# BAS-ADAM: An ADAM based Approach to Improve the Performance of Beetle Antennae Search Optimizer

*Abstract*—**In this paper, we propose enhancements to Beetle Antennae Search (BAS) algorithm, called BAS-ADAM, to smoothen the convergence behavior and avoid trapping in local-minima for a highly non-convex objective function. We achieve this by adaptively adjusting the step-size in each iteration using the Adaptive Moment Estimation (ADAM) update rule. The proposed algorithm also increases the convergence rate in a narrow valley. A key feature of the ADAM update rule is the ability to adjust the step-size for each dimension separately instead of using the same step-size. Since ADAM is traditionally used with gradient-based optimization algorithms, therefore we first propose a gradient estimation model without the need to differentiate the objective function. Resultantly, it demonstrates excellent performance and fast convergence rate in searching for the optimum of non-convex functions. The efficiency of the proposed algorithm was tested on three different benchmark problems, including the training of a high-dimensional neural network. The performance is compared with Particle Swarm Optimizer (PSO) and the original BAS algorithm.**

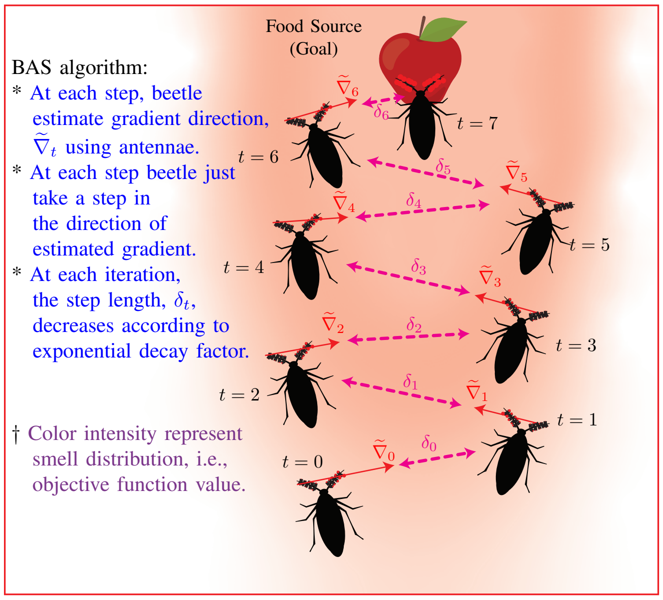
*Index Terms*—**Metaheuristic optimization, Beetle Antennae Search, Neural network, ADAM, Gradient estimation, and Nature-inspired algorithms**

## I. Introduction

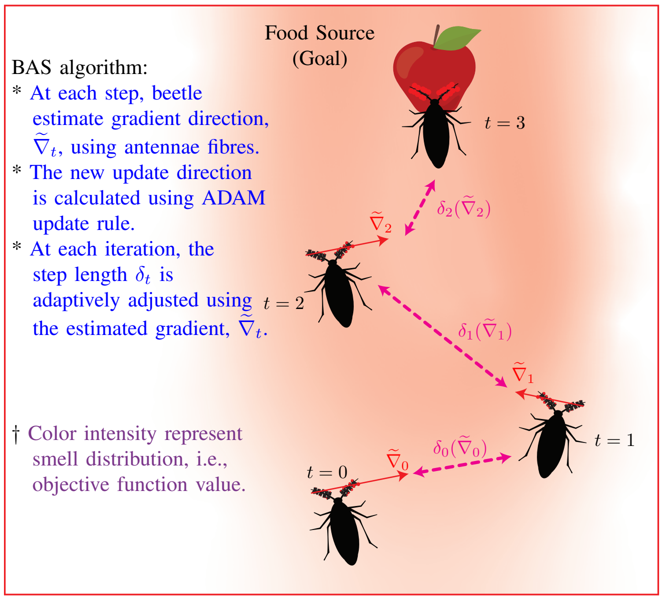
Optimization plays an integral part in the efficient operation of almost all real-world systems [1], [2]. Additionally, with the rise of machine learning in recent years, the development of numerically efficient and robust optimization algorithms has been a topic of great research interest [3]–[5]. Conventional optimization techniques use gradient-based methods to search for the optimum value [6]–[9]. Although these algorithms have proven to be quite stable [10], [11], however, they require the symbolic or numerical computation of the gradient direction. Most of the practical optimization problems are highly nonlinear with multimodal objective functions and non-convex constraints. Numerically evaluating the gradient for such function can be a computationally intensive task [12]. Additionally, the computation of gradient imposes several conditions about continuity and differentiability of the objective function [13]. Such conditions do not hold for the vast majority of the systems, e.g., integer programming [14]. In fact, for the vast majority of the optimization problem, specifically in the control system [15], an accurate model of the system might be unknown in advance and require real-time estimation of parameters [16], [17].

The large majority of the gradient-based optimization is well-suited for computing systems with high computation power. However, their implementation on low-end embedded systems is challenging due to their limited computation power. To overcome such challenges, a new class of optimization algorithm called metaheuristic optimization has gained the attention of researchers [18]–[20]. These algorithms are mostly inspired by natural phenomena and do not require the computation of the gradient of the objective function. These algorithms have been shown in the literature to possess excellent convergence properties and high numerical efficiency. Even for problems in control systems where the model is unknown apriori [21], such algorithms have been employed for the parameter estimation, and tuning of controller gains [22]. One of the significant advantages offered by metaheuristic optimization is a relaxation on the conditions of continuity and differentiability. These algorithms can be effectively employed for solving discrete optimization problems. Additionally, because of their low complexity, they can be efficiently implemented on embedded systems with limited computational power. Given the advantages of metaheuristic algorithms, they have found their applications in several real-world systems [23]–[27].

Almost all the metaheuristic optimization optimization algorithm proposed in literature [28]–[33] have found their inspiration in natural processes. Most of these algorithms have been inspired by the behavior of the animals, whereas others found their inspiration from the biological and chemical systems [34]. For example, biological evolution have given rise to a complete class of metaheuristic algorithm [35], [36] e.g. Genetic Algorithms (GAs) [37], [38]. Other commonly used metaheuristic algorithms inspired by the behavior of living organisms include Particle Swarm Optimization (PSO) [39], [40]. PSO was inspired by the swarming behavior commonly observed in birds and insects. Swarming behavior [41] has been of particular interest to researchers because it was observed that large groups of the birds and insects are able to coordinate by just following a set of straightforward rules. Although each member of the swarm is only aware of its limited surrounding environment, still their combined effort can solve a large-scale, complicated task. This behavior is termed as swarm intelligence [42], [43]. Swarm behavior can be considered as an optimization process in which a group of birds works together to maximize the survivability of the whole swarm. Swarm intelligence has inspired several other algorithms, e.g., Ant Colony Optimization (ACO) [44], which is based on the social behavior of ants. An ant colony can contain millions of ants, but their social behavior and swarm intelligence help them efficiently search for food, thus maximizing the productivity of the colony. Similarly, the Artificial Fish Swarm Algorithm [45] is based on the swarming behavior of fishes and other aquatic lives. Although several nature-inspired algorithm metaheuristic algorithms have been presented in literature, few of them are mentioned here:


Food Source (Goal)
BAS algorithm:
* At each step, beetle estimate gradient direction, $\tilde{\nabla}_t$ using antennae.
* At each step beetle just take a step in the direction of estimated gradient.
* At each iteration, the step length, $\delta_t$, decreases according to exponential decay factor.
† Color intensity represent smell distribution, i.e., objective function value.
$t = 0$
$t = 1$
$t = 2$
$t = 3$
$t = 4$
$t = 5$
$t = 6$
$t = 7$
$\delta_0$ $\delta_1$ $\delta_2$ $\delta_3$ $\delta_4$ $\delta_5$ $\delta_6$
$\tilde{\nabla}_0$ $\tilde{\nabla}_1$ $\tilde{\nabla}_2$ $\tilde{\nabla}_3$ $\tilde{\nabla}_4$ $\tilde{\nabla}_5$ $\tilde{\nabla}_6$

