Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Optimization of decision trees using modified African buffalo algorithm

Archana R. Panhalkar [a,*], Dharmpal D. Doye [b]

[a] Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India
[b] Electronics and Telecommunication Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India

ARTICLE INFO

Article history:
Received 5 October 2020
Revised 18 January 2021
Accepted 18 January 2021
Available online 4 February 2021

Keywords:
Decision tree
Swarm intelligence
Nature-inspired
Data mining
African Buffalo optimization
C4.5

ABSTRACT

Decision tree induction is a simple, however powerful learning and classification tool to discover knowledge from the database. The volume of data in databases is growing to quite large sizes, both in the number of attributes and instances. Some important limitations of decision trees are instability, local decisions, and overfitting for this extensive data. The simple, effective and non-convergence nature of the African Buffalo Optimization (ABO) algorithm makes it suitable to solve complex optimization problems. In this paper, we propose the African Buffalo Optimized Decision Tree (ABODT) algorithm to create globally optimized decision trees using the intelligent and collective behaviour of African Buffalos. The modified African Buffalo optimization algorithm is used to create efficient and optimal decision trees. To evaluate the efficiency of the proposed African Buffalo Optimized Decision Tree algorithm, experiments are performed on 15 standard UCI learning repository datasets that are of various sizes and domains. Results show that the African Buffalo Optimized Decision Tree algorithm globally optimizes decision trees, increases accuracy and reduces the size of a decision tree. These optimized trees are stable and efficient than conventional decision trees.

1. Introduction

Large databases are important norms in today's digital world of big data. Data mining is the practice of investigating useful samples from these large databases collected from various sources. Data mining is considered the heart of analytic efforts across different industries and disciplines like communications, manufacturing, education, social media, insurance and retail. Depending upon user requirements, a variety of data mining techniques are used like classification, clustering, regression, summarization, association and anomaly detection. Classification is one of the important data analysis tasks used to categorize data efficiently. Among various classification techniques, the decision tree is found to be a simple, expressive, robust and efficient classifier (Han and Kamber, 2006).

A decision tree is a supervised expressive classifier that consists of nodes collection where internal nodes are testing nodes and leaf nodes are decision nodes. The key challenge in the decision tree implementation is to discover which attributes need to select at each level. Handling this selection is known as the attributes selection. There are diverse attributes picking measures to select the attribute which can best perform at each level. Some popular traditional algorithms like CART (Brieman et al., 1984), ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) are used to create accurate and robust decision trees. These methods are easy to understand and interpret. All these algorithms follow a top-down greedy approach to build a decision tree. These methods make a "greedy" choice to determine which attribute should be tested in the decision tree and how to determine splits. Besides the best expressive, easy to understand, comprehend qualities, decision trees are having the following major limitations (Bertsimas and Dunn, 2017; Bennett and Blue, 1996):

- As decision trees are unstable, small variations in training data can considerably change the organization of the decision tree.
- Due to local decisions to select the best attribute at each decision, produces inaccurate and suboptimal trees.
- Every decision is done depending on available data at each node, so fails to utilize characteristics of all data points, which leads to weak classification capability.
- Due to local search, influential splits are getting hidden at the back of weaker splits.

* Corresponding author.
E-mail address: archana10bhosale@rediffmail.com (A.R. Panhalkar).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.01.011
1319-1578/© 2021 The Authors. Published by Elsevier B.V. on behalf of King Saud University.

- Decision trees are one step optima, therefore each split is determined without consideration of future splits and adverse effects on a decision tree.
- The top-down approach of decision trees fails to handle penalties while growing trees more complex. This leads to overfitting of decision trees.
- The top-down induction approach typically optimizes an impurity measure while picking splits, rather than the use of the misclassification rate of training data points.
- Calculations of inducing trees become very complicated and lengthy, mainly if numerous values are uncertain or if several outcomes are linked.
- Decision tree composition is prone to sampling because these are robust to outliers and having a tendency to overfit. If sampled training data is a bit different than testing, then decision trees may produce an incorrect classification.

The main aim of this research is to overcome these limitations of decision trees and to create optimized and global decision trees. Nature-inspired algorithms like swarm intelligence algorithms are found to be promising in optimization problems due to their two important characteristics adaptability and scalability (Brezočnik et al., 2018; Abualigah, 2019). So to create a globally optimized decision tree, a novel nature-inspired algorithm called African Buffalo Optimized Decision Tree (ABODT) is proposed. This novel approach uses a modified version of a swarm intelligence algorithm called African Buffalo Optimization (Odili et al., 2015) to construct an efficient and optimized decision tree.

The rest of the paper has the following sections: in Section 2, brief discussions about various methods to optimize decision trees along with some meta-heuristic approaches are explained. In Section 3, basic African Buffalo Optimization is explained. The basic top-down approach to construct a decision tree is elaborated in Section 4. Section 5 gives a detailed discussion about a novel approach to optimize decision trees using modified ABO. Discussion on experiments and results are carried out in Section 6. Finally, the conclusion of the paper is given in Section 7.

2. Literature review

To avoid overfitting and large growing trees, pruning is one of the best mechanisms applied while constructing a decision tree or after the construction of a tree (Windeatt and Ardeshir, 2001; Esposito et al., 1997). Decision tree pruning eliminates one or more subtrees from a decision tree. It avoids overfitting of trees and pruned trees are more accurate in classifying testing data. Different methods have been proposed for decision tree pruning (Han and Kamber, 2006; Bertsimas and Dunn, 2017; Bennett and Blue, 1996). These methods selectively substitute a subtree with a leaf, if it does not decrease classification accuracy over the pruning data set. The main drawback of pruning is that it may increase the classification errors on the training data set, but it improves the accuracy of classification on unseen data points. The problem of overgrowing trees may be solved by using global decisions at each node while selecting a splitting attribute. The global selection approach increases the quality of a decision tree. This leads to constructing the whole decision tree in a single step, allowing every split to be found with complete knowledge of all other decision splits. This would result in the optimal tree for the given training data set. To find optimal decision trees is an NP-complete problem (Laurent and Rivest, 1976). Due to the number of attributes, outsized depth of the tree and numerous split points, tree splitting is not global. Thus the globally optimal tree is also practically impossible. To avoid classification errors, a decision tree should not be overfitted and be globally optimal.

Many efforts were carried out by researchers to develop effective ways of building optimal trees. Linear (Bennett and Evans, 1992) and continuous optimization (Bennett and Blue, 1996) approaches are used to construct optimal decision trees. Linear optimization is also found to be promising to optimize various classifiers (Bhosale and Chaudhari, 2019). Continuous optimization uses nonparametric objective error functions based on Frank Wolfe and Extensive point Tabu search algorithms (Bennett and Blue, 1996). A multi-linear programming non-greedy based method for global tree optimization methods produce high accuracy trees than standard greedy trees (Bennett, 1994). This method first creates a general decision tree and then fixes the structure of a tree. It minimizes classification errors to fix the structure of a tree. Iterative linear programming is used to create optimized decision trees. To create optimal binary decision trees, vector space partition is done using dynamic programming (Payne and Meisel, 1977). The evolutionary approach is used as a semi optimal hyperplane to construct the depth of a decision tree in the creation of optimal decision trees (Son, 1998). The major limitation of all these methods is that they are applicable to create only binary decision trees. Practically these methods fail to create the best optimal trees in confined time. Recently Mixed Integer Optimization is used to create optimal decision trees (Bertsimas and Dunn, 2017). The minimum depth and optimal nonbinary decision trees created using the enumeration approach (Tzirakis and Tjortjis, 2016).

