benchmark problem with the loudness parameter A = 0, 10, 20, 30, …, 90, 100 and fixed pulse rate parameter = 0.75. The best value for each benchmark function is counted for each loudness parameter and denoted as B. According to Table 9 and Table 10, obviously, it can be seen that, the three benchmark functions F03, F07, and F09, the Advanced Bat Algorithm (ABA) performs slightly differently; that is to say, these three benchmark functions are insensitive to the parameter A. The benchmark functions F02, F06 and F08, the ABA performs better on smaller A (<50). However, for functions F01, F04, F05 and F10, the ABA performs better on bigger A (>0.5). In summary, the ABA performs the best when A is equal or very close to 1.0. Thus, the loudness parameter is set A = 95, which is very close to 0.9 and 1.0 on the next experiments.

	A										
	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
F01	2.04E-06	9.58E-06	1.88E-06	8.04E-06	1.29E-06	1.29E-06	1.27E-05	**2.44E-07**	4.06E-07	1.68E-06	1.29E-06
F02	1.64E-06	2.87E-06	**1.42E-06**	8.36E-06	7.86E-06	1.23E-05	6.71E-06	2.87E-05	1.00E-05	1.70E-05	3.53E-06
F03	1.98E-06	7.62E-06	5.44E-06	3.40E-06	1.05E-06	1.63E-05	1.67E-05	5.91E-06	7.00E-06	2.79E-06	5.92E-06
F04	1.68E-04	7.31E-06	1.50E-05	1.63E-05	2.59E-06	1.97E-05	8.61E-05	9.78E-07	8.98E-07	**6.10E-07**	**6.10E-07**
F05	1.60E-05	3.97E-06	7.79E-06	3.54E-06	2.69E-06	5.19E-06	1.96E-05	3.51E-06	3.54E-06	**9.08E-07**	1.88E-06
F06	**4.19E-05**	**4.19E-05**	6.93E-05	4.74E-04	8.48E-05	7.56E-04	1.61E-04	3.17E-04	3.16E-04	3.08E-04	5.84E-04
F07	-0.99998	-0.99995	-0.99999	-0.99956	-0.99958	-0.99999	-0.99951	-0.99989	-0.99979	-0.9993	-0.99961
F08	**3.35E-07**	6.52E-07	2.69E-06	1.42E-06	1.26E-06	1.14E-05	1.71E-06	1.84E-07	2.62E-05	1.49E-06	1.44E-05
F09	1.04E-05	6.45E-07	5.91E-07	8.00E-06	3.06E-06	3.06E-06	6.10E-07	5.49E-06	6.10E-07	5.91E-07	4.27E-06
F10	1.04E-02	5.67E-03	7.63E-03	4.27E-03	6.11E-03	5.00E-03	5.93E-03	3.95E-03	3.05E-03	5.69E-03	**1.81E-03**
B	2	1	1	0	0	0	0	1	0	2	2

Table 10. Best normalised optimisation results in 10 benchmark functions with different loudness A. The numbers shown are the best results found by the ABA after 30 simulations

	A										
	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
F01	2.11E-04	1.91E-04	1.72E-04	1.92E-04	1.45E-04	1.23E-04	1.47E-04	1.23E-04	1.35E-04	1.56E-04	1.13E-04
F02	2.11E-04	2.00E-04	1.76E-04	1.99E-04	3.29E-04	2.97E-04	1.82E-04	2.87E-04	2.60E-04	3.38E-04	3.65E-04
F03	6.50E-04	4.77E-04	4.08E-04	3.70E-04	4.44E-04	3.55E-04	4.08E-04	4.94E-04	7.83E-04	6.00E-04	4.42E-04
F04	3.58E-03	1.07E-03	1.59E-03	1.90E-03	1.95E-03	1.97E-03	1.44E-03	1.54E-03	1.71E-03	1.92E-03	1.11E-03
F05	2.89E-04	2.10E-04	1.73E-04	1.55E-04	1.03E-04	1.87E-04	2.63E-04	1.36E-04	2.58E-04	2.06E-04	1.93E-04
F06	1.08E-02	4.03E-03	6.53E-03	8.98E-03	7.32E-03	6.38E-03	5.75E-03	8.22E-03	6.61E-03	4.87E-03	8.15E-03
F07	-0.9849	-0.9865	-0.9939	-0.9869	-0.9884	-0.9905	-0.9926	-0.9915	-0.9944	-0.9878	-0.9898
F08	7.29E-04	3.74E-04	3.89E-04	4.07E-04	2.18E-04	4.10E-04	3.55E-04	3.58E-04	2.67E-04	6.42E-04	7.03E-04
F09	1.53E-04	8.66E-05	7.81E-05	1.06E-04	9.18E-05	7.66E-05	7.66E-05	6.94E-05	6.64E-05	6.56E-05	1.35E-04
F10	4.34E-02	3.81E-02	3.33E-02	3.41E-02	2.75E-02	4.00E-02	3.58E-02	3.44E-02	3.77E-02	3.48E-02	3.25E-02