(a)


Food Source (Goal)
BAS algorithm:
* At each step, beetle estimate gradient direction, $\tilde{\nabla}_t$, using antennae fibres.
* The new update direction is calculated using ADAM update rule.
* At each iteration, the step length $\delta_t$ is adaptively adjusted using the estimated gradient, $\tilde{\nabla}_t$.
† Color intensity represent smell distribution, i.e., objective function value.
$t = 0$
$t = 1$
$t = 2$
$t = 3$
$\delta_0(\tilde{\nabla}_0)$ $\delta_1(\tilde{\nabla}_1)$ $\delta_2(\tilde{\nabla}_2)$
$\tilde{\nabla}_0$ $\tilde{\nabla}_1$ $\tilde{\nabla}_2$

(b)

Fig. 1: Illustration of the working of Original BAS and the BAS-ADAM Algorithm proposed in this paper using beetle analogy. (a) Original BAS. (b) BAS-ADAM.

Beetle Antennae Search (BAS) [46], [47], Cuckoo Search [48], [49], Invasive Weed Optimization (IWO) [50], Honey Bee Algorithm (HBA) [51], [52], and Firefly Algorithms (FAs) [53]. These metaheuristic algorithms have been shown to have good performance in real-world practical scenarios. However, their application in machine learning is still limited. The machine learning models are usually very large-scale, having a large number of parameters. The training of these models require searching very high-dimensional spaces, and conventional metaheuristic algorithm does not scale well to such high-dimensions [54].

As already mentioned in the abstract, in this paper, we consider Beetle Antennae Search (BAS) algorithm [46], [47]. Food foraging behavior observed in beetles inspired BAS. Beetle behavior is of particular research interest because unlike other insects, beetles usually do not work in a swarm and have the ability to search for food individually. Beetles can search for food using its antennae and sensitive sense of smell. Besides, the beetle antennae have several small fibers, and by sensing the difference of smell at each of the fiber, the beetle can develop a map of smell intensity of surrounding. This map helps in finding the direction of maximum smell change. Inspired by this concept, an estimation model can be developed, which helps in approximating the gradient direction. The estimation of an approximated gradient is a key feature of BAS, which distinguishes it from another metaheuristic algorithm. Since its introduction, BAS has found its application in several real-world systems [17], [55]–[64]. The working of the original BAS can be described like this; at each iteration, the value of the objective function is computed at each antennae fiber location, a vector is drawn from the fiber with the lowest value toward the fiber with the highest value, the vector represents the direction of the approximated gradient. The approximated gradient is then used to update the location of the beetle. The working of the original BAS is shown in Fig. 1(a).

One of the key limitations of the BAS algorithm is the selection of a suitable value for initial step size and proper step size annealing. Careful tuning of these parameters is required to achieve a fast convergence rate and stable performance. The previous works on BAS [46], [47] use exponential decay and power-law annealing. However, these methods are not adaptive to the shape of the objective function, i.e., the step size change at the same rate. This behavior can results in slow convergence if the objective function has a narrow valley. From optimization literature, we know that for such objective functions, the gradient-based method, especially Stochastic Gradient Descent (SGD), shows poor performance since the estimated point can bounce back and forth between walls of the valley during iterations [65]. It increases the number of iteration required to reach the optimal point. To improve the convergence rate, several modifications to the SGD algorithm have been proposed in the literature, which adaptively adjusts the step size and direction. On such popular algorithm is Adaptive Moment Estimation (ADAM) [6]. The ADAM algorithm updates the value of step size based on all the past values of gradient and square of gradients. The idea of ADAM is borrowed from momentum SGD [65] but show more robustness and stability near-optimal points. For this reason, we choose the ADAM algorithm to adaptively update the step size during runtime instead of manually adjusting the parameters in the beginning. The working of the BAS-ADAM is shown in Fig. 1(b). It can be seen that it shows much faster convergence as compared to the original BAS.

To demonstrate the fast convergence, efficiency, and effectiveness of the proposed BAS-ADAM algorithm, we performed several sets of experiments and compared its performance with other benchmark metaheuristic algorithm. In the literature on metaheuristic algorithms, the performance of algorithms is verified using low dimensional benchmark functions [51]. In this paper, we took the rigor of the testing process one step further and used the BAS-ADAM to train a hidden layer neural network with nonlinear activation functions. Additionally, we used two other benchmark optimization problems, which were similar to other works in metaheuristic literature. We solved these benchmark optimization problems using the PSO, a state of the art metaheuristic algorithm, original BAS, and BAS-ADAM. It is shown in results that the BAS-ADAM show superior performance as compared to original BAS and PSO in solving the benchmark optimization problems, including the training of neural networks.

The rest of the paper is organized as follow; Section II present the details of the BAS-ADAM algorithm, Section III present the benchmark optimization problems we used in this paper, Section IV present the experimental results, and Section V concludes the paper.

## II. BAS-ADAM Algorithm

In this section, we will describe the technical details and mathematical formulation of the BAS-ADAM algorithm and outline its steps.

### A. Beetle Antennae Search

Here we will describe the BAS algorithm [46] and present the gradient estimation model. Consider the optimization problem:

$$\max_{\mathbf{x}} \ f(\mathbf{x}), \tag{1}$$

where $f(\mathbf{x}) \in \mathbb{R}^1$ is the objective function and $\mathbf{x} \in \mathbb{R}^n$ is the domain of objective function. BAS algorithm treats $f(\mathbf{x})$ as the smell intensity distribution in an $n$-dimensional space and tries to search for a point $\mathbf{x}^*$ at which the value of smell intensity is maximum. The point $\mathbf{x}^*$ corresponds to the position of the food source, and the objective of the beetle is to search this point. If the beetle is standing at point $\mathbf{x}_t$ at time instant $t$, then to find the direction of the food source, the beetle measure the intensity of smell at each of its antennae fibers. For mathematical formulation, consider the beetle having a total of $m$ antennae fibers. The location of each fiber is represented by a normally distributed normalized random position vector $\overrightarrow{\mathbf{b}}$ relative to the beetle position $\mathbf{x}_t$. The set of $m$ position vectors can be represented as $\mathcal{B} = \{\overrightarrow{\mathbf{b}}_1, \overrightarrow{\mathbf{b}}_2, ..., \overrightarrow{\mathbf{b}}_m\} \subset \mathbb{R}^n$. Using the set $\mathcal{B}$, the location of each antennae fiber can be written as

$$\begin{aligned}
\mathbf{x}^{\mathbf{b}_1} &= \mathbf{x}_t + \eta_t \overrightarrow{\mathbf{b}}_1, \\
\mathbf{x}^{\mathbf{b}_2} &= \mathbf{x}_t + \eta_t \overrightarrow{\mathbf{b}}_2, \\
&\vdots \ \ \vdots \ \ \vdots \\
\mathbf{x}^{\mathbf{b}_m} &= \mathbf{x}_t + \eta_t \overrightarrow{\mathbf{b}}_m,
\end{aligned} \tag{2}$$

where $\eta_t$ represents the length of the beetle antennae. The actual location of each antennae fibre can be written as a set $\mathcal{X} = \{\mathbf{x}^{\mathbf{b}_1}, \mathbf{x}^{\mathbf{b}_2}, ..., \mathbf{x}^{\mathbf{b}_m}\}$.