From the literature, it is observed that very less work is done in the optimization of decision trees except for pruning methods. Very few methods are based on the evolutionary approach (Son, 1998) and meta-heuristic approaches used in literature to create optimized decision trees. Some decision tree building algorithms based on swarm intelligence uses Ant Colony Optimization to construct trees (Otero et al., 2012; Boryczka and Kozak, 2015). Otero and Freitas (Otero et al., 2012) proposed a novel method to build a decision tree called Ant-Tree-Miner by the selection of decision nodes using the amount of pheromone and heuristic information. This method uses construction graph representation where nodes of a decision tree are vertices and edges represent branches from decision nodes. Each vertex is represented by a triple [level, condition, edge]. The pheromone matrix is created by mapping triple into pheromone value. Boryczka and Kozak (Boryczka and Kozak, 2015) induced decision trees using the same approach, followed by Ant-Tree-Miner (Otero et al., 2012) with some modifications. In this modified approach, modified decision criteria are used to select decision nodes and each ant creates a decision tree. It selects the best decision tree created by ants. All these meta-heuristic approaches only concentrated on the accuracy of decision trees and failed to optimize trees. Ant Colony based miner (Ant-Miner) (Parpinelli et al., 2002) extracts accurate classification rules using Ant Colony optimization. It uses the sequential covering approach to find rules that completely covers all training instances. After finding rules, Ant-Miner performs pruning of rules, due to this efficient classification rules get produced. Ant Colony Decision Tree (ACDT) is a decision tree building algorithm based on Ant Colony Optimization (Boryczka and Kozak, 2010). In ACDT, ants are used to select the best attribute for splitting at each node. It uses towing criteria based on heuristic function and pheromone values to find a superior attribute.

Nature-inspired algorithms are promising problem-solving approaches and methods that catch the attention of researchers for their superior performance (Jr et al., 1307; Yang, 2014; Odili et al., 2017; Abualigah and Diabat, 2020; Abualigah et al., 2020). Various nature-inspired algorithms (Jr et al., 1307) are artificial neural networks, evolutionary computing, fuzzy systems, and swarm intelligence. These algorithms are capable of solving several real-world problems. Among all these promising methods, Swarm intelligence has been marked as one of the multidisciplinary approaches that focus on the collective behaviours of ants, flocks of birds, schools of fish and herds of land animals. Heuristics methods (Yang, 2014) use trial and error to find satisfactory results to a complex practical problem within time. It is not possible to find all possible solutions for complex problems. These algorithms must find feasible and better solutions within less time. Heuristic algorithms do not have any guarantee that they can find the best solutions. The nature-inspired algorithms will work in iterations and produce efficient solutions. Meta-heuristic algorithms have two major components exploration and exploitation. Generating diverse and global solution is called exploration. Exploitation means searching in a local region to find a current better solution by analyzing the information in this local region. The most popular combinatorial optimization problem is the Travelling Salesman's Problem (TSP). The performance of 11 nature inspired algorithms is analyzed on TSP (Odili et al., 2017). African Buffalo Optimization and Ant colony optimization were found to be superior for finding the optimal or near-optimal solution. The advantage of heuristics algorithms in solving the problems is striking because of their influential and robust searchability and its measures in dealing the high-dimensional problems (Abualigah et al., 2020).

Swarm intelligence comprises a population of agents having decentralized control (Li and Zhang, Aug. 2008). These agents interact locally with each other and with their environment. The motivation comes from the natural world, especially biological systems. These agents follow straightforward rules dictating how individual agents should behave (local decision) and to a certain random interaction between each other builds ''intelligent" global behaviour, strange to the individual agents. In swarm intelligence, agents choose their actions and then carry them out. Their flexible, robust, scalable, decentralized and self-organizational behaviour gives the best performance in optimization tasks. Swarm intelligence algorithms like Multi-vers optimization algorithms can be hybridized with other optimization algorithms to improve optimization in complex problems (Abualigah, 2020).

African Buffalo Optimization (ABO) is one of the simple non parameterized and efficient algorithms (Odili et al., 2015). The performance of ABO for different applications is compared and verified in the literature (Alweshah et al., 2020; Bera et al., 2020; El-Ashmawi, 2018; Odili and Kahar, 2015; Padmapriya and Maheswari, 2017; Singh et al., 2020). ABO based two-layer optimization method is proposed to determine the sizes and best possible sites of Battery Energy Storage System and Wind Turbines at the same time (Singh et al., 2020). A modified ABO algorithm is used for the integration of distributed sources of energy very effectively. It shows that the speed of ABO is better than Randomized Insertion. M. Alweshah et al. (Alweshah et al., 2020) proposed an effective method for neural networks based on ABO. The probabilistic neural networks (PNN) are trained and weights in the trained network are adjusted by applying ABO algorithm. ABO based PNN gives efficient performance for various datasets by increasing the accuracy of classification. To work in a large heterogeneous network with several intermediate nodes, ABO achieved efficient performance for controlling wireless sensor nodes (Bera et al., 2020). This is one of the effective methods to manage all sensor nodes in the network between the source node to the destination node. ABO is efficiently applied to globally reduce traffic in the network and energy consumption of intermediate sensor nodes. To solve various real-world problems, collaborative teams in social networking play an important role. W. Ashmawi (El-Ashmawi, 2018) applied ABO algorithm to reduce communication costs in these collaborative teams of social networks. ABO optimizes the cost of communication by combining a discrete crossover operator with a swap sequence operator. This improved ABO algorithm first finds a solution and then applies a discrete operator between the best herd fitness and exploration parameter of buffalo. Odili et al. (Odili and Kahar, 2015) applied ABO algorithm to optimize various numerical functions. These 20 benchmark functions are constrained to unconstrain, uni-modal to multi-modal from different domains which proves that ABO is the best meta-heuristic algorithm to perform global optimization. R. Padmapriya and D. Maheshwari (Padmapriya and Maheswari, 2017) applied ABO optimization to minimize interference in mobile cellular communication. This approach involves a combined support vector machine with ABO to increase throughput and capacity of wireless channels. Real-time classifiers require less response time and are possible with the help of optimization methods (Anwar and Azrag, 2015; Bhosale and Chaudhari, 2019; Bhosale and Chaudhari, 2017).

To deal with all limitations of decision trees described in Section 1, novel nature-inspired decision tree optimization techniques to optimize decision trees based on the modified African Buffalo Optimization algorithm. It creates efficient, accurate, small-sized decision trees. Results obtained on the various UCI machine learning repository data sets (Lichman, 2013) shows that it performs outstanding compared to existing decision tree construction approaches.