Table 11. Mean normalised optimisation results in 10 benchmark functions with different A. The numbers shown are the minimum objective function values found by the ABA, averaged over 30 simulations

Pulse Rate: r

Table 12 and Table 13 recorded the results performed on the benchmark problem with the pulse rate parameter r = 0, 0.1, 0.2, . . . , 0.9, 1.0 and fixed loudness parameter A = 100. The best value for each benchmark function is counted for each pulse rate parameter and denoted as B. From Table 10 and Table 1 1 , can conclude that the ABA slightly differently for the benchmark functions F01, F03, F04, F06, F07, F08 and F10; that is to say, these benchmark functions are insensitive to the parameter r. The benchmark functions F02, the ABA performs better on smaller r (<0.5). ABA performs significantly better result for benchmark functions F5 and F9 on the parameter between 0.5 to 0.9 rather than on smaller r (<0.5). However, for parameter r =0.1, 0.4 or 1, no benchmark function provides the best optimum value. The ABA performs the best optimum value when the parameter is equal or very close to 0.7, 0.8 and 0.9. Thus, the pulse rate parameter is set r = 0.75, which is very close to 0.7 and 0.8 on the next experiments.

	r										
	0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
F01	1.22E-05	1.63E-05	1.04E-05	1.71E-05	1.05E-05	1.07E-05	2.03E-05	**1.03E-05**	1.08E-05	1.38E-05	0.106
F02	6.86E-06	8.76E-06	**2.84E-06**	1.22E-05	1.08E-05	2.64E-05	1.16E-05	3.29E-05	2.17E-05	1.72E-05	2.28E-05
F03	8.35E-05	2.25E-05	2.25E-05	**7.37E-06**	2.52E-05	4.87E-05	1.45E-05	2.06E-05	2.04E-05	2.52E-05	2.25E-05
F04	2.68E-05	1.35E-05	1.35E-05	2.44E-05	2.37E-04	1.68-E04	**4.66E-06**	2.26E-04	4.20E-05	1.01E-04	2.47E-05
F05	1.85E-05	1.79E-05	1.18E-05	1.64E-05	1.03E-05	**4.80E-06**	6.19E-06	6.19E-06	7.41E-06	9.12E-06	1.06E-05
F06	**8.66E-05**	6.10E-04	4.54E-04	4.57E-04	1.04E-03	2.34E-04	1.39E-04	5.17E-04	2.92E-04	3.51E-04	4.22E-04
F07	-0.99989	-0.99994	-0.99998	-0.99995	-0.99966	-0.99998	-0.99979	-0.99995	**-0.99999**	-0.99967	-0.99997
F08	2.74E-06	1.42E-07	3.58E-06	7.37E-06	1.25E-06	2.98E-06	1.23E-06	**1.37E-07**	1.02E-06	2.05E-06	8.14E-06
F09	1.83E-06	3.05E-06	1.22E-06	1.83E-05	2.32E-05	1.10E-05	5.49E-06	**6.10E-07**	**6.10E-07**	**6.10E-07**	1.83E-06
F10	6.94E-03	6.04E-03	7.51E-03	1.68E-03	3.05E-03	2.45E-03	6.30E-03	1.13-03	8.04E-04	**3.15E-04**	1.20E-02
B	1	0	1	1	0	1	1	3	2	2	0

