International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 15, No.1 (Jan-24)

http://dx.doi.org/10.12785/ijcds/150109

Decision Tree Induction Using Evolutionary Algorithms: A Survey

Maryam H. Bahar [1] **and Hadeel Noori Saad** [2]

[1]*Information Technology College/ Babylon University, Babil, Iraq*
[2]*Department of Computer, Faculty of Education for Women, University of Kufa, Najaf, Iraq*

Received 2 Jun. 2023, Revised 17 Oct. 2023, Accepted 16 Nov. 2023, Published 1 Jan. 2024

Abstract: An evolutionary methods for an induction-based decision trees made a wide step development in machine learning field. In this context, the majority of researches recently concentrates on techniques that use developing decision trees as an alternative to the traditional heuristic top-down divide-and-conquer strategy. Evolutionary algorithms play an important role in improving decision tree classifier parts. The main contributions of our article are twofold, first it provides a survey of evolutionary algorithms with decision trees. Second, it reviews a taxonomy that encompasses techniques mentioned above as a backbone in creating enhanced decision trees, and an evolved construction components of decision trees. The article covering researches in the period 2011-2023, these researches proposed different evolution paradigms encompasses :feature categorization, splitting nodes , complex to simple decision rules, tree size, etc parameters of DT. Finally, a detailed scenarios and results had been analyzed, highlighting the weaknesses and areas of strength with respect to processing time, accuracy, and required space.

Keywords: Classification, Decision tree, Evolutionary algorithms, Genetic algorithms, Machine learning, Survey

1. INTRODUCTION

The candidate solutions that are represented by a population of individuals (referred to as chromosomes) evolve in evolutionary algorithms (EAs). Individuals are able to reproduce and are prone to genetic variations. Better-adapted individuals have a greater probability of surviving and passing to the next generation due to their effectiveness (represented as a fitness function value) influenced by environmental pressure. After a population has been subjected to genetic operators, the resultant individuals are examined for fitness, and then the solutions that will be preserved throughout iteration will be chosen or reproduced by a selection method. A predetermined number of generations or extra intricate stopping criteria are often created as an evolution algorithm starting criteria. It is very clear that EAs was drawn inspiration from biological evolution and do not attempt to reproduce nature completely. The most well-known and significant methods are as follows[1] [2]:

- Genetic algorithms [3], typically work with binary-coded, fixed-size chromosomes. Standard genetic operators are preferred in this representation. Crossover, which allows two individuals to share their genetic material, this is the (core operator) that is frequently implemented on the chromosomes, whereas the mutation operator has a low probability (minor operator).
- Evolutionary strategies [4]: continuous function optimization, use representations that are specific to the problem, like a real values vector. Mutation play as the primary operator, where've its ranges are self-adapting. Based on population sizes and selection criteria, different types of strategies are defined by the relationships between the parent population and the modified individual's population. Parents can compete with mutants or ignore them.
- Genetic programming [5]: was proposed as a mechanism for automating the evolution of computer procedures. A dynamically evolving syntax tree is often used to represent the program. The program's variables and constants (referred to as terminals) are leaves, while the arithmetic operations are represented by the internal nodes (referred to as functions). So, the program's alphabet is determined by its terminals and functions. Many types of specialized discrimination operators can be proposed, with the most major being an exchange of subtrees between two trees and a mutation in a subtree.