After creating the set $\mathcal{X}$ of the antennae fiber locations, we evaluate the objective function $f$ at each of its points. Thus creating a set $\mathcal{F}$ of the objective values

$$\mathcal{F} = \{f(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\} = \{f(\mathbf{x}^{\mathbf{b}_1}), f(\mathbf{x}^{\mathbf{b}_2}), ..., f(\mathbf{x}^{\mathbf{b}_m})\}.$$

### B. Gradient Estimation

Now we will formulate the strategy to estimate the gradient direction using the element of setting $\mathcal{X}$ and $\mathcal{F}$. Note that there is a one-to-one correspondence between the elements of $\mathcal{X}$ and $\mathcal{F}$. Based on this correspondence, we first create a subset $\mathcal{X}_l$ from the set $\mathcal{X}$ corresponding to the $k(< m)$ lowest value in set $\mathcal{F}$. Similarly, we create a set $\mathcal{X}_h$ corresponding to the $k$ highest value in the set $\mathcal{F}$. The value of $k$ controls the robustness of the gradient estimation. In mathematical formulation, we create the following sets

$$\begin{aligned}
\mathcal{F}_l &= \{f : f \subset \mathcal{F} \wedge f \in \min_k \mathcal{F}\}, \\
\mathcal{F}_h &= \{f : f \subset \mathcal{F} \wedge f \in \max_k \mathcal{F}\},
\end{aligned} \tag{3}$$

where notation $\min_k$ and $\max_k$ are used to represent $k$ minimum and maximum value from the corresponding sets. Then we create the sets $\mathcal{X}_l$ and $\mathcal{X}_h$ using the values in sets $\mathcal{F}_l$ and $\mathcal{F}_h$

$$\begin{aligned}
\mathcal{X}_l &= \{\mathbf{x} : \mathbf{x} \subset \mathcal{X} \wedge f(\mathbf{x}) \in \mathcal{F}_l\}, \\
\mathcal{X}_h &= \{\mathbf{x} : \mathbf{x} \subset \mathcal{X} \wedge f(\mathbf{x}) \in \mathcal{F}_h\}.
\end{aligned} \tag{4}$$

For a beetle analogy, $\mathcal{X}_l$ and $\mathcal{X}_h$ are sets of $k$ antenna fiber locations, where the smell intensity is minimum and maximum, respectively. We calculate the centroid of sets $\mathcal{X}_l$ and $\mathcal{X}_h$ as

$$\begin{aligned}
\mathbf{x}_l &= \sum_{\mathbf{x} \in \mathcal{X}_l} \mathbf{x}/k \\
\mathbf{x}_h &= \sum_{\mathbf{x} \in \mathcal{X}_h} \mathbf{x}/k.
\end{aligned} \tag{5}$$

The vectors $\mathbf{x}_l \in \mathbb{R}^n$ and $\mathbf{x}_h \in \mathbb{R}^n$ calculated in 5 corresponds to locations around antennae fibres at which the value of the objective function is minimum and maximum respectively. A vector starting from point $\mathbf{x}_l$ to $\mathbf{x}_h$ represents the direction of maximum value change, i.e., gradient direction. Therefore the estimated gradient direction can be denoted as

$$\widetilde{\nabla}_t = \mathbf{x}_h - \mathbf{x}_l, \tag{6}$$

where $\widetilde{\nabla}_t$ represent the estimated gradient direction on time instant $t$. We will now use the estimated gradient direction to update the location of beetle from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$ using ADAM update rule in next subsection.

### C. ADAM Update Rule

Based on the value of estimated gradient $\widetilde{\nabla}_t$, we can formulate an update rule as

$$\mathbf{x}_{new} = \mathbf{x}_t + \delta_t(\widetilde{\nabla}_t), \tag{7}$$

where $\delta_t$ is the step size and proportional to the distance between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. The step length $\delta_t$ is a function of estimated gradient $\widetilde{\nabla}_t$ and we defined it according to the ADAM update rule as

$$\delta_t(\widetilde{\nabla}_t) = \delta_0 \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \tag{8}$$

where $\delta_0$ is the initial step size, and its value is constant. $\hat{m}_t$ and $\hat{v}_t$ represent the first and second moment of the estimated gradient $\widetilde{\nabla}_t$ and its value is calculated as follow,

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1 - \beta_1)\text{sign}(f(\mathbf{x}_r) - f(\mathbf{x}_l))\widetilde{\nabla}_t \\
v_t &= \beta_2 v_{t-1} + (1 - \beta_2)\widetilde{\nabla}_t^2.
\end{aligned} \tag{9}$$

The factor of $\text{sign}(f(\mathbf{x}_r) - f(\mathbf{x}_l))$ is multiplied to make sure that $\widetilde{\nabla}_t$ always point toward the direction of increasing objective function value. $(.)^2$ functions represent a piecewise square operation of the elements of vector $\widetilde{\nabla}_t$. As noted in ADAM literature, the values of $m_t$ and $v_t$ are biased toward zero. The correction factor for this biasness can be applied as

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{10}$$

By putting the values of $\hat{m}_t$ and $\hat{v}_t$ in (8), we can calculate the value of $\delta_t(\widetilde{\nabla}_t)$ and the value of $\mathbf{x}_{new}$. For further increasing the efficiency of the algorithm, the value of $\mathbf{x}_{t+1}$ is only updated if there is an improvement in the value of objective function, i.e., $f(\mathbf{x}_{t+1}) > f(\mathbf{x}_t)$. For this we keep track of the best solution $\mathbf{x}_{best}$ and corresponding objective function value $f_{best} = f(\mathbf{x}_{best})$. Therefore, the final update rule for the value of $\mathbf{x}_{t+1}$ can be written as

$$\mathbf{x}_{t+1} = \begin{cases} \mathbf{x}_{new}, & \text{if } f(\mathbf{x}_{new}) > f_{best}, \\ \mathbf{x}_t, & \text{otherwise.} \end{cases} \tag{11}$$

The presented BAS-ADAM algorithm is given in Algorithm 1; it can be summarized as

1) Begin with an random starting point $\mathbf{x}_0$.
2) Generate a set of $m$ normalized random position vectors, $\mathcal{B} \subset \mathbb{R}^n$.
3) Calculate the set for location of antennae fibres, $\mathcal{X} \subset \mathbb{R}^n$.
4) Calculate the set of the objective function values $\mathcal{F} = \{f(\mathbf{x}^{\mathbf{b}_1}), f(\mathbf{x}^{\mathbf{b}_2}), ..., f(\mathbf{x}^{\mathbf{b}_m})\}$.
5) Calculate the sets $\mathcal{F}_l$ and $\mathcal{F}_h$ of $k(< m)$ lowest and highest values from set $\mathcal{F}$ using (3).
6) Calculate the set $\mathcal{X}_l \subset \mathcal{X}$ and $\mathcal{X}_h \subset \mathcal{X}$ according to the values in $\mathcal{F}_l$ and $\mathcal{F}_h$.
7) Calculate the centroid $\mathbf{x}_l$ and $\mathbf{x}_h$ of the set $\mathcal{X}_l$ and $\mathcal{X}_h$.
8) Calculate the value of estimated gradient $\widetilde{\nabla}_t$ using (6).
9) Calculate the first and second moments of estimated gradient using (9).
10) Calculate the unbiased moments using (10).
11) Determine candidate for update point $\mathbf{x}_{new}$ using (7).
12) The value of $\mathbf{x}_{t+1}$ is only updated if there is improvement in objective function value at $\mathbf{x}_{new}$.
13) If $t < t_{max}$, where $t_{max}$ is maximum number of iterations, then go back to step 2.

## III. Validation Methodology

This section presents the validation methodology used to test the efficiency, convergence, and efficacy of the proposed BAS-ADAM algorithm. We selected a set of three benchmark problems. The first is to search for the optimum value of Michalewicz function [66]. The second is a linear regression optimization problem, and the 3rd problem involves training of a single hidden layer neural network.