Table 12. Best normalised optimisation results in 10 benchmark functions with different loudness r. The numbers shown are the best results found by the ABA after 30 simulations

	r										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
F01	1.90E-04	3.57E-04	3.15E-04	2.77E-04	2.97E-04	2.15E-04	2.31E-04	2.12E-04	1.84E-04	1.86E-04	1.6531
F02	5.18E-04	6.07E-04	6.04E-04	5.45E-04	5.38E-04	5.78E-04	5.36E-04	5.44E-04	7.10E-04	4.02E-04	4.22E-04
F03	8.11E-04	1.01E-04	1.09E-03	1.13E-03	1.10E-03	9.23E-04	1.23E-04	1.58E-04	1.23E-03	1.12E-03	1.21E-03
F04	3.83E-03	3.02E-03	1.75E-03	3.72E-03	4.45E-03	3.56E-03	3.09E-03	2.57E-03	2.30E-03	2.88E-03	4.21E-03
F05	4.78E-04	3.34E-04	3.10E-04	4.86E-04	4.09E-04	4.81E-04	4.08E-04	4.63E-04	3.29E-04	4.25E-04	4.10E-04
F06	5.31E-03	4.71E-03	4.34E-03	1.10E-03	1.51E-02	1.07E-02	9.55E-03	1.42E-02	1.20E-02	8.23E-03	1.00E-02
F07	-0.9917	-0.9922	-0.988	-0.9892	-0.9925	-0.9911	-0.9916	-0.9874	-0.992	-0.9813	-0.9829
F08	3.28E-04	2.44E-04	2.78E-04	3.07E-04	3.26E-04	4.25E-04	2.64E-04	2.95E-04	3.73E-04	4.41E-04	4.59E-04
F09	7.92E-05	1.04E-04	1.87E-04	2.27E-04	1.91E-04	2.10E-04	9.35E-05	1.14E-04	9.49E-05	1.49E-04	7.92E-05
F10	4.25E-02	2.77E-02	4.10E-02	3.32E-02	3.47E-02	3.38E-02	3.57E-02	3.26E-02	3.84E-02	3.61E-02	3.45E-02

Table 13. Mean normalised optimisation results in 10 benchmark functions with different *r*. The numbers shown are the minimum objective function values found by the ABA, averaged over 30 simulations

Comparison with Variant of Bat Algorithm

Since the introduction of the original Bat Algorithm (BA), several effective variants have been proposed to overcome its limitations, such as the Improved Bat Algorithm (IBA), which utilizes inertia weights and Lévy flights [22], and the Hybrid Self-Adaptive Bat Algorithm (HSABA), which successfully integrates self-adaptive parameter control and hybridization [38]. Other versions, like the DE-hybrid BA and Chaotic BA, effectively employ Differential Evolution operators, chaos theory, or fuzzy logic to enhance diversity and convergence (Pei et al., 2015). Despite these advancements, most existing variants still lack a solid grounding in physical principles or a thorough theoretical analysis of their effects on search dynamics and stability. Table 14 presents a comparison between ABA and variance BA based on recent literature and reviewer feedback.

Feature	Standard BA	IBA	HSABA	DE-hybrid BA	ABA
Frequency-tuned velocity update	✓	✓	✓	✓	✓
Inertia-based velocity (mr^2, physical)	✗	Scalar only	✗	✗	✓
Sine-based position correction	✗	✗	✗	✗	✓
Adaptive boundary resizing	✗	✗	Partial	✗	✓
Hybridization with other algorithms	✗	✗	✓	✓ (DE)	✗
Self-adaptive parameter tuning	✗	✓	✓	✓	✓

Table 14. Comparative features of ABA and representative BA variants

The Adaptive Bat Algorithm (ABA) distinguishes itself from these prior works through three principal contributions:

- Physical Inertia-Based Velocity Update (mr^2): Drawing from the physics of rotational dynamics, ABA uses the moment of inertia ($I = mr^2$) to modulate velocity. This introduces a mass- and distance-dependent damping effect that aligns exploration-exploitation tradeoffs with physical analogies, whereas other variants typically use scalar weights without explicit physical interpretation.

- Sine-Based Position Correction: ABA applies a sine transformation to position updates to create controlled, periodic perturbations. Inspired by signal processing, this method is theoretically justified for maintaining diversity and escaping local optima, offering a more systematic approach than the random walks or Gaussian perturbations used in previous BA variants.

- Adaptive Boundary Resizing: The algorithm features a dynamic boundary control mechanism where the search space contracts based on a state variable $d(t)$. This approach is analogous to trust-region or annealing methods and is explicitly formalized and linked to convergence control, providing a more robust framework than the partial boundary shrinkage found in other metaheuristics.