Evolutionary programming [6]: An often detailed representation of the problem domain (originally, individuals that can emulate finite state machines). Similar to evolutionary techniques for optimizing real value vectors, a series of

E-mail address: maryam@itnet.uobabylon.edu.iq, hadeel.jrew@uokufa.edu.iq

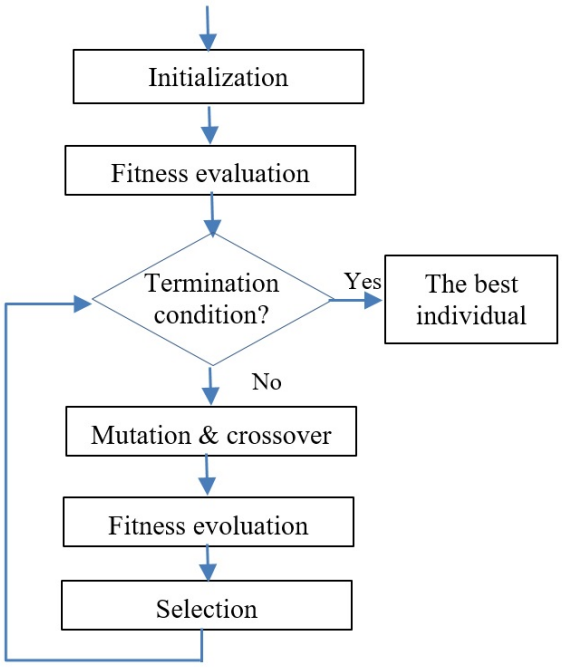

Figure 1. Typical evolutionary (genetic) algorithm process [7]

real values is developed. New individuals can only be produced through mutations, which often involve inserting random variations from a predetermined distribution to a parents. Figure1 shows a schematic representation of the EA's standard iterative procedure. Although the specific realizations may alter significantly, the essential elements and actions remain essentially constant [7]. At the beginning of the procedure, the population is initialized. The initial population that arises should be extremely diverse and cover the whole search region, because this will surely assist evolution. Normally, a random number might be utilized to generate an individual. The algorithm's main loop can start after these new individuals have their fitness functions evaluated. Selection of individuals for reproduction is followed by the employment of differentiation operators, evaluation of the altered or newly created individuals, and succession when the next population is formed. At the end of each generation, a termination condition is checked. Imposing time limitations or simulating a specific number of iterations is the most basic situation. When the anticipated pattern is satisfied, the process is ended for more complex solutions, where specific population attributes committed to the algorithm's convergence may be seen. The individual with the highest finding of fitness is the algorithm's final output. Genetic operators are used to separate the individuals in each generation. There are two common operators: crossover and mutation. The main mechanism of the evolutionary search is individual differentiation, which should ensure that a good balance between exploitation (diligent searching close to the current place) and exploration (checking out new, frequently distant areas) is retained. The mutation fragmentarily disrupts a single chromosome at random, and more often than not, this causes the original individual to move somewhat in the search space. Obviously, more alterations are possible if the modifications reach the most important regions of the chromosomes. In a crossover, since the parents are often fully recombined, naturally, the number of mutations is greater than in a local mutation; hence, the children are often more unique from their parents. This results in remote transpositions, which may be useful when it's crucial to explore new territory. Selection mechanisms are the primary factors that drive this kind of simulated evolution. Individuals who are more fit and, hence, better acclimated to their surroundings should have a higher probability of survival than those who are less fit. The descendants will only be produced by individuals who have been chosen and reproduced. This mechanism has a significant impact on search strategies and may determine how efficiently and effectively evolution proceeds. It's important to find a decent balance between ensuring that a population is sufficiently diverse and elevating the most talented individuals. If the best individuals are given too much attention, only locally optimal solutions may converge too quickly. The search will move much more slowly and take up more computer resources if the better contenders aren't given enough encouragement. Although maintaining the proper balance throughout the algorithm's various phases is undoubtedly a difficult endeavor, selection algorithms created over time have made it possible to accomplish this goal [7]. Going above and beyond to create a specialized EA is frequently advantageous when trying to apply the evolutionary approach to a particular situation. The best outcomes are attained if the algorithm can be modified to take into account the specifics of the problem [2] and use this information to choose the elements that are most appropriate for the problem from among a large variety of options. This makes it possible to significantly reduce the search space in many applications, which accelerates the search for superior solutions. It is important to select the values (from the standard technique) of various regular as well as additional control parameters in such a unique way, as is desired (concerning the specialized elements). Despite assertion that EAs are rather resistant to tiny changes in fundamental parameter values, the optimal settings are typically discovered through many experiments with various parameter values. Although automatic tuning is predicted to be more effective than manual tuning, it obviously requires more processing power. In EAs, many methods of parameter control are addressed [8]. The following paragraphs will explore the roadmap of this article, where the attention is focused on decision tree, and EAS constructions (area of interest). The rest of the paper will organized as follows : Advantage and disadvantage of implementing evolution algorithms on decision trees will discussed to clarify the limitations of the of each case construction, section 6 gives a detailed review that covering the state-of-the- art researches under considerations, these researches implements evolution strategies with decision trees.