### A. Michalewicz Function

Michalewicz function [66] is a highly nonlinear, multimodal, non-convex function. This problem involves searching for the minima of the Michalewicz function. Since it is a multimodal function, there is a high chance that the searching algorithm gets trap in local minima. Therefore it is heavily used in literature to test the performance of metaheuristic algorithms. Michalewicz function is defined as

**Algorithm 1:** BAS-ADAM Optimizer

```
Input: An objective function $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$, and
       values of parameters: $\mathbf{x}_0$, $d_0$, $\eta$, $\gamma$, $\beta_1$, $\beta_2$, and
       $t_{stop}$.
Output: Optimal solution $\mathbf{x}^*$ to the problem:
        $\max_{\mathbf{x}}\ f(\mathbf{x})$ and optimal value $f(\mathbf{x}^*)$.
$t \leftarrow 1$
$\mathbf{x}_{best} \leftarrow \mathbf{x}_0$
$f_{best} \leftarrow f(\mathbf{x}_{best})$
while $t < t_{stop}$ do
    Generate a set of $m$ random direction vectors,
      $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m : \mathbf{b}_i \in \mathbb{R}^n \wedge ||\mathbf{b}_i||_2 = 1\}$.
    Calculate the set, $\mathcal{X} = \{\mathbf{x}^{\mathbf{b}_1}, \mathbf{x}^{\mathbf{b}_2}, ..., \mathbf{x}^{\mathbf{b}_m}\}$, of the
      beetle antennae fibre location according to (2).
    Calculate the set, $\mathcal{F} = \{f(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$, of
      objective function values at location in set $\mathcal{X}$.
    Select $k$ lowest and highest values from set $\mathcal{F}$ to
      create the set $\mathcal{F}_l$ and $\mathcal{F}_h$ using (3).
    Using $\mathcal{F}_l$ and $\mathcal{F}_h$, create set $\mathcal{X}_l$ and $\mathcal{X}_h$ according
      to (4).
    Calculate the centroids $\mathbf{x}_l$ and $\mathbf{x}_h$ of the elements
      of set $\mathcal{X}_l$ and $\mathcal{X}_h$ using (5).
    Estimate the gradient direction, $\widetilde{\nabla}_t$, using (6).
    Calculate the first moment $m_t$ and second
      moments $v_t$ of the estimated gradient using (9).
    Correct the biasness in the calculated moments
      using (10).
    Calculate a condidate for update-value $\mathbf{x}_{new}$ using
      (7).
    if $f(\mathbf{x}_{new}) > f_{best}$ then
        $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_{new}$
        $f_{best} \leftarrow f(\mathbf{x}_{new})$
        $x_{best} \leftarrow \mathbf{x}_{new}$
    else
        $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t$
        $f_{best} \leftarrow f_{best}$
        $x_{best} \leftarrow x_{best}$
    end
    t ← t + 1;
end
```

$$\mathcal{M}(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i) \sin\Big(\frac{ix_i^2}{pi}\Big)^{2m}, \tag{12}$$

where $d$ is the dimension of input $\mathbf{x}$ to the Michalewicz function, i.e., $\mathbf{x} \in \mathbb{R}^d$. The value of $m$ is directly proportional to the narrowness of the valleys for this function. For a given value of $d$, the Michalewicz function have a total of $d!$ optimum in range $[0, \pi]^d$. Plot of the function in the range of interest is shown in Fig. 2. Since our original algorithm was designed according to maximization problem (1) and this is a minimization problem, we can transform it into the following maximization problem

$$\mathbf{x}^* = \min_{\mathbf{x}}\ \mathcal{M}(\mathbf{x})\ \equiv\ \max_{\mathbf{x}}\ -\mathcal{M}(\mathbf{x}). \tag{13}$$

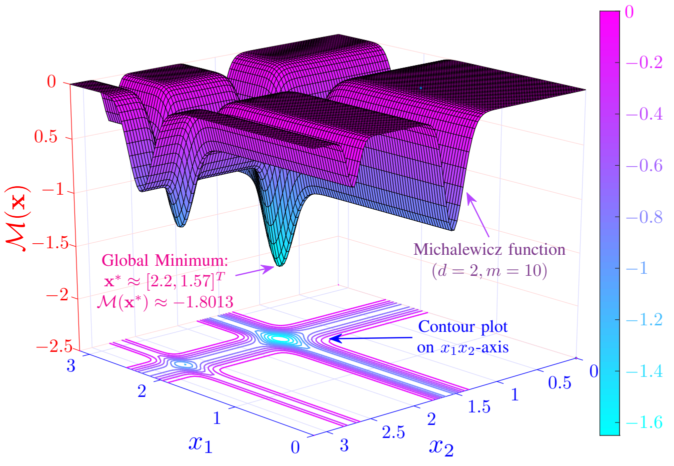


Fig. 2: Plot of the Michalewicz function selected as validation problem formulated in section III-B. It is a non-convex function with several local optimum. Its theoritically known optimal solution is shown on the graph.

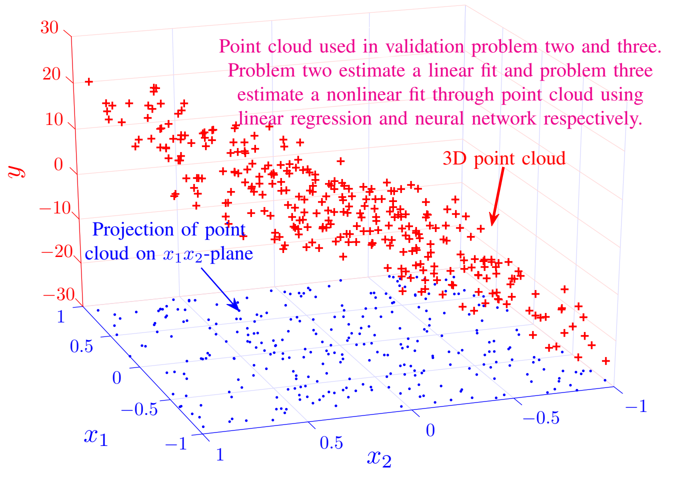


Fig. 3: Point cloud used in validation problems problem two and three in Sections III-B and III-C respectively. $x_1, x_2$ are independent variables and $y$ is dependent variables.

We used $d = 2$ and $m = 10$ in our experiments. Value of $d = 2$ allow us to easily visualize the convergence performance using 2D contour plots. The results are presented in Section IV.

## B. Linear Regression

For Michalewicz function, the objective function was fixed, and the optimum value is already known in the literature. However, our second problem involves solving a linear regression optimization problem in which the objective function is defined using randomly generated data points. Since linear regression is essentially the least-square optimization problem; therefore, the objective function is convex. The linear regression optimization problem can be defined like this: given a vector of $n$ independent variables and one dependent variable $([x_1, x_2, ..., x_n], y)$, estimate the parameter $\{\theta_0, \theta_1, \theta_2, ..., \theta_n\}$ which results in best fit to following equation. The criteria for the best fit is usually defined in the least square sense

$$y = \theta_n x_n + \theta_{n-1} x_{n-1} + ... + \theta_1 x_1 + \theta_0 = \sum_{i=0}^{n} \theta_i x_i, \quad (14)$$

where $x_0 = 1$ is used for ease of mathematical notation. The above equation can be simplified using matrix operations

$$y = \boldsymbol{\theta}^T \mathbf{x},$$

where $\mathbf{x} = [x_0, x_1, x_2, ..., x_n]^T \in \mathbb{R}^{n+1}$ and $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, ..., \theta_n] \in \mathbb{R}^{n+1}$.