3. African buffalo optimization algorithm

Odili, Kahar and Anwar (Odili et al., 2015) proposed a novel, faster and simple African Buffalo Optimization algorithm to optimize the travelling salesman problem. ABO is inspired by the movements of African buffalos which are herbivorous and migrate in search of lush green pastures. The herd of buffalos is searching for secure and greenery areas around South Africa. African buffaloes have three extraordinary attributes that help them to survive. The first important attribute is that they are communicating intelligently with each other using their ''maaa" and ''waaa" sounds. ''waaa" vocalization is an indication to other buffalos in the herd that keep moving because the place is not suitable or it is dangerous (prone to the attack of other animals). ''maaa" vocalization indicates other buffalos in the herd that stay on to use the present location as it is safe and good grazing pastures. The second attribute is their extensive memory capacity to keep track of thousands of kilometers routes. The last and very important attribute is the democratic nature of buffalos. If opposing calls in the herd by some buffaloes, then ''election" is taken by buffaloes to take the final decision to move or stay. All these intelligent behaviour of buffaloes gives larger productive results. All these intelligent behaviour of buffaloes to optimize a decision tree are used in ABODT. In this work, a modified version of ABO is used to create optimized decision trees, which are efficient and optimized. These optimized decision trees give better performance than other optimization techniques and pruning of decision trees.

The ABO algorithm (Odili et al., 2016) starts with placing each buffalo randomly in an n-dimensional solution space. In this algorithm buffalos, ''maaa" (stay on to exploit) signals indicate exploitation and the ''waaa" (move ahead) signal represents an exploration of a new search. After placing buffalos randomly, each buffalo's exploitation (''maaa" value)Ma_{k+1} is updated with the help of bgr_{max} (best fitness of herd) and bpr_{max} (best fitness of individual buffalo) using Eq. (1).

Algorithm 1. African Buffalo optimization(ABO)

```
1. Consider $n$ buffalos with objective optimization function
$f(x)x = (x1, x2, \cdots\cdots\cdots xn)^T$
2. Initialization of buffalo: Place each buffalo randomly at
   nodes in solution space.
3. for each $k = 1$ $to$ $n$ ‘do
```

(*continued on next page*)

(*continued*)

Algorithm 1. African Buffalo optimization(ABO)

```
Update fitness value using Eq. (1).
```

$$Ma_{k+1} = Ma_k + lfrand_1(bgr_{max} - Wa_k) + lfrand_2(bpr_{max.k} - Wa_k) \quad (1)$$

```
4. Update the location of the buffalos in relation to $bgr_{max}$ and
   $bpr_{max.k}$ using Eq. (2).
```

$$Wa_{k+1} = (Wa_k + Ma_k)/\pm 0.5 \quad (2)$$

```
5. If $bgr_{max}$ updated then go to step 6 else go to step 2.
6. Check for stopping criteria, if it is not met then go to step 3.
end
7. Output best optimized solution.
```

If the current fitness of buffalo is better than the individual's maximum fitness ($bpr_{max.k}$), then it keeps the location vector for this particular buffalo. If the current fitness is better than the herd's overall best fitness then this herd's maximum fitness (bgr_{max}) get saved and used in the next iteration. By using Eq. (2), all buffalos location is updated and looks at the next buffalo in the population. After that check for global best fitness meets the exit criteria, it ends the algorithm and gives the saved current location vector as the solution to the given problem. If the best buffalo location is not improved for the number of iterations, the whole herd is re-initialized and goes to step 2. In Eq. (1), $lfrand_1$ and $lfrand_2$ are learning parameters that take random values in the range [0–1].

Due to the flexible behaviour of this nature-inspired algorithm, modified ABO is used to optimize the decision tree. The resulting optimized tree is capable of performing global optimization. This efficient, non parameterized and simple algorithm was never used before in the context of a decision tree. The advantage of this modified ABO method is that it does not depend on parameters, simple to design and easy to implement. ABODT algorithm is elaborated in Section 5 in detail.

4. Decision tree learners

Decision tree learners are influential analytical models that are simple to understand, visualize, apply and evaluate (Quinlan, 1986). The decision tree is a supervised classifier. It trains a decision tree classifier and uses it for testing unseen instances. One of the major advantages of a decision tree is that it requires fewer data to train and gives the best evaluation performance. The decision tree building algorithms (Quinlan, 1986, 1993) use a divide and conquer method to construct a tree. It constructs a decision tree using a given T_{train} training set. This training set contains a set of instances depending upon the size of the dataset used for training. Each instance consists of attribute values along with the class. It calculates the frequency of a class for instances in a training set T_{train}. If all instances are of the same class, then node is created with that class. But, if set T_{train} contains instances of more than one class, the best attribute using splitting criteria is selected for splitting. The training set T_{train} is partitioned into k different subsets $\{T_1, T_2, \cdots\cdots\cdots T_k\}$ using the test on selected attributes. Recursively the algorithm is applied to each non-empty partition. The algorithm for building a decision tree (Quinlan, 1986, 1993) is as follows:

Algorithm 2. Decision Tree Construction

```
Build_Tree
  ($T_{train}, A$)
1. Create a decision node $nd$.
2. If all data samples from $T_{train}$ are of the homogeneous class
   $C_i$ then make $nd$ as a leaf node with label $C_i$.
3. If attribute set $A$ is empty then make $nd$ as a leaf node with
   the most common class $C_i$ label (using majority voting).
4. Select attribute $A_j$ from attribute set $A$ , with the highest
   information gain, then Label $nd$ with attribute $A_j$.
5. For each value $v$ of attribute $A_j$:
   a. Make a branch from $nd$ with condition $A_j = v$.
   b. Let $S_{train}$ be the subset of samples in $T_{train}$ with $A_j = v$.
   c. If $S_{train}$ is NIL then-Attach $A_j$ label to the leaf node with
      the most common class in $T_{train}$, else Attach node created by Build_Tree ($T_{train}, A$).
```

This decision tree algorithm builds a tree model using a recursive method. Once the model is trained, the unseen samples are applied to check the accuracy of a model. CART (Brieman et al., 1984), ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) are some popular and widely used methods for creating a decision tree. Because of their performance, these are used in the construction of decision tree forests and hybridized classifiers. The main aim of ABODT is to overcome the drawbacks of decision trees stated in section 1. These trees take local decisions based on available data, so fail for a global decision. Proposed method is optimizing these trees to take global decisions with the help of ABO algorithm. These decision trees are optimized by taking characteristics of African buffaloes. Results show that optimized decision trees by using African Buffalo Optimized Decision Tree (ABODT) method are capable of taking global decisions and perform best as compared to methods in the literature. We are using Weka implementation of J48 algorithm to apply our ABODT algorithm (Hall et al., 2009).

5. Proposed System methodology

A decision tree is one of the promising classifiers for categorizing data accurately. Decision trees are prone to over-fitting, under-fitting and local decisions (Otero et al., 2012; Boryczka and Kozak, 2015). To improve and globally optimize the decision tree, various metaheuristic algorithms, including genetic algorithm and ant colony optimization are used in the literature (Otero et al., 2012; Boryczka and Kozak, 2015, 2010; Parpinelli et al., 2002; Jr et al., 1307; Yang, 2014; Odili et al., 2017). All these algorithms are parametric and require reinforcement to achieve better results and make them faster. ABO is found to be promising in information propagation because the entire herd works as a unit. In this paper, modified African Buffalo Optimization is used to create a globally optimal and accurate decision tree. African Buffalo Optimized Decision Tree (ABODT) is proposed to create an efficient, accurate and optimized decision tree. The ABODT algorithm is less parametric. Behavior and characteristics of African Buffalos are the main motivation to optimize decision trees. This algorithm uses all attributes of African buffalos for improving the classification performance of the decision tree. The following characteristics of African buffalos are used in modified ABO to create globally optimized decision trees, which are not inculcated in the standard ABO algorithm created by Odili et al. (2015).