2. DECISION TREE

A decision tree (DT) is a classifier that is a tree representation in a form that resembles a flowchart. Because of their understandable nature and resemblance to human reasoning, DTs are frequently used to illustrate classification models. In comparison to other learning algorithms, DT induction algorithms provide a number of pros, including resistance to noise, a light computation's cost for model construction, and low capacity to handle redundant features. Additionally, the induced model typically has a strong capacity for generalization [9]. The majority of DT induction methods use a top-down construct, greedy, recursive partitioning technique. Several impurity metrics, like information gain, are used to select a discriminating property that is associated with an internal node, gain ratio, Gini index, and distance-based metrics. The fact that a greedy search frequently results in less-than-ideal solutions is a significant disadvantage. Additionally, small data sets for attribute selection at a tree's deepest nodes due to recursive data partitioning may result in data overfitting. The insertion of an ensemble of trees is one of the solutions that have been suggested to address these issues. The final classification is usually determined via a voting system, and ensembles are produced by inducing several trees from trained samples, Ensembles have the drawback of losing the simplicity of understanding the analysis of a single DT. Indeed, it is frequently the case that categorization models merged in an ensemble are somewhat inconsistent with one another; this inconsistency is important to raise the ensemble's forecast accuracy. So, where comprehensibility is important; ensembles aren't the good choice for applications. An induced DTs are becoming more popular because they explore global solutions rather than local ones and manage attribute interactions better than greedy methods [10].

3. DTS- EAS CONSTRUCTION

Evolution Algorithms for evolving DTs, must be used with knowledge of both processes, Evolution Algorithms and DT. The whole induction of the DT process is seen in Figure 2. The procedure is broken down into three stages: the preinduction stage, which involves the production of data for use in the induction and evaluation of DTs; the induction stage, which involves inducing DTs with the help of the prepared data; and the postinduction stage, which involves optimizing the DTs that have been induced [11]. The standard procedure for using EAs to resolve a problem is shown in Figure 3. Setting up the environment comes first, followed by selecting the genetic operators, control parameters. The initial population of humans can be produced after the setup in order to begin the evolutionary process. The evolutionary cycle then begins, in which every individual assessed using the fitness function is defined, a choice is made based on the evaluation of fitness, and the selected individuals go through cross-over and mutation to produce offspring, which act as placeholders for the subsequent generation of individuals. The evolutionary process keeps going until the specified termination conditions are satisfied. The fitness function is crucial for determining each

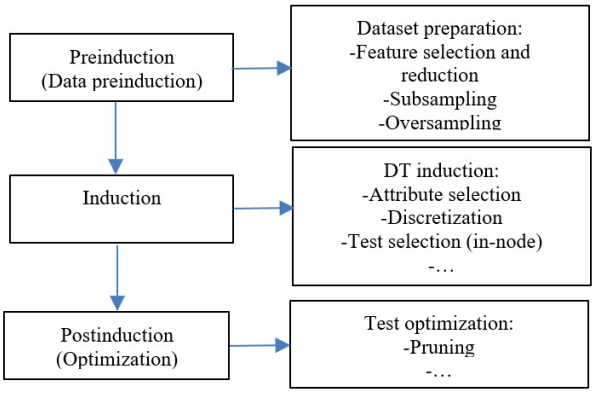

Figure 2. The overall construction method for DTs. [11]

individual's quality and, as a result, for arriving at a globally ideal solution. Given that DTs are classifiers, classification accuracy is the obvious fitness function. The fitness function typically has multiple objectives (accuracy, size, cost, etc.) [11].