If we are provided with a set of $m$ datapoints for independent and dependent variables, i.e., $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_{m+1}\}$ and $\{y_1, y_2, ...y_m\}$ respectively, we can define a vector of residual error values as

$$\mathcal{R}(\boldsymbol{\theta}) = [y_1 - \boldsymbol{\theta}^T \mathbf{x}_1,\ y_2 - \boldsymbol{\theta}^T \mathbf{x}_2,\ ...,\ y_m - \boldsymbol{\theta}^T \mathbf{x}_m]^T,$$

based on this vector, we can define a cost function based on square-sum of the residual values as

$$\mathcal{C}(\boldsymbol{\theta}) = ||\mathcal{R}(\boldsymbol{\theta})||_2, \\ = \sum_{i=0}^{m} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2.$$

Therefore, linear regression can be written as the following optimization problem

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \ \mathcal{C}(\boldsymbol{\theta}) \ \equiv \ \max_{\boldsymbol{\theta}} \ -\mathcal{C}(\boldsymbol{\theta}). \quad (15)$$

During experiments, we used $n = 2$, $m = 300$, i.e., 300 samples points. The randomly generated dataset is shown in Fig. 3. It can be seen from (1) that for $n = 2$, three parameters need to be estimated, i.e., $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3] \in \mathbb{R}^3$. In Section IV we will present the numerical results for this problem.

## C. Neural Network Training

As the third benchmark problem, we choose the training of a hidden layer neural network with nonlinear activation functions. Most of the metaheuristic algorithms show poor performance when it comes to the training of a machine learning model because these models are highly nonlinear and contain a large number of parameters. Without the information about the gradient direction, searching a high dimensional space becomes very challenging. Since our metaheuristic algorithm involves a rough estimation of the gradient direction, it is expected to show better performance for high dimensional search spaces. Additionally, we used the ADAM algorithm to adjust step size adaptively, therefore the proposed algorithm also does an excellent job in avoiding the local minimum.

It is known from machine learning literature that hidden layer neural networks are excellent in end to end learning [67], i.e., they can model an unknown process by just observing the input-output data points. They can model an arbitrary nonlinear process without needing an apriori mathematical model. Therefore we used the point cloud of Fig. 3 same as linear regression. However, this time, instead of using a linear model of (14), we used a neural network. It is expected that the neural network will provide a better fit to the point cloud and therefore produce a lower value for the cost function. The architecture of the neural network we used in this paper has two inputs, ten hidden neurons, and one output. Now we will derive the objective function for this optimization problem. Let $\mathbf{x} \in \mathbb{R}^n$ and $y \in \mathbb{R}^1$ be the input and output of the neural network, respectively. The feedforward equation of the neural network can be written as

$$\mathbf{o_1} = \phi(\mathbf{x}^T\mathbf{W_1} + \mathbf{b_1}),$$
$$y = \mathbf{o_1}\mathbf{W_2} + \mathbf{b_2}, \tag{16}$$

where $\mathbf{o_1} \in \mathbb{R}^k$ is output of the hidden layer. $\phi(.)$ is an activation function, in our experiments we choose hyperbolic tangent sigmoid as the activation function, i.e., $\phi(.) = \text{tansig}(.)$. The matrices $\mathbf{W_1} \in \mathbb{R}^{n\times k}$ and $\mathbf{W_2} \in \mathbb{R}^{k\times 1}$ are trainable weights of the hidden layer and output layer respectively. Similarly, $\mathbf{b_1} \in \mathbb{R}^k$ and $\mathbf{b_2} \in \mathbb{R}$ are the biases of hidden and output layer. These matrices can be written as follow

$$\mathbf{W_1} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} & \dots & w_{1k}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{13}^{(1)} & \dots & w_{2k}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n1}^{(1)} & w_{n2}^{(1)} & w_{13}^{(1)} & \dots & w_{nk}^{(1)} \end{bmatrix},$$

$$\mathbf{W_2} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} & w_{31}^{(2)} & \dots & w_{k1}^{(2)} \end{bmatrix}^T,$$

and,

$$\mathbf{b_1} = \begin{bmatrix} b_1^{(1)} & b_2^{(1)} & b_3^{(1)} & \dots & b_k^{(1)} \end{bmatrix}, \quad \mathbf{b_2} = \begin{bmatrix} b_1^{(2)} \end{bmatrix}.$$

Now let us consider a set of $m$ samples of input and output data points. Similar to the approach we used in III-B, we can define the following cost function for training the neural network

$$\mathcal{C}(\mathbf{W_1}, \mathbf{W_2}, \mathbf{b_1}, \mathbf{b_2}) = \sum_{i=0}^{m} (y_i - \phi(\mathbf{x}_1^T\mathbf{W_1} + \mathbf{b_1})\mathbf{W_2} - \mathbf{b_2})^2.$$

Using the cost function $\mathcal{C}(\mathbf{W_1}, \mathbf{W_2}, \mathbf{b_1}, \mathbf{b_2})$ we can formulate the training of the neural network as the following optimization problem

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \ \mathcal{C}(\boldsymbol{\theta}) \ \equiv \ \max_{\boldsymbol{\theta}} -\mathcal{C}(\boldsymbol{\theta}). \tag{17}$$

In our experiments we used a neural network with 10 hidden neurons, 2 inputs and 1 output. There are a total of 41 trainable parameters in such a neural network. In summary, the problem 3 have these final values for the optimization problem: $n = 2$, $k = 10$, $\phi(.) = \text{tansig}(.)$ and $m = 300$, i.e., 300 training samples. The results are presented in Section IV.

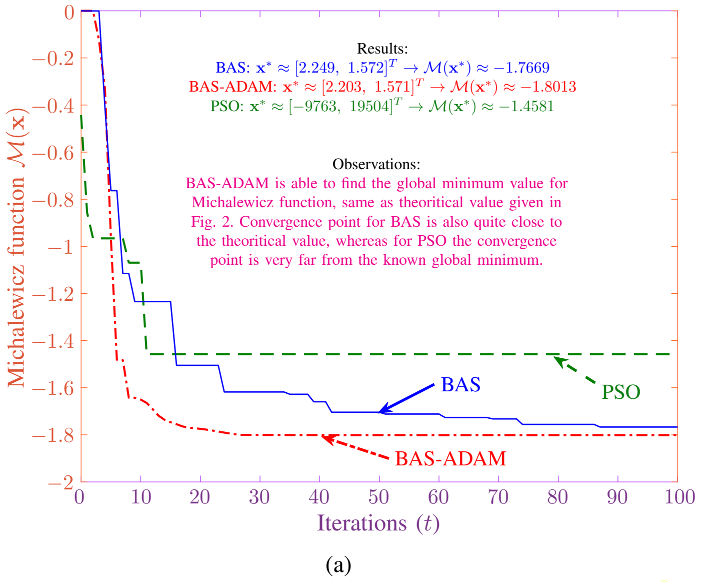
## IV. Results & Discussion

Now we will present and discuss the experimental results for the validation problems presented in Sections III-A, III-B and III-C. Additionally, we compared the performance of the proposed BAS-ADAM algorithm with the PSO and original BAS algorithm. For the PSO algorithm, we used its implementation provided by MATLAB's global optimization toolbox [68], whereas for BAS-ADAM and original BAS, we wrote the code in MATLAB.

### A. Solution: Michalewicz Function

For Michalewicz function, we used $d = 2$ and $m = 10$ in our experimentation. From literature [66] it is known that this function have a global minimum at $\mathbf{x}^* \approx [2.20, \ 1.57]$ and minimum value of the function is $\mathcal{M}(\mathbf{x}^*) \approx -1.8013$.

The performance of the BAS-ADAM, along with PSO and original BAS, is shown in Fig. 4. The Fig. 4(a) shows the decrease in value of residual error with increase in iterations.