- When African buffalos move from one position to another in search of green pasture, the best buffalo move towards the green pasture, if the target area with green pasture is safe then only it calls other buffalos.

- After the first buffalo migrated to the target area, then other buffalos move one by one to the target area.
- After migration, if any buffalo finds or recognizes the target area is not safe then that buffalo return back.
- If unsafe buffalo moves backward, then the remaining buffalos take decision based on the decision of the majority buffalos. If the majority of buffalos migrated towards the target area compared to buffalos coming back then these buffalos migrate towards the target area. This property of buffalos is called an election and depending on the majority, buffalos take a decision.
- If compared to several buffalos moving back, maximum buffalos are migrated to the target area, then the target area is safe and secure for buffalos.

All these characteristics are used in the modified buffalo optimization algorithm. The modified ABO with the above characteristics creates optimized trees. The proposed ABODT algorithm creates an optimized decision tree whose size is less compared to the original size of decision trees. The general procedure of the proposed ABODT is shown in Fig. 1 with a block diagram. This block diagram shows major steps in the proposed ABODT algorithm which starts from taking input data and performing optimization of decision trees using ABODT algorithm.

Following are the steps of the modified ABO algorithm which creates globally optimal decision trees and based on this, ABODT algorithm is implemented and evaluated. Modified ABO algorithm is depicted in Algorithm 3:

Algorithm 3. Modified ABO algorithm

1. Initialization: Randomly place buffalos in n-dimensional search space.
2. Move best Buffalo to the target green pasture: The buffalo with the highest fitness is moved towards target safe and lust green area.
3. Best buffalo give “waaa” call to remaining buffalos: When best buffalo finds that the target area is suitable and safe then it calls other buffalos with “waaa” call.
4. Calculate Fitness of buffalos in the target area: After moving each buffalo to the target area, the new fitness of each buffalo is calculated using “maaa” call. If the herd’s fitness (bgrmax) is increased after adding this new fitness then buffalo will remain in the target area.
5. If migrating buffalo finds target area insecure then it returns back: In the target area, if any migrated buffalo finds insecurity then buffalo moves back towards the old area.
6. Check fitness of target area: If the majority buffalos migrate to the target area means the target area is secure and safe, which indicates the fitness of the target area is good compared to the old area.

The steps of ABODT algorithm to create a globally optimized decision tree using a modified ABO algorithm are shown in Algorithm 4. The ABODT starts with the construction of decision trees from training data and then a modified African Buffalo Optimization

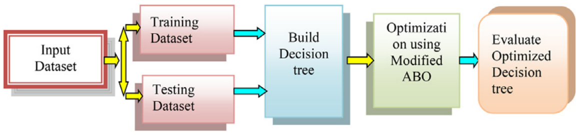
Input Dataset
Training Dataset
Testing Dataset
Build Decision tree
Optimization using Modified ABO
Evaluate Optimized Decision tree

Fig. 1. General Block diagram of proposed ABODT algorithm.

algorithm is applied to optimize the decision tree. The ABODT algorithm improves various parameters like the accuracy and size of a decision tree and avoids over-fitting by creating a globally optimized decision tree. The accuracy of a decision tree is calculated by the number of unseen instances correctly classified by a decision tree out of the total number of instances supplied for testing. The total number of nodes in the decision tree is depicted as the size of a decision tree.

In ABODT, C4.5 decision tree (Quinlan, 1986) using Algorithm 1 is constructed and then it is optimized using a modified ABO algorithm. The C4.5 decision tree (Quinlan, 1986) was found to be a promising classifier, which gives excellent performance for all types of data. Some basic terminologies are used in the ABODT algorithm. Nodes of decision trees are depicted as buffalos. Each non-leaf node represents the best feature selected for comparison while creating a decision tree. Each non-leaf node corresponds to one buffalo in ABODT. Suppose n number of nodes are created in the decision tree. This set of nodes (attributes or features) with fitness is input to the modified ABO algorithm. The fitness calculation for each node in each iteration evaluates the competence with other nodes. The main objective to calculate fitness is the selection of the best node on the upper level of the tree. These nodes also consider as the best features in an individual tree after applying optimization criteria.

In ABODT, the fitness of any node is calculated using the heuristic function called the information gain ratio. The heuristic is a function that is used in a knowledgeable search, and it finds the best path. It input the current state of the agent and finds the estimation of how close the agent is to the target goal. The heuristic method gives the best solution maximum time, but it sure to find a better solution in minimum time. The gain ratio (Quinlan, 1993) is used to find a good attribute from a set of attributes that split the data so that each successor node is as pure as possible. It selects the best attribute with high degree order which distributes instances of a single class from other classes. Information gain is biased towards multi-valued attributes (Han and Kamber, 2006). This drawback is removed by gain ratio, so to select a node with the best fitness, ABODT uses the gain ratio which is used in C4.5. The training set T_{train} consist of $C = \{C_1, C_2, C_3, \cdots .C_n\}$ be n different classes. The probability p_i that an instance belongs to a class C_i is calculated using Eq. (3)

$$p_i = \frac{freq(C_i, T_{train})}{|T_{train}|} \tag{3}$$

Where, $|T_{train}|$ is the total number of instances in T_{train}. The number of instances belongs to a class C_i is given by$freq(C_i, T_{train})$. Information gain of T_{train} is calculated using Eq. (4)

$$info(T_{train}) = -\sum_{i=1}^{n} p_i \times \log_2(p_i) \tag{4}$$

The dataset T_{train} partitioned into s partitions based on domain values of a non-class attribute A_i. The information obtained by this portioning process is given by Eq. (5)

$$info(A_i, T_{train}) = -\sum_{j=1}^{s} \frac{|T_{trainj}|}{|T_{train}|} info(T_{trainj}) \tag{5}$$

The information gain of attribute A_i is calculated using Eq. (6)

$$gain(A_i) = info(T_{train}) - info(A_i, T_{train}) \tag{6}$$

The problem of information gain is that it is biased towards attributes with many values. So Gain ratio which is used as fitness in ABODT is calculated using Eq. (7).

$$gainratio(A_i) = \frac{gain(A_i)}{splitinfo(A_i)} \tag{7}$$

The $splitinfo(A_i)$ calculates information gained by splitting the training set T_{train} into s subset on test attribute A_i. The $splitinfo(A_i)$ is calculated by Eq. (8).

$$splitinfo(A_i) = -\sum_{j=1}^{s} \frac{|T_{trainj}|}{|T_{train}|} \times \log_2\left(\frac{|T_{trainj}|}{|T_{train}|}\right) \tag{8}$$