4. THE ADVANTAGES OF EAS WITH DT INDUCTION

The main advantage of evolving DTs is their ability to avoid the local optimum. EAs are less likely to converge on local optima since they can conduct a strong search in the space of potential solutions to reach a global minimum. Additionally, as a result of this global search, EAs typically handle attribute interactions better than greedy techniques [10], allowing them to identify intricate attribute correlations that the greedy evaluation method missed. The ability to intentionally bias the search space through multi-objective optimization is another benefit of evolving DTs. The ability of EAs to organically optimize several objectives may be essential in a number of application domains. Cost-sensitive classification, which is one of the major medical classification issues, may benefit from optimizing strategies that address multiple mistake costs. Another crucial component that may be easily added to a multi-objective EA for DT induction is parsimony pressure [11].

5. THE DISADVANTAGE OF EAS IN DT INDUCTION

Time restrictions are the fundamental drawback of the evolutionary induction of DTs. EAs are a reliable but expensive computational heuristic. Nevertheless, as time went on, quicker processing resources made it possible for EAs to be used in a wider range of applications. Because of advancements in parallel processing, EAs can now be better examined within reasonable execution times. The time required to prepare the data for data mining purposes typically takes far longer in real-world applications than the time required to induce a classification or regression model. As a result, the lengthy processing times of EAs are not always the process's bottleneck in real-world applications. The size of parameters that must be achieved in order

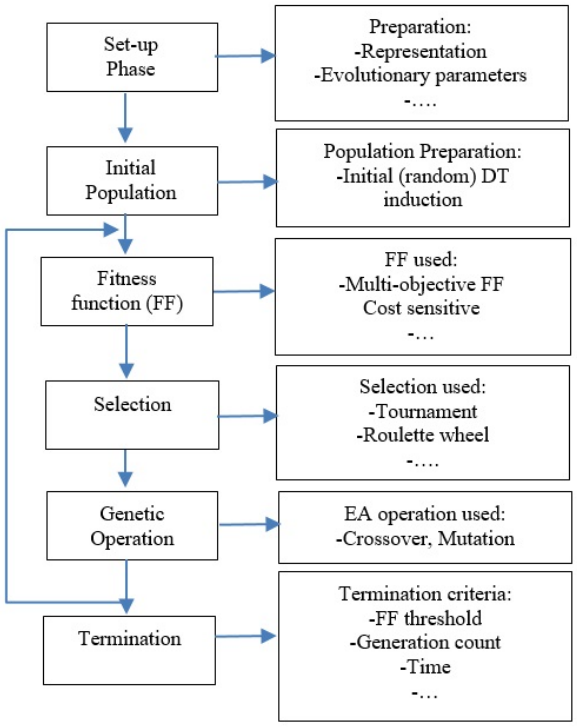

Figure 3. EAs' fundamental method (for DT construction) [11])

to conduct a complete EA settings is another drawback. Although they believe it to be a secondary issue, Espejo et al. [12] acknowledge this issue in GP-based classifiers. Since genetic algorithms and genetic programming share many of the same factors, the same problem arises for them [13].