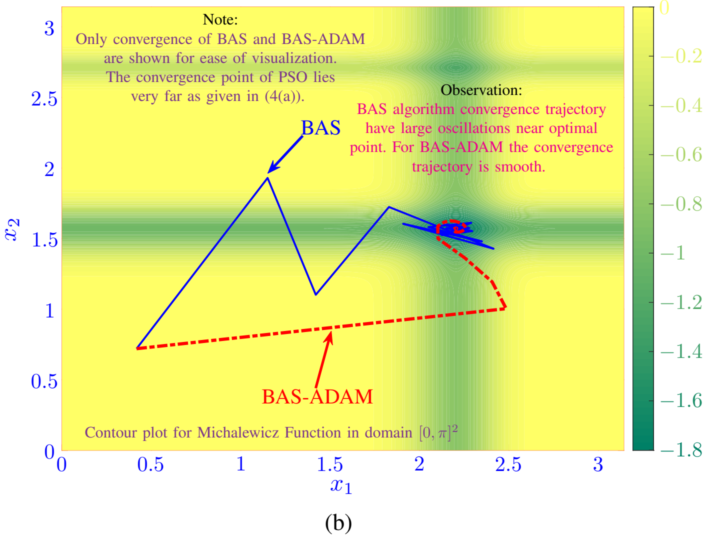
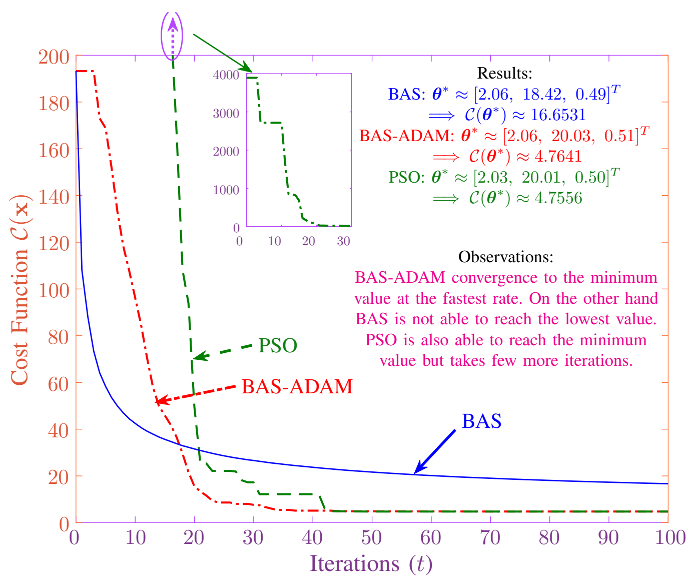
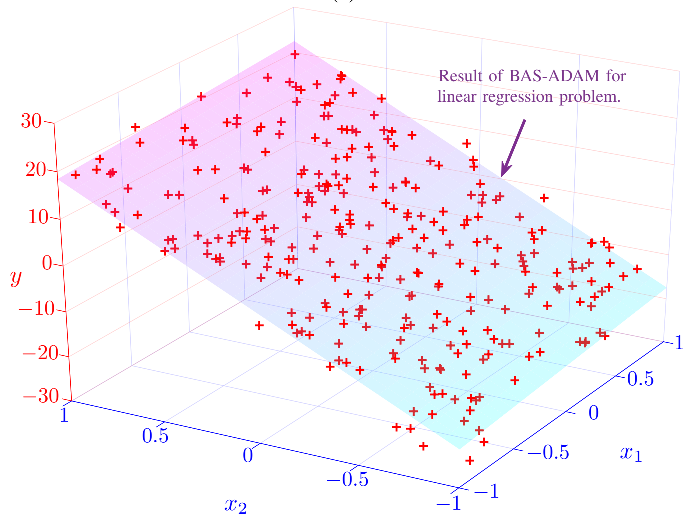


Fig. 4: Performance of optimization algorithms for Michalewicz function minimization given in Section III-A.

The plot of error residual clearly shows that the BAS-ADAM shows the best convergence performance as compared to the other two optimization algorithms. The optimal point and minimum value reached by the BAS-ADAM comes quite close to the one given in the literature. BAS-ADAM is able to reach the theoretical minimum value of $-1.8013$, whereas, for original BAS, the final value is $-1.7669$. For PSO, the final value is $-1.4581$, and the final point is $[-9763, \ 19504]^T$, which is a local optimum, very far from the know global optimum $[2.20, \ 1.57]$. Similarly, Fig. 4(b) shows the convergence trajectory of BAS-ADAM and original BAS. This plot shows that the ADAM-BAS is able to reach the optimum point quite smoothly, whereas, for original BAS, the convergence trajectory shows many vibrations. It demonstrates the robustness of the BAS-ADAM optimizer. The convergence trajectory for PSO is not included for the purpose of visualization since its final point lies far off the graph.



(a)



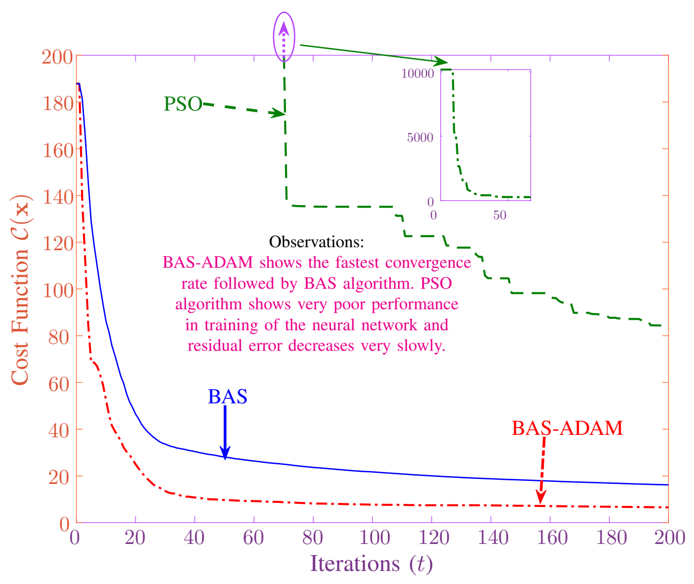
(b)

Fig. 5: Performance of optimization algorithms for linear regression validation problem given in Section III-B.

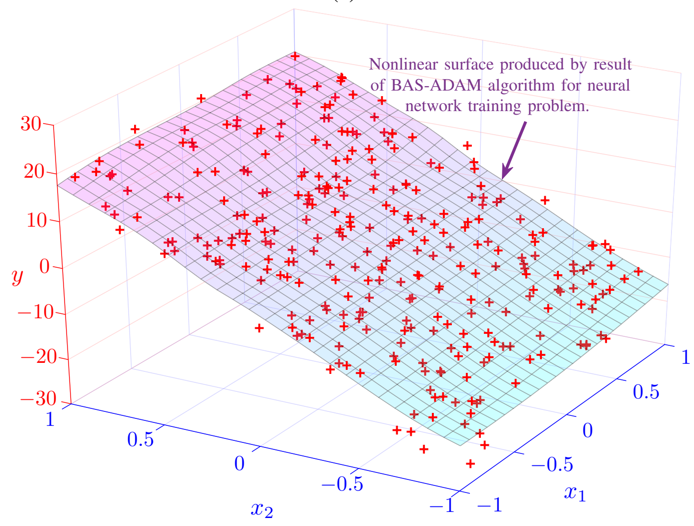
### B. Solution: Linear Regression

The second validation problem involves solving the linear regression optimization problem given in (15). It is a convex optimization problem with a single global optimum and no local optimum. We tuned the parameters for BAS and BAS-ADAM until we reached the best performance.