The gain ratio of a node is the gain ratio of a attribute tested at that node. To select the best buffalo, ABODT uses the fitness of the node which is calculated using the gain ratio, which gives an efficient performance in the optimization of a decision tree. These buffalos make "maaa" and "waaa" to reposition randomly based on the best buffalo fitness. The entire ABODT algorithm is given in Algorithm 4 as follows:

Algorithm 4. Proposed African Buffalo Optimized Decision Tree (ABODT)

```
Input: Decision Tree Tr($nd_1, nd_2, nd_3, \cdots.., nd_n$) using training set$T_{train}$
Output : Optimized $Opt\_Tr$.
Begin
1. Randomly place buffalos at each node in Tr
2. Initialize
$Opt\_Tr \leftarrow \{\emptyset\}$
for each node $nd$ in Tr do
    $Wa_{nd} \leftarrow 0$
    $Ma_{nd} \leftarrow 0$
end
3. for each node $nd$ in Tr do
4. Move the best buffalo (node) to the target area ($Opt\_Tr$).
$Opt\_Tr \leftarrow Opt\_tr \cup \{nd\}$
5. $lfrand_1$ = Math.Random()
$lfrand_2$ = Math.Random()
$bgr_{max}$ = Mean$\sum_{k=0}^{m} fitness(k)$
$bgr_{max} = fitness(k)$
where $m$ is the total number of migrated buffalos (nodes) to the target
area($Opt\_Tr$).
6. Update Exploitation of each buffalo
```

$$Ma_{nd+1} = Ma_{nd} + lfrand_1(bgr_{max} - Wa_{nd}) + lfrand_2(bpr_{max.nd} - Wa_{nd}) \tag{9}$$

```
7. Update exploration of each buffalo
```

$$Wa_{nd+1} = (Wa_{nd} + Ma_{nd})/\pm 0.5 \tag{10}$$

```
8. Is $bgr_{max}$ updated? If yes go to step-4 else goto step 7
9. Remove node from $Opt\_Tr$ (Buffalo return back due to unsafe target area)
$Opt\_tr \leftarrow Opt\_tr - \{nd\}$ goto step-4
10. end
11. Evaluate optimized tree $Opt\_Tr$ using a training set$T_{test}$
```

Entire ABODT algorithm 4 is explained in detail with the following stages:

Initialization: This is the basic step of every meta-heuristic algorithm used to initialize the population. In the ABODT algorithm, the population of buffalos is the number of non-leaf nodes in a decision tree *Tr*. The configuration process achieves by putting buffalos at k^{th} node in search space at random as shown in step-1 of Algorithm 4. Each non-leaf node represents a buffalo. The fitness of each node is a gain ratio *(bpr_{max})* of that node. As shown in step-2, initialize the optimized tree as empty and exploitation and exploration of each buffalo to 0.

Move node with the highest fitness to the target optimized list and update Location of Individual Node: Select node from decision tree Tr with the highest fitness and move to target optimized decision tree list. Update exploitation of node using Eq. 9 as shown in step-5 and step-6. bgr_{max} is herd's fitness which is an optimized decision tree's fitness. Every meta-heuristic algorithm uses fitness as an essential factor for optimization, so it is calculated for each node in tree Tr using Eq. 9. Each buffalo's updated exploitation Ma_{nd+1} is calculated using the previous exploitation location Ma_{nd}. The information-based parameters $lfrand_1$ and $lfrand_2$ helps the buffalos in deciding the necessity for relocation. This is achieved by subtracting the present exploration location Wa_{nd} from their generally best bgr_{max} and personal best bpr_{max}. Finally, migrated buffalo updates it's position using Eq. (10) to best fit in the target area.

Determine the fitness and encourage other nodes to explore: In this phase, each node determines its location according to its former maximum position bpr_{max} as well as some evidence obtained from the exploits of the nearest neighbours. This is performed using Eq. (9). This enhances the algorithm to facilitate the direction of nodes in reference to the optimization process. An important decision is taken in this step is to decide whether exploration will lead to achieving better fitness or not as shown in step-7. If it fails to achieve, then the current exploration is not updated. Following the herd's best buffalo fitness, update the new location of buffalo (Exploration) Wa_{nd+1} using Eq. (10), which is used to update the value of the herd's best fitness bgr_{max}. In Eq. (10), the exploitation driver is selected as 0.5 for the best performance.

Check the fitness and take decision based on updated fitness: If the herd's fitness bgr_{max} is increased, then the movement of buffalos is considered. If bgr_{max} is not increased, then check for stopping criteria otherwise return buffalos and repeat the process for remaining buffalos as shown in step-8 and step-9.

Evaluate the optimized list of nodes: Once the criteria have met, the algorithm terminates the execution, like the travelling salesman problem, GA, PSO, etc. according to the optimization strategy we need to evaluate an optimized tree on the testing dataset.

After stopping, ABODT produces the best-optimized tree with string feature selection and optimization algorithm. The entire flow of an ABODT algorithm is shown in Fig. 2. The entire flow shows that after building a decision tree, the modified ABO algorithm is applied. The main aim to apply ABO algorithm is to obtain global optimization of decision tree classifiers and the resulting tree should be accurate and efficient. The efficiency of any algorithm is measured by the time and space required for an algorithm to find a result. The ABODT algorithm classifies instances accurately as well as it requires less time and space to classify the unseen instances.

6. Experimental results

6.1. Dataset description

The ABODT algorithm is evaluated on 15 various standard datasets from the UCI machine learning repository (Lichman, 2013). These datasets are from various domains with numerical and cat-