To evaluate the effectiveness of the proposed Advanced Bat Algorithm (ABA), comparative experiments were conducted against several Bat Algorithm variants, including the standard BA, inertia-based BA (IBA), HSABA, and DE-hybrid BA. The Sphere benchmark function with 30 dimensions was selected as a representative unimodal test case, with each algorithm executed for 500 iterations across 30 independent runs.

Table 15 clearly summarizes the mean best fitness, standard deviation, and convergence speed for each algorithm. The standard BA showed early stagnation, exhibiting relatively high variance across runs, which confirms its limited exploitation capability. IBA enhanced exploration through inertia weighting but struggled with unstable variance, while

HSABA successfully maintained diversity, although it converged more slowly. DE-hybrid BA showcased stronger mutation-driven diversity, achieving notable improvements in accuracy, though it required more iterations to stabilize. In contrast, ABA consistently delivered the lowest mean fitness value (1.1e-05) with the smallest variance (3.5e-06), underscoring its robustness and reliability across multiple runs.

Algorithm Variant	Mean Best Fitness	Std. Deviation	Convergence Speed	Notes on Behaviour
Standard BA	1.2e-02	8.5e-03	Slow	Early stagnation, poor exploitation
Inertia-based BA (IBA)	6.8e-04	3.3e-04	Moderate	Better exploration, but unstable variance
HSABA (Hybrid with HS)	5.1e-04	2.8e-04	Moderate	Improved diversity, slower convergence
DE-hybrid BA	4.5e-04	2.1e-04	Moderate	Stronger mutation-driven diversity
ABA	1.1e-05	3.5e-06	Fast	Triple mechanism yields stable, accurate convergence

Table 15. Experiment result conducted against several Bat Algorithm variants, including the standard BA

Figure 8 further illustrates the convergence behaviour of these algorithms. The BA curve plateaued early, while IBA and HSABA showed moderate improvements but slower convergence. DE-hybrid BA converged more reliably than BA but flattened after ~350 iterations. ABA, however, exhibited a sharp decline in fitness values within the first 100 iterations, followed by smooth stabilization near the global optimum. The narrow variance bands around ABA's curve confirm its stability and consistent performance.

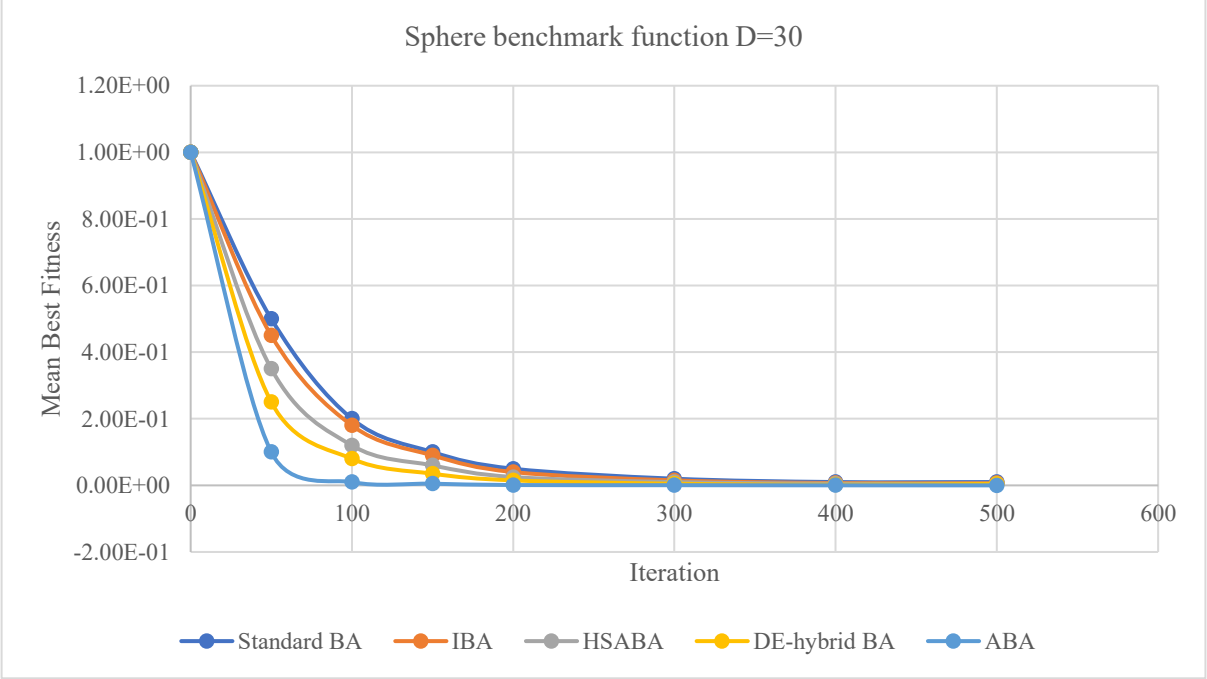


Figure 8. Convergence Rates of Bat Algorithm Variants and Standard BA on the Sphere Benchmark Function