The experimental results are shown in Fig. 5. The plot in Fig. 5(a) shows the decrease in value of error residual with iterations. The plot shows that BAS-ADAM shows the final rate of convergence, followed by PSO and original BAS. However, the final reached by original BAS, 16.6531 is much higher as compared to the values reached by BAS-ADAM and PSO. BAS-ADAM took about 33 iterations to reach near the optimum point, while PSO took around 43 iterations. These results again prove the faster rate of convergence and superior performance of the BAS-ADAM algorithm. Using the final estimated parameters from BAS-ADAM, we can plot a linear fit through the point cloud of Fig. 3. The fitted plane is shown in Fig. 5(b).



(a)



(b)

Fig. 6: Performance of optimization algorithms for neural network validation problem given in Section III-C.

### C. Solution: Neural Network

Among the three validation problems, the training of a hidden layer neural network is the most challenging one. As mentioned in III-C, we used a neural network with ten hidden neurons to model the point cloud in Fig. 3. The topology of the neural network includes two inputs, one fully connected hidden layer with ten neurons and one output.

Fig. 6 shows the experimental results for training the neural network using the three optimization algorithms. The plot of error residuals is shown in Fig. 6(a). According to the plot, the BAS-ADAM shows the best performance, followed by BAS and PSO. The BAS-ADAM is able to reach the objective function value of around 6.5, i.e., $\mathcal{C}(\mathbf{W}_1^*, \mathbf{W}_2^*, \mathbf{b}_1^*, \mathbf{b}_2^*) \approx 6.5$. Whereas original BAS and PSO get stuck in a local minimum and reach a minimum value of around 16 and 80. It is also worth noting that although the problem of linear regression and neural network are modeling the same point cloud using linear regression and neural network, respectively. The reason for the poor performance of PSO and original BAS is the high dimensionality of the objective functions. The neural has a total of 41 trainable parameters. The huge search space decreases the efficiency of PSO, original BAS, and other similar metaheuristic optimization algorithms. Fig. 6(b) shows a nonlinear surface to model the point cloud of Fig. 3. The surface is drawn using the final estimated parameters for the BAS-ADAM algorithm.

## V. Conclusion

In this paper, we presented a robust nature-inspired metaheuristic algorithm, called BAS-ADAM. The proposed algorithm is inspired by the BAS algorithm and improves its performance for the objective function, having very steep valleys. We achieve this by using the adaptive moment estimation (ADAM) technique widely applied in gradient-based optimization algorithms. The proposed algorithm estimated a gradient direction at each iteration using the antennae fibers of the beetle. The estimated gradient is then used as input to the ADAM algorithm, which computes its first and second moments. These moments are then used to adjust the step size for the next iterations. This technique offers a significant advantage as compared to original BAS because it automatically adapts the step size for each dimension according to the shape of the objective function. Therefore the BAS-ADAM shows much smoother convergence performance near the optimum points, as shown in the experiments. For original BAS, the step size is calculated using exponential decay rule and independent of the shape of the objective function. It is demonstrated through extensive experiments that the proposed algorithm offers fast and robust convergence as compared to other metaheuristic optimization algorithms. We selected three benchmark optimization problems to test the performance of the proposed algorithm. We repeated the same experiments with original BAS and PSO optimizers to present the comparative results. The experimental results show that, on average, BAS-ADAM takes fewer iterations and able to reach better optimum values as compared to original BAS and PSO. Additionally, we trained a hidden layer neural network, which shows the potential of the proposed algorithm in real-time machine learning applications.

## References

[1] H. Wang, P. X. Liu, X. Xie, X. Liu, T. Hayat, F. E. Alsaadi, Adaptive fuzzy asymptotical tracking control of nonlinear systems with unmodeled dynamics and quantized actuator, Information Sciences (2018).

[2] C. Yang, J. Luo, C. Liu, M. Li, S.-L. Dai, Haptics electromyogrphy perception and learning enhanced intelligence for teleoperated robot, IEEE Transactions on Automation Science and Engineering (2018).

[3] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, Siam Review 60 (2) (2018) 223–311.

[4] S. Theodoridis, Machine learning: a Bayesian and optimization perspective, Academic Press, 2015.

[5] T. Liu, B. Tian, Y. Ai, L. Li, D. Cao, F.-Y. Wang, Parallel reinforcement learning: a framework and case study, IEEE/CAA Journal of Automatica Sinica 5 (4) (2018) 827–835.

[6] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[7] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[8] Y. Shi, Y. Zhang, Solving future equation systems using integral-type error function and using twice znn formula with disturbances suppressed, Journal of the Franklin Institute 356 (4) (2019) 2130–2152.

[9] L. Xiao, S. Li, F.-J. Lin, Z. Tan, A. H. Khan, Zeroing neural dynamics for control design: comprehensive analysis on stability, robustness, and convergence speed, IEEE Transactions on Industrial Informatics 15 (5) (2018) 2605–2616.

[10] Y. Zhang, Z. Qi, B. Qiu, M. Yang, M. Xiao, Zeroing neural dynamics and models for various time-varying problems solving with zlsf models as minimization-type and euler-type special cases [research frontier], IEEE Computational Intelligence Magazine 14 (3) (2019) 52–60.

[11] Y. Zhang, Z. Qi, M. Yang, J. Guo, H. Huang, Step-width theoretics and numerics of four-point general dtzn model for future minimization using jury stability criterion, Neurocomputing 357 (2019) 231–239.

[12] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Advances in neural information processing systems, 2013, pp. 315–323.

[13] J. D. Lee, M. Simchowitz, M. I. Jordan, B. Recht, Gradient descent only converges to minimizers, in: Conference on Learning Theory, 2016, pp. 1246–1257.

[14] W.-N. Chen, J. Zhang, H. S. Chung, W.-L. Zhong, W.-G. Wu, Y.-H. Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Transactions on evolutionary computation 14 (2) (2010) 278–300.

[15] S. Ling, H. Wang, P. X. Liu, Adaptive fuzzy dynamic surface control of flexible-joint robot systems with input saturation, IEEE/CAA Journal of Automatica Sinica 6 (1) (2019) 97–107.

[16] H. Nguyen-Xuan, G. Liu, C. a. Thai-Hoang, T. Nguyen-Thoi, An edge-based smoothed finite element method (es-fem) with stabilized discrete shear gap technique for analysis of reissner–mindlin plates, Computer Methods in Applied Mechanics and Engineering 199 (9-12) (2010) 471–489.

[17] A. H. Khan, S. Li, X. Luo, Obstacle avoidance and tracking control of redundant robotic manipulator: An rnn based metaheuristic approach, IEEE Transactions on Industrial Informatics (2019).

[18] Y. Zhou, L. Kong, Z. Wu, S. Liu, Y. Cai, Y. Liu, Ensemble of multi-objective metaheuristic algorithms for multi-objective unconstrained binary quadratic programming problem, Applied Soft Computing 81 (2019) 105485.

[19] J. A. Parejo, A. Ruiz-Cortés, S. Lozano, P. Fernandez, Metaheuristic optimization frameworks: a survey and benchmarking, Soft Computing 16 (3) (2012) 527–561.

[20] C. Blum, A. Roli, M. Sampels, Hybrid metaheuristics: an emerging approach to optimization, Vol. 114, Springer, 2008.

[21] H. Wang, P. X. Liu, X. Zhao, X. Liu, Adaptive fuzzy finite-time control of nonlinear systems with actuator faults, IEEE transactions on cybernetics (2019).

[22] O. Roeva, T. Slavov, Pid controller tuning based on metaheuristic algorithms for bioprocess control, Biotechnology & Biotechnological Equipment 26 (5) (2012) 3267–3277.

[23] A. Song, W.-N. Chen, T. Gu, H. Yuan, S. Kwong, J. Zhang, Distributed virtual network embedding system with historical archives and set-based particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2019).