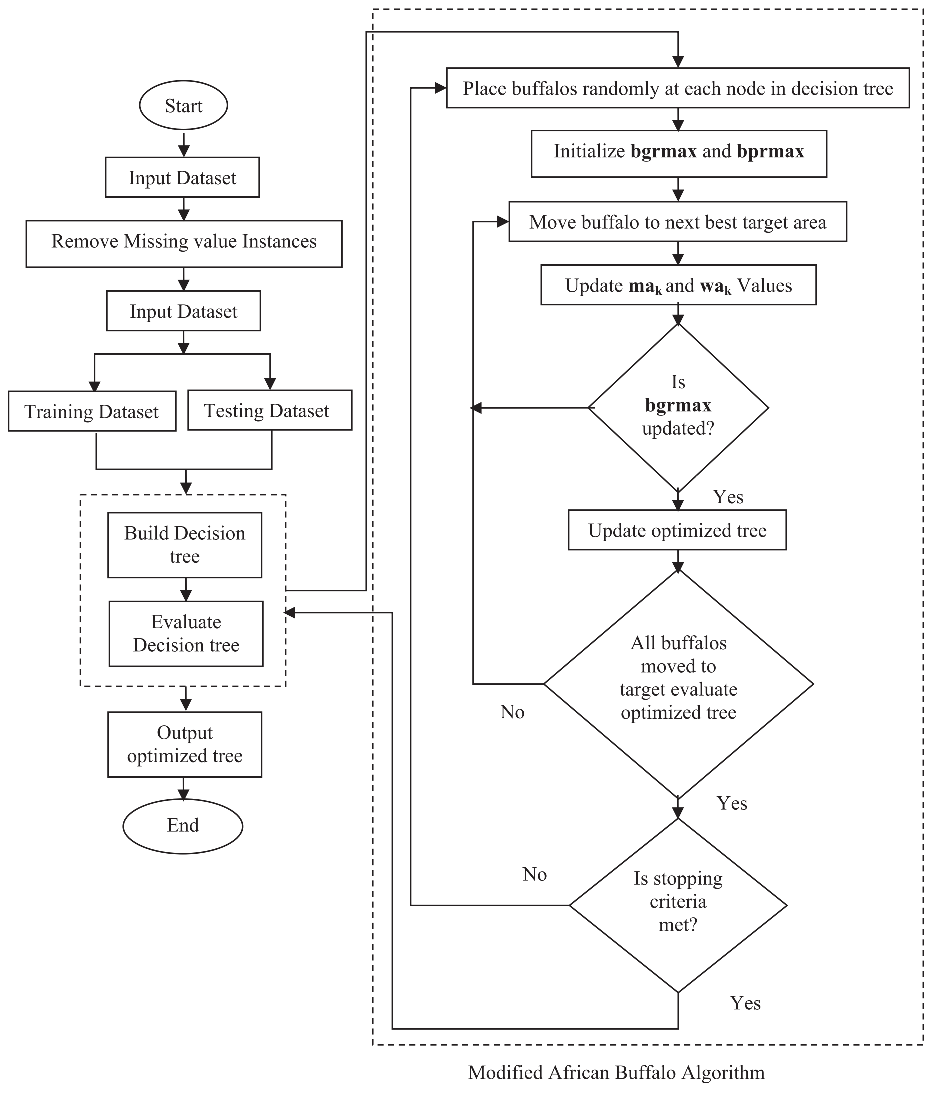
Start
Input Dataset
Remove Missing value Instances
Input Dataset
Training Dataset
Testing Dataset
Build Decision tree
Evaluate Decision tree
Output optimized tree
End
Place buffalos randomly at each node in decision tree
Initialize bgrmax and bprmax
Move buffalo to next best target area
Update ma_k and wa_k Values
Is bgrmax updated?
Yes
Update optimized tree
All buffalos moved to target evaluate optimized tree
No
Yes
Is stopping criteria met?
No
Yes
Modified African Buffalo Algorithm

Fig. 2. Detailed Flow of an entire ABODT algorithm.

egorical attributes which show that ABODT with modified ABO gives better performance than other meta-heuristic algorithms. To evaluate the results of the proposed method, a 10-fold cross-validation method is used. Each dataset is divided into ten parts.

Table 1
Summary of datasets.

Dataset	Instances	Attributes	Classes
Car Evaluation	1728	6	4
Iris	150	4	3
Lung Cancer	32	56	4
Nursery	12,960	8	5
Mushroom	8124	22	2
Lymphography	148	18	4
Wine	178	13	3
Sonar	208	60	2
Glass	214	9	6
Liver Disorder	345	6	2
Zoo	101	17	7
kr-vs-kp	3196	36	2
Tic-Tac-Toe	958	9	2
Pima Indian Diabetes	768	8	2
Statlog Vehicle	846	18	4

In every fold, training is done on nine parts of the dataset and one part is used for evaluating the performance on the ABODT algorithm. Detail description of these UCI datasets is given in Table 1.

6.2. Comparison of ABODT with basic decision tree algorithms

To prove the efficiency of the ABODT algorithm, first results are compared with C4.5 basic decision trees which are promising basic classifiers. Results of prediction accuracy are calculated for all folds and then the average result is calculated. Before applying algorithms, records with missing values are removed from a dataset. Predication accuracy is calculated by the number of instances classified correctly to the total number of instances tested. The C4.5 decision tree algorithm is found to be an efficient and accurate classifier and to tackle optimized tree characteristics, ABODT is compared with C4.5 for various parameters like accuracy for unseen instances, size of classifier and number of leaves. Any optimized tree has high accuracy, minimum leaves and small size compared to the original decision tree. Table 2 shows a comparison of accuracy (%), size of tree and number of leaves of ABODT with C4.5.

Results of various sized datasets show that the accuracy of ABODT is better than the C4.5 decision tree classifier. The accuracy

Table 2
Accuracy, size and leaves comparison of C4.5 with the proposed ABODT method.

Dataset	Method	Accuracy	No. of leaves	Size of tree
Car evaluation	C4.5	92.47	131	182
	ABODT	**94.10**	**116**	**159**
Lung cancer	C4.5	40.65	6	11
	ABODT	**47.33**	6	**9**
Vehicle	C4.5	73.80	98	195
	ABODT	**75.74**	**75**	**159**
Sonar	C4.5	72.57	18	35
	ABODT	**73.10**	**14**	**29**
IRIS	C4.5	96	5	9
	ABODT	**97.12**	**4**	**8**
Liver	C4.5	63.67	26	51
	ABODT	**71.88**	**22**	**43**
Diabetes	C4.5	73.83	20	39
	ABODT	**76.19**	**17**	**30**
Wine	C4.5	93.28	5	9
	ABODT	**95.32**	**5**	**9**
Glass	C4.5	66.82	30	59
	ABODT	**73.24**	**20**	**47**
Mushroom	C4.5	**100**	24	29
	ABODT	96.2	**19**	**25**

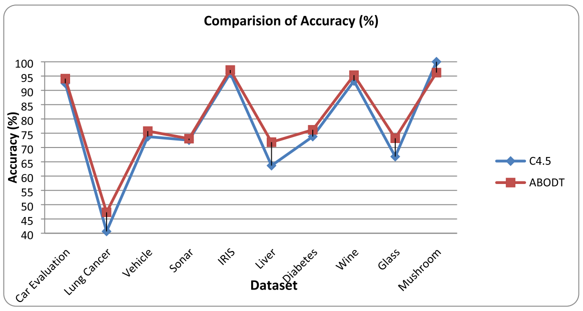

Fig. 3. Accuracy (%) comparison of C4.5 with the proposed ABODT method.

of the Mushroom dataset is decreased but the size and number of leaves are less compared to the C4.5 decision tree. As shown in Fig. 3, the accuracy of the decision created by ABODT is increasing from1% to 8% due to moving buffalos from one position to another for global optimization. The modified ABO algorithm avoids the overfitting of decision trees caused by excess data. Decision trees created by using ABODT are smaller than the original C4.5 decision trees. For the Liver dataset and Glass dataset, ABODT not only increases accuracy but also optimizes a tree. An optimized tree is measured by using the size of a tree and the number of leaves in decision trees. As the number of leaves more means comparisons are more, which leads to increased time for testing. The ABODT makes the tree globally optimal and accurate as compared to the C4.5 decision tree.