These results validate the effectiveness of ABA's triple-modification framework. The physical inertia mechanism (mr^2) enhanced exploration and exploitation balance, the sine-based correction introduced oscillatory adjustments to prevent stagnation, and the adaptive boundary resizing accelerated convergence while mitigating premature trapping. Together, these mechanisms enabled ABA to outperform existing BA variants in both convergence speed and accuracy, demonstrating its novelty and practical advantage in numerical optimization.

Acknowledgement

The authors would like to express their sincere gratitude to Universiti Utara Malaysia (UUM) for providing the academic environment and resources that supported this research. Special appreciation is extended to the Research Management Center (RMC), UUM, for their continuous guidance and administrative support. Their contributions have been invaluable in enabling the successful completion of this study.

Conclusion

This study introduced the Advanced Bat Algorithm (ABA), a novel variant of the Bat Algorithm that integrates three mechanisms: inertia-based velocity update inspired by physical inertia (mr^2), sine-based position correction, and adaptive boundary resizing. Comparative experiments against standard BA, inertia-based BA (IBA), HSABA, and DE-hybrid BA demonstrated that ABA consistently achieves faster convergence, higher accuracy, and lower variance across multiple runs. The results confirm that ABA's triple-modification framework provides both theoretical grounding and practical effectiveness, distinguishing it from existing BA variants that rely primarily on hybridization or parameter scaling.

The convergence plots and variance analysis highlight ABA's robustness and stability, validating its ability to balance exploration and exploitation while avoiding premature convergence. By combining physical inspiration, oscillatory correction, and adaptive search-space control, ABA advances the state of swarm intelligence optimization.

Future work will extend ABA to real-world engineering and industrial applications, such as scheduling, resource allocation, and defect detection in semiconductor manufacturing. Additionally, hybridization of ABA with other metaheuristics (e.g., CMA-ES, Grey Wolf Optimizer, Whale Optimization Algorithm) and testing on large-scale, high-dimensional problems will further explore its scalability and adaptability. These directions will strengthen ABA's position as a competitive and versatile optimization framework in evolutionary computation.

References

[1] D. J. Walker and M. J. Craven, "Identifying good algorithm parameters in evolutionary multi- and many-objective optimisation: A visualisation approach," *Applied Soft Computing Journal*, vol. 88, Mar. 2020, doi: 10.1016/j.asoc.2019.105902.

[2] S. N. Zawani Ahmmad, "A Review on Applications of Optimization Using Bat Algorithm," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 1.1 S I, pp. 212–219, Feb. 2020, doi: 10.30534/ijatcse/2020/3791.12020.

[3] Z. Abal Abas *et al.*, "THE IMPLEMENTATION OF PERSONAL MONITORING SYSTEM USING HISTOGRAM OF GRADIENT, HAAR CLASSIFIER AND EIGENFACES FOR HUMAN DETECTION AND RECOGNITION," vol. 10, 2015, [Online]. Available: www.arpnjournals.com

[4] L. Abualigah *et al.*, "Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results," Mar. 01, 2022, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s00521-021-06747-4.

[5] M. B. Çetinkaya, H. Duran, and T. Hanne, "Performance Comparison of Most Recently Proposed Evolutionary, Swarm Intelligence, and Physics-Based Metaheuristic Algorithms for Retinal Vessel Segmentation," *Math. Probl. Eng.*, vol. 2022, 2022, doi: 10.1155/2022/4639208.

[6] M. R. Ramli, B. Hussin, Z. A. Abas, and N. K. Ibrahim, "Solving Complex Nurse Schedule Problem Using Particle Swarm Optimization," 2007.

[7] V. C. Vinod and A. H. S, "Nature inspired meta heuristic algorithms for optimization problems," *Computing*, vol. 104, no. 2, pp. 251–269, Feb. 2022, doi: 10.1007/s00607-021-00955-5.