[24] C. Yang, G. Peng, Y. Li, R. Cui, L. Cheng, Z. Li, Neural networks enhanced adaptive admittance control of optimized robot–environment interaction, IEEE transactions on cybernetics 49 (7) (2018) 2568–2579.

[25] L. Cheng, W. Liu, C. Yang, T. Huang, Z.-G. Hou, M. Tan, A neural-network-based controller for piezoelectric-actuated stick–slip devices, IEEE Transactions on Industrial Electronics 65 (3) (2017) 2598–2607.

[26] H. Liu, L. Cheng, M. Tan, Z. Hou, Containment control of general linear multi-agent systems with multiple dynamic leaders: A fast sliding mode based approach, IEEE/CAA Journal of Automatica Sinica 1 (2) (2014) 134–140.

[27] H. Liu, M. Zhou, Q. Liu, An embedded feature selection method for imbalanced data classification, IEEE/CAA Journal of Automatica Sinica 6 (3) (2019) 703–715.

[28] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver press, 2010.

[29] J. Krause, J. Cordeiro, R. S. Parpinelli, H. S. Lopes, A survey of swarm algorithms applied to discrete optimization problems, in: Swarm Intelligence and Bio-Inspired Computation, Elsevier, 2013, pp. 169–191.

[30] H. Shayanfar, F. S. Gharehchopogh, Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems, Applied Soft Computing 71 (2018) 728–746.

[31] E. Wari, W. Zhu, A survey on metaheuristics for optimization in food manufacturing industry, Applied Soft Computing 46 (2016) 328–343.

[32] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, Q. Zhu, A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE transactions on neural networks and learning systems 27 (3) (2015) 579–592.

[33] X. Luo, M. Zhou, Y. Xia, Q. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1273–1284.

[34] J. Li, Q. Pan, P. Duan, H. Sang, K. Gao, Solving multi-area environmental/economic dispatch by pareto-based chemical-reaction optimization algorithm, IEEE/CAA Journal of Automatica Sinica (2017).

[35] P. J. Angeline, G. M. Saunders, J. B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE transactions on Neural Networks 5 (1) (1994) 54–65.

[36] G. Wei-Shang, S. Cheng, Iterative dynamic diversity evolutionary algorithm for constrained optimization, Acta Automatica Sinica 40 (11) (2014) 2469–2479.

[37] D. E. Golberg, Genetic algorithms in search, optimization, and machine learning, Addion wesley 1989 (102) (1989) 36.

[38] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, arXiv preprint arXiv:1712.06567 (2017).

[39] J. Kennedy, R. Eberhart, C. 1995. particle swarm optimization, in: IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, 2001, pp. 1942–1948.

[40] J. Wang, T. Kumbasar, Parameter optimization of interval type-2 fuzzy neural networks based on pso and bbbc methods, IEEE/CAA Journal of Automatica Sinica 6 (1) (2019) 247–257.

[41] H. Shi, L. Wang, T. Chu, Swarming behavior of multi-agent systems, Journal of Control Theory and Applications 2 (4) (2004) 313–318.

[42] M. G. Hinchey, R. Sterritt, C. Rouff, Swarms and swarm intelligence, Computer 40 (4) (2007) 111–113.

[43] X. Feng, Y. Wang, H. Yu, F. Luo, A novel intelligence algorithm based on the social group optimization behaviors, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (1) (2018) 65–76.

[44] M. Dorigo, G. Di Caro, Ant colony optimization: a new meta-heuristic, in: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), Vol. 2, IEEE, 1999, pp. 1470–1477.

[45] M. Neshat, G. Sepidnam, M. Sargolzaei, A. N. Toosi, Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications, Artificial intelligence review 42 (4) (2014) 965–997.

[46] X. Jiang, S. Li, Bas: beetle antennae search algorithm for optimization problems, arXiv preprint arXiv:1710.10724 (2017).

[47] X. Jiang, S. Li, Beetle antennae search without parameter tuning (bas-wpt) for multi-objective optimization, arXiv preprint arXiv:1711.02395 (2017).

[48] X.-S. Yang, S. Deb, Engineering optimisation by cuckoo search, arXiv preprint arXiv:1005.2908 (2010).

[49] J. Zhao, S. Liu, M. Zhou, X. Guo, L. Qi, Modified cuckoo search algorithm to solve economic power dispatch optimization problems, IEEE/CAA Journal of Automatica Sinica 5 (4) (2018) 794–806.

[50] A. R. Mehrabian, C. Lucas, A novel numerical optimization algorithm inspired from weed colonization, Ecological informatics 1 (4) (2006) 355–366.

[51] S. Nakrani, C. Tovey, On honey bees and dynamic server allocation in internet hosting centers, Adaptive Behavior 12 (3-4) (2004) 223–240.

[52] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artificial intelligence review 31 (1-4) (2009) 61–85.

[53] X.-S. Yang, Firefly algorithms for multimodal optimization, in: International symposium on stochastic algorithms, Springer, 2009, pp. 169–178.

[54] M. Shang, X. Luo, Z. Liu, J. Chen, Y. Yuan, M. Zhou, Randomized latent factor model for high-dimensional and sparse matrices from industrial applications, IEEE/CAA Journal of Automatica Sinica 6 (1) (2018) 131–141.

[55] Z. Zhu, Z. Zhang, W. Man, X. Tong, J. Qiu, F. Li, A new beetle antennae search algorithm for multi-objective energy management in microgrid, in: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE, 2018, pp. 1599–1603.

[56] X. Yin, Y. Ma, Aggregation service function chain mapping plan based on beetle antennae search algorithm, in: Proceedings of the 2nd International Conference on Telecommunications and Communication Engineering, ACM, 2018, pp. 225–230.

[57] X. Lin, Y. Liu, Y. Wang, Design and research of dc motor speed control system based on improved bas, in: 2018 Chinese Automation Congress (CAC), IEEE, 2018, pp. 3701–3705.

[58] Y. Sun, J. Zhang, G. Li, Y. Wang, J. Sun, C. Jiang, Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes, International Journal for Numerical and Analytical Methods in Geomechanics 43 (4) (2019) 801–813.

[59] Q. Wu, X. Shen, Y. Jin, Z. Chen, S. Li, A. H. Khan, D. Chen, Intelligent beetle antennae search for uav sensing and avoidance of obstacles, Sensors 19 (8) (2019) 1758.

[60] M.-j. LIN, Q.-h. LI, A hybrid optimization method of beetle antennae search algorithm and particle swarm optimization, DEStech Transactions on Engineering and Technology Research (ecar) (2018).

[61] Q. Wu, H. Lin, Y. Jin, Z. Chen, S. Li, D. Chen, A new fallback beetle antennae search algorithm for path planning of mobile robots with collision-free capability, Soft Computing (2019) 1–12.

[62] Y. Fan, J. Shao, G. Sun, Optimized pid controller based on beetle antennae search algorithm for electro-hydraulic position servo control system, Sensors 19 (12) (2019) 2727.

[63] S. Xie, X. Chu, M. Zheng, C. Liu, Ship predictive collision avoidance method based on an improved beetle antennae search algorithm, Ocean Engineering 192 (2019) 106542.

[64] J. Yang, Z. Peng, Beetle-swarm evolution competitive algorithm for bridge sensor optimal placement in shm, IEEE Sensors Journal (2019).

[65] N. Qian, On the momentum term in gradient descent learning algorithms, Neural networks 12 (1) (1999) 145–151.

[66] A. P. Engelbrecht, Fitness function evaluations: A fair stopping condition?, in: 2014 IEEE Symposium on Swarm Intelligence, IEEE, 2014, pp. 1–8.

[67] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[68] Mathworks, MATLAB: Global Optimization Toolbox 2018b, The Mathworks Inc., 2018.