6.3. Comparison of ABODT with other meta-heuristic algorithms

To validate the performance of the ABODT algorithm, experiments are carried out to compare ABODT with efficient decision tree methods based on the meta-heuristic algorithms. The ABODT algorithm is unique and the first method that uses ABO algorithm to optimize the decision tree. The main feature of ABO algorithm is that buffalos moved from one position to another for increasing the fitness of the group. Increased fitness leads to improvement in the accuracy of a decision tree. Ant Miner (Parpinelli et al., Aug. 2002) and ACDT (Boryczka and Kozak, 2010) are meta-heuristic algorithms used to optimized decision trees using any colony optimization algorithm. ABODT is compared with these meta-heuristic algorithms. Accuracy results (%) of the comparison between Ant miner and ACDT are shown in Table 3. Results of experiments show that ABODT performs better than Ant miner and ACDT for maximum datasets. In Ant Miner and ACDT algorithms, the behavior of ants is used to create optimized decision trees. In the ACDT algorithm, population size 10 is used to carry out experiments. In Ant Miner, the population of ants is equal to the number of complete rules generated by using a decision tree. In the ABODT algorithm, the population depends on the size of decision trees. The Number of buffalos is equal to the number of decision nodes in the C4.5 decision tree. So a population of ABODT depends on the number of nodes present in a decision tree.

Table 3
Accuracy (%) comparison of Ant Miner and ACDT with the proposed ABODT method.

Dataset	Ant Miner	ACDT	Proposed
Zoo	85.41	96.8	**98.01**
Lymphography	75.53	80.23	**86.10**
Breast cancer	92.44	92.58	**96.25**
Tic-Tac-Toe	73.24	**93.16**	87.70
kr-vs-kp	92.97	99.39	**99.60**
Nursery	86.23	**99.41**	98.30

Prediction accuracy for 6 different UCI datasets is tested using 10-fold cross-validation. Predication accuracy is the ratio of correctly classified instances from the test dataset. From these datasets, the accuracy of Zoo, Lymphography, Breast Cancer and Kr-vs-Kp dataset gives more accuracy for the ABODT algorithm than the other two algorithms. ACDT algorithm gives better accuracy for Tic-Tac-Toe and Nursery datasets. As shown in Fig. 4, the accuracy of ACDT and ABODT is better than Ant miner algorithms. Now, the improvement of ABODT algorithm is evaluated using statistical significance tests (Demsar, 2006). In this paper, the statistical evaluation of results is done with the Friedman Test (Friedman, 1937, 1940). It is non-parametric statistics use to verify if a particular factor affects when we perform the repeated measures type of experiments.

To assess the improvement in results, the statistical significance test is carried using the Friedman Test. The null-hypothesis to test using the Friedman test is that all algorithms are equivalent. Friedman Test is used to compare multiple classifier models on various datasets. It ranks classification algorithms for each dataset separately. In ABODT statistical analysis number of classifiers (k) is 3 and number of datasets (N) are 6 (Table 3). The Friedman statistic X_F^2 (Demsar, 2006) is calculated using Eq. (11).

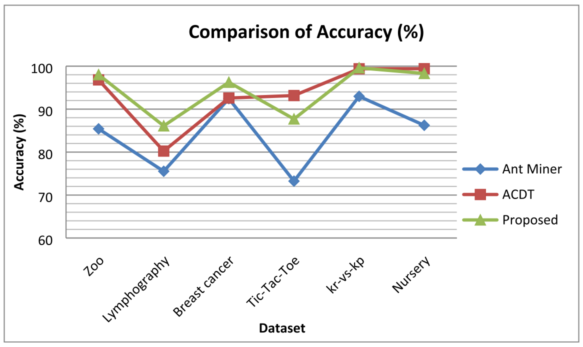

Fig. 4. Accuracy (%) comparison of the proposed ABODT method with Ant Miner and ACDT.

$$X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{11}$$

Where, R_j^2 is average rank jth of k algorithm. The Friedman statistic is distributed according to X_F^2 with $k-1$ degrees of freedom. An associated study (Iman and Davenport, 1980) shown that that Friedman's X_F^2 is undesirably conservative and relaxed to compute F_F as shown in Eq. (12).

$$F_F = \frac{(N-1)X_F^2}{N(k-1)-X_F^2} \tag{12}$$

For our experimental results X_F^2 is calculated as 9.3333 and F_F is 17.4997. With $\propto$=0.05, the critical value $F(2,10)$ is 4.10 (Demsar, 2006). The null-[hypothesis is rejected as the critical value is lower than the respective statistics ($F_{\propto=0.05}(2,10) < F_F$). The null hypothesis is rejected, so a post-hoc test such as the Bonferroni-Dunn test is performed (Demsar, 2006; Dunn, 1961). To detect significant pairwise differences of Ranks between the control classifier (ABODT) and all other classifier algorithms, Critical Difference (CD) is calculated to be: 1.3528 (Demsar, 2006). Critical Difference (CD) remains lower the pairwise differences of Ranks between the control classifier (ABODT) and all other classifiers. It indicates that the proposed algorithm of decision tree optimization (ABODT) performs significantly better compared to other compared classifiers.

6.4. Suitability and limitations of ABODT

Swarm intelligence algorithms are found to be promising to solve complex optimization problems (Abualigah, 2020). African buffalo Optimization algorithm is one of the nonparametric algorithm used in recent years for optimization. The proposed ABODT algorithm is based on African Buffalo Optimization to create efficient and optimized decision trees. This method increases the decision-making ability of a decision tree classifier and globally optimizes a decision tree. Another advantage of ABODT algorithm is that it is suitable for any size and type of dataset. When the classifier is of small size then the space required to save a trained classifier is less and the prediction time is also get reduced. So ABODT creates small-sized decision trees, which have less time and space complexity. One of the main advantages of the proposed methods is that it avoids overfitting of a tree when a decision tree is created from a large dataset.

Optimization of decision trees is done sequentially by migrating each node from source space to target space. If Dataset is large then the tree becomes very large and more nodes are created. To create an optimized tree using the proposed method is time consuming to shift nodes from one position to another sequentially.

7. Conclusion

The excellent performance of the ABO is proof of the fact that ABO has excellent potential in solving optimization problems using comparatively fewer constraints than most other meta-heuristic algorithms. ABO is a stable, efficient and non-convergence algorithm. To tackle the main problems of decision trees like local decisions, unstable nature and overfitting are overcome using a modified ABO algorithm. ABO algorithm is modified using the migration feature of African buffalos. In this paper, the proposed ABODT algorithm is applied to the decision trees to create an optimized and efficient tree. The main contribution of ABODT is that it creates globally optimized trees, which are stable and avoid overfitting. Experiments on 15 datasets show the extraordinary performance of innovative and creative ABODT algorithm to obtain optimal or near-optimal decision trees. From results, it is proved that the modified ABO algorithm with extraordinary features of African buffalos produces a decision tree that is more accurate and small-sized. The accuracy of the optimized decision tree is increased by 1% to 8% and the size of a decision tree is also reduced.