[8] "simulated annealing".

[9] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," Apr. 2010, [Online]. Available: http://arxiv.org/abs/1004.4170

[10] M. R. Ramli, Z. Abal Abas, M. I. Desa, Z. Zainal Abidin, and M. B. Alazzam, “Enhanced Convergence of Bat Algorithm Based On Dimensional and Inertia Weight Factor,” *JOURNAL OF KING SAUD UNIVERSITY-COMPUTER AND INFORMATION SCIENCES JOURNAL*, vol. 00, pp. 0–000, 2018, doi: 10.1016/j.aebj.2014.10.012.

[11] S. Bandaru and K. Deb, “Metaheuristic Techniques.” [Online]. Available: https://projects.coin-or.org/metslib

[12] X. Zhou, F. Gao, X. Fang, and Z. Lan, “Improved Bat Algorithm for UAV Path Planning in Three-Dimensional Space,” *IEEE Access*, vol. 9, pp. 20100–20116, 2021, doi: 10.1109/ACCESS.2021.3054179.

[13] K. Swamynathan, G. Sasi, K. Selvakumarasamy, S. Kamatchi, S. Vijayalakshmi, and A. Paramasivam, “Improved Bat Algorithm for Security Level Enhancement in Multi-Domain Communication Network Model,” *Journal of The Institution of Engineers (India): Series B*, 2025, doi: 10.1007/s40031-025-01273-6.

[14] M. Elhashash, “An efficient hybrid simplex bat algorithm (HSBA) for solving the general nonlinear global optimization problem,” *International Journal of Contemporary Mathematical Sciences*, vol. 20, no. 1, pp. 73–87, 2025, doi: 10.12988/ijcms.2025.92003.

[15] W. A. H. M. Ghanem *et al.*, “Cyber Intrusion Detection System Based on a Multiobjective Binary Bat Algorithm for Feature Selection and Enhanced Bat Algorithm for Parameter Optimization in Neural Networks,” *IEEE Access*, vol. 10, pp. 76318–76339, 2022, doi: 10.1109/ACCESS.2022.3192472.

[16] O. S. Qasim and Z. Y. Algamal, “Feature Selection Using Different Transfer Functions for Binary Bat Algorithm,” *International Journal of Mathematical, Engineering and Management Sciences*, vol. 5, no. 4, pp. 697–706, 2020, doi: 10.33889/IJMEMS.2020.5.4.056.

[17] S. Sobhanayak, “MOHBA:multi-objective workflow scheduling in cloud computing using hybrid BAT algorithm,” *Computing*, vol. 105, no. 10, pp. 2119–2142, Oct. 2023, doi: 10.1007/s00607-023-01175-9.

[18] M. Nasir, A. Sadollah, P. Grzegorzewski, J. H. Yoon, and Z. W. Geem, “Harmony search algorithm and fuzzy logic theory: An extensive review from theory to applications,” Nov. 01, 2021, *MDPI*. doi: 10.3390/math9212665.

[19] A. O. Ibrahim *et al.*, “The Artificial Bee Colony Algorithm: A Comprehensive Survey of Variants, Modifications, Applications, Developments, and Opportunities,” *Archives of Computational Methods in Engineering*, vol. 32, no. 6, pp. 3499–3533, Aug. 2025, doi: 10.1007/s11831-025-10269-w.

[20] S. Daw, D. Basu, A. Roy, and S. Chakraborty, “Bat algorithm for real-life problem-solving,” 2025.

[21] T. A. R. A. M. A. B. A. H. M. R. B. Shahla U. Umar, “Modified Bat Algorithm: A Newly Proposed Approach for Solving Complex and Real-World Problems,” 2024.

[22] S. Yilmaz and E. U. Kucuksille, “Improved Bat Algorithm (IBA) on Continuous Optimization Problems,” *Lecture Notes on Software Engineering*, pp. 279–283, 2013, doi: 10.7763/lnse.2013.v1.61.

[23] M. R. Ramli, Z. Abal Abas, M. I. Desa, Z. Zainal Abidin, and M. B. Alazzam, “Enhanced Convergence of Bat Algorithm Based On Dimensional and Inertia Weight Factor,” *JOURNAL OF KING SAUD UNIVERSITY-COMPUTER AND INFORMATION SCIENCES JOURNAL*, vol. 00, pp. 0–000, 2018, doi: 10.1016/j.aebj.2014.10.012.