As ABODT performs optimization of decision tree by migrating nodes so requires more time. In the future, we are applying this ABO algorithm to the random forest to minimize time complexity and improve the performance of each decision tree in the forest. Also, ABO algorithm can be tested for various parameters to improve the performance of ABODT.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abualigah, L.M.Q., 2019. Feature selection and enhanced krill herd algorithm for text document clustering. Springer, Berlin, pp. 1–165.
Abualigah, L., 2020. Multi-verse optimizer algorithm: A comprehensive survey of its results, variants, and applications. Neural Comput. Appl. 32, 12381–12401.
Abualigah, L., Diabat, A., 2020. A comprehensive survey of the Grasshopper optimization algorithm: results, variants, and applications. Neural Comput. Appl. 32, 15533–15556.
Abualigah, L., Shehab, M., Alshinwan, M., Mirjalili, S., Abd Elaziz, M., 2020. Ant Lion Optimizer: A comprehensive survey of its variants and applications. Arch. Comput. Methods Eng.
Abualigah, L., Shehab, M., Alshinwan, M., Mirjalili, S., 2020. Salp swarm algorithm: A comprehensive survey. Neural Comput. Appl. 32, 11195–11215.
Alweshah, M., Rababa, L., Ryalat, M.H., Momani, A.l., Ababneh, M.F., 2020. African buffalo algorithm: Training the probabilistic neural network to solve classification problems. J. King Saud Univ.-Comput. Inform. Sci.
Anwar, S., Azrag, M.A.K., 2015. In: A comparative study of African buffalo optimization and randomized insertion algorithm for asymmetric travelling salesman's problem. IEEE, pp. 90–95.
Bennett, K.P., 1992. Decision tree construction via linear programming. In: Evans, M. (Ed.), Proceedings of the 4th midwest artificial intelligence and cognitive science society conference, pp. 97–101.
Bennett, K.P., 1994. Global tree optimization: A non-greedy decision tree algorithm. Comput. Sci. Stat., 156
K. P. Bennett, J. A. Blue, "Optimal decision trees", Rensselaer Polytechnic Institute Math Report, pp. 214-224, 1996.
Bera, S., Das, S.K., Karati, A., 2020. Intelligent Routing in Wireless Sensor Network Based on African Buffalo Optimization. In: Nature Inspired Computing for Wireless Sensor Networks. Springer, Singapore, pp. 119–142.
Bertsimas, D., Dunn, J., 2017. Optimal classification trees. Mach. Learn. 106 (7), 1039–1082.
R. Bhosale and N. Chaudhari, "Real time enhanced speech recognition technique to operate computer system using SVM," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, pp. 529-533, 2017 doi: 10.1109/RTEICT.2017.8256653.
Bhosale, R.S., Chaudhari, N.S., 2019. Reduction of Time for speech recognition system by shrinking Adaptive layers of ADAG for Real Time System Using ADAT SVM. Int. J. Eng. Adv. Technol. (IJEAT) 9 (1), 4367–4371.
Bhosale, R.S., Chaudhari, N.S., 2019. Accelerating speech recognition system by adam optimization and CNN for Real Time System using GPU. Int. J. Control Autom. 12 (4), 11–19.

Boryczka, U., Kozak, J., 2010. In: Ant colony decision trees–a new method for constructing decision trees based on ant colony optimization. Springer, Berlin, Heidelberg, pp. 373–382.
Boryczka, U., Kozak, J., 2015. Enhancing the effectiveness of ant colony decision tree algorithms by co-learning. Appl. Soft Comput. 30, 166–178.
Brezočnik, L., Fister, I., Podgorelec, V., 2018. Swarm intelligence algorithms for feature selection: A review. Appl. Sci. 8 (9), 152.
Brieman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Chapman & Hall/CRC.
Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.
Dunn, O.J., 1961. Multiple comparisons among means. J. Am. Stat. Assoc. 56, 52–64.
El-Ashmawi, H., 2018. Walaa, ''An Improved African Buffalo Optimization Algorithm for Collaborative Team Formation in Social Network''. Int. J. Inf. Technol. Computer Sci. 5, 16–29.
Esposito, F., Malerba, D., Semeraro, G., 1997. A comparative analysis of methods for pruning decision trees. IEEE Trans. Pattern Anal. Mach. Intell. 19, 476–491.
Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Statist. Assoc. 32, 675–701.
Friedman, M., 1940. A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. 11, 86–92.
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: An update. SIGKDD Explorations.
Han, J., Kamber, M., 2006. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers.
Iman, L., Davenport, J.M., 1980. ''Approximations of the critical region of the friedman statistic'', *Communications*. Statistics, 571–595.
Jr, I. Fister, X. S.Yang, I. Fister, J. Brest & D. Fister, '' A brief review of nature-inspired algorithms for optimization'', arXiv preprint arXiv:1307.4186, 2013.
Laurent, H., Rivest, R.L., 1976. Constructing optimal binary decision trees is NP-complete. Inform. Process. Lett. 5 (1), 15–17.
Li, T., Zhang, J., Aug. 2008. Asymptotically optimal decentralized control for large population stochastic multiagent systems. IEEE Trans. Autom. Control 53 (7), 1643–1660. https://doi.org/10.1109/TAC.2008.929370.
M. Lichman, UCI machine learning repository, 2013 http://archive.ics.uci.edu/ml.
Odili, J.B., Kahar, M.N., Anwar, S., 2015. African buffalo optimization: A swarm-intelligence technique. Procedia Computer Sci. 76, 443–448.
Odili, J.B., Kahar, M.N.M., 2015. ''Numerical function optimization solutions using the African buffalo optimization algorithm (ABO)''. J. Adv. Math. Computer Sci., 1–12
Odili, J.B., Kahar, M.N.M., Noraziah, A., 2016. Convergence analysis of the African buffalo optimization algorithm. Int. J. Simul.: Syst. Sci. Technol. 17 (44), 44–141.
Odili, J.B., Kahar, M.N., Noraziah, A., Zarina, M., Haq, R.U., 2017. Performance Analyses of Nature-inspired Algorithms on the Traveling Salesman's Problems for Strategic Management. Intell. Autom. Soft Comput., 1–11
Otero, F.E., Freitas, A.A., Johnson, C.G., 2012. Inducing decision trees with an ant colony optimization algorithm. Appl. Soft Comput. 12 (11), 3615–3626.
Padmapriya, R., Maheswari, D., 2017. ''Channel allocation optimization using african buffalo optimization-super vector machine for networks''. Asian J. Inform. Technol., 783–788
Parpinelli, R.S., Lopes, H.S., Freitas, A.A., Aug. 2002. Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. 6 (4), 321–332. https://doi.org/10.1109/TEVC.2002.802452.
Payne, H.J., Meisel, W.S., 1977. An algorithm for constructing optimal binary decision trees. IEEE Trans. Comput. 100 (9), 905–916.
Quinlan, J.R., 1986. Induction of decision trees. Mach. Learn. 1, 86–106.
Quinlan, J.R., 1993. C4.5: Programs Programming for Machine Learning. Morgan Kaufman, San Francisco.
Singh, P., Meena, N.K., Slowik, A., Bishnoi, S.K., 2020. Modified african buffalo optimization for strategic integration of battery energy storage in distribution networks. IEEE Access 8, 14289–14301. https://doi.org/10.1109/ACCESS.2020.2966571.
Son, N.H., 1998. From optimal hyperplanes to optimal decision trees. Fundamenta Informaticae 34, 145–174.
Tzirakis, P., Tjortjis, C., 2016. T3c: Improving a decision tree classification algorithms interval splits on continuous attributes. Adv. Data Anal. Classification, 1–18.
Windeatt, T., Ardeshir, G., 2001. ''An empirical comparison of pruning methods for ensemble classifiers'', *in Proc.* In: of Forth International Conference on Advances in Intelligent Data Analysis, pp. 208–217.
Yang, X.S., 2014. Nature-inspired optimization algorithms. Elsevier, pp. 20–27.