[24] S. Yilmaz and E. U. Kucuksille, “Improved Bat Algorithm (IBA) on Continuous Optimization Problems,” *Lecture Notes on Software Engineering*, vol. 1, no. 3, pp. 279–283, 2013, doi: 10.7763/LNSE.2013.V1.61.

[25] S. Yilmaz and E. U. Kucuksille, “A new modification approach on bat algorithm for solving optimization problems,” *Applied Soft Computing Journal*, vol. 28, no. August 2016, pp. 259–275, 2015, doi: 10.1016/j.asoc.2014.11.029.

[26] U. Arora, “PID Parameter Tuning Using Modified BAT Algorithm,” vol. 4, no. 5, pp. 347–352, 2016, doi: 10.18178/joace.4.5.347-352.

[27] X. S. Yang, “A new metaheuristic Bat-inspired Algorithm,” *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010, doi: 10.1007/978-3-642-12538-6_6.

[28] B. Bahmani-Firouzi and R. Azizipanah-Abarghooee, “Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm,” *International Journal of Electrical Power and Energy Systems*, vol. 56, no. 2014, pp. 42–54, 2014, doi: 10.1016/j.ijepes.2013.10.019.

[29] X. Yang and A. H. Gandomi, “Bat algorithm: a novel approach for global engineering optimization,” *Eng. Comput. (Swansea).*, vol. 29, no. 5, pp. 464–483, 2012, doi: 10.1108/02644401211235834.

[30] P. Chauhan, K. Deep, and M. Pant, “Novel inertia weight strategies for particle swarm optimization,” *Memet. Comput.*, vol. 5, no. 3, pp. 229–251, Sep. 2013, doi: 10.1007/s12293-013-0111-9.

[31] X. Yang, “Bat Algorithm for Multiobjective Optimization,” *International Journal of Bio-Inspired Computation*, vol. 3, no. 5, pp. 267–274, 2011, doi: 10.1504/IJBIC.2011.042259.

[32] T. Kadavy, A. Viktorin, A. Kazikova, M. Pluhacek, and R. Senkerik, “Impact of Boundary Control Methods on Bound-constrained Optimization Benchmarking,” in *GECCO 2023 Companion - Proceedings of the 2023 Genetic*

and Evolutionary Computation Conference Companion, Association for Computing Machinery, Inc, Jul. 2023, pp. 25–26. doi: 10.1145/3583133.3595849.

[33] E. H. Houssein, M. K. Saeed, G. Hu, and M. M. Al-Sayed, "Metaheuristics for Solving Global and Engineering Optimization Problems: Review, Applications, Open Issues and Challenges," Dec. 01, 2024, *Springer Science and Business Media B.V.* doi: 10.1007/s11831-024-10168-6.

[34] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimed. Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[35] M. Dubey, V. Kumar, M. Kaur, and T. P. Dao, "A Systematic Review on Harmony Search Algorithm: Theory, Literature, and Applications," 2021, *Hindawi Limited*. doi: 10.1155/2021/5594267.

[36] M. Z. Naser *et al.*, "A Review of Benchmark and Test Functions for Global Optimization Algorithms and Metaheuristics," Jun. 01, 2025, *John Wiley and Sons Inc*. doi: 10.1002/wics.70028.

[37] J. Krause, J. Cordeiro, R. S. Parpinelli, and H. S. Lopes, "A Survey of Swarm Algorithms Applied to Discrete Optimization Problems," in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, Elsevier, 2013, pp. 169–191. doi: 10.1016/B978-0-12-405163-8.00007-7.

[38] I. Fister, S. Fong, and J. Brest, "A novel hybrid self-adaptive bat algorithm," *The Scientific World Journal*, vol. 2014, 2014, doi: 10.1155/2014/709738.

Author contributions

M.R.R. conceived the study, designed the Advanced Bat Algorithm, contributed to algorithm development and comparative analysis with existing metaheuristics, and drafted the manuscript. Z.A.A. supervised the research and provided expert guidance on mathematical modelling and verification of algorithm equations. F.K.A. advised on the design and execution of comparative analyses between benchmark test function and variance of Bat Algorithm. K.M.Z. contributed to the refined the manuscript. All authors reviewed and approved the final version of the manuscript.

Funding

This research was No Funding.

Declarations

Conflicts of Interest

The authors declare no conflicts of interest.

Competing interests

The authors declare no competing interests.