6. REVIEW

In this section, we provide a comprehensive survey of evolutionary algorithms and their application to decision trees, which can serve as a valuable resource for researchers and practitioners in the field, and present a taxonomy of techniques related to decision tree construction, including those that enhance decision trees and evolve their components, which can help in organizing and understanding the different approaches used in the literature. R. C. Barros, M. P. Basgalupp, et al. [13] presented a survey of evolutionary algorithms designed for decision tree induction from the beginning until 2011. Which this study covers a period from 2011 to 2023, which ensures a comprehensive analysis of the relevant literature in the domain of evolutionary algorithms and decision trees. This approach allows the authors to present a detailed and up-to-date review of the state-of-the-art techniques. Work in [14] proposed Mutual information and the t-statistic. They used the best 10 and 20 genes to evaluate GP and DT predictions, as well as those made using genetic programming and evolved DTs. Genetic programming fared better as a classifier for this set of data based on mutual information-based feature selection and

TABLE I. DISCIPULS-Based Genetic Programming Parameters [14]

Parameter	Value
Size of population	500
Frequency of mutation	95%
Frequency of recombination	50%
Size of max program	512 bytes

TABLE II. GA-Tree genetically evolved DT parameters [14]

Parameter	Value
Generations	100
Population	100
Probability of cross over	0.99
Probability of mutation	0.01

the area under the receiver operating characteristic curve (AUC). The DISCIPULUS genetic programming system was used (Demo Version). They classified the datasets using the GP tool after applying the feature selection criteria (t-statistics and mutual information). The majority of the parameters were configured using the software's default (preset) settings (Table 1). The settings include the specified numbers for the random population size, maximum program size, mutation rate, and recombination rate. While the programs are evolving, the GP tool includes the opportunity to rank the attributes. This feature was utilized to cross-check the accuracy of forecasts for the top 10 and 20 genes based on t-statistics and MIFS's top 50 genes. This was accomplished by selecting subsets with the highest input impact values. After characteristics were determined using the t-statistic and mutual information, when using the program, the majority of the default options (preset parameters) were utilized (see Table 2). (Prefer More Accurate Trees) is a tree size parameter. The data was pre-processed using colon cancer benchmark dataset downloaded from the (Kent Ridge Biomedical Datasets) website. Before using a classifier, the raw data was normalized for feature selection. Two techniques were used to pick the features: mutual information using non standardized data and t-statistics using standardized data. The 50, 20, and 10 best genes from each feature selection approach were fed into genetic programming and genetically evolved DTs, respectively. When the prediction accuracy of genetically developed DTs was evaluated, mutual information feature selection-based genes outperformed t-statistic-selected genes. The prediction accuracy for the 10 best genes selected based on mutual information 88.33% is the highest among genetic-based evolved DTs associated with t-statistic feature selection and mutual information techniques. Particularly when applied to the colon cancer dataset, genetic programming performs better overall than genetically evolved DTs. The authors in [15] focus on evolutionary DTs (EDTs) to tackle medical problems. The majority of the unique evolutionary DT construction algorithms are investigated on an intensive data sets from diverse areas using data sets from the UCI library.

They showed that tree's individual nodes were optimized to result in the formation of very complex nodes. As a result, it becomes considerably more challenging to analyze individual decisions (at each node). Where, simple conditions are simpler to understand than complex neural networks or calculations. Such method reduces a DT classifier's core features (readability) and turns it into a (black box) classifier. Black-box classifiers are generally less appropriate for usage in the medical industry. The improvement of the training set selection procedure and DT construction are two crucial study areas for employing evolution to improve DT performance for medical applications, requires scaling issues with enormous data sets, selecting training data chunks, representations of training sets inspired by evolution, imbalanced data sets, and sampling training data features are all investigated in [15] related to evolutionary optimization (EO) of training sets. The research has shown an improvement on the training set's evolutionary optimization which enhanced the performance of the resulting classifier. Furthermore, it has little effect on the critical components of a DT classifier (comprehensiveness, high dimensionality, etc.). With many trials, the resulting trees are pruned and perform at least as well as classically formed DTs. Even if the identical algorithm and genetic parameter values are used in each run, the outcome may differ since the evolutionary process contains uncertainty. This technique required numerous executions in order to evaluate the results. In [11], the authors analyze the evolutionary design of DTs with the most prevalent strategies. They take a step-by-step case study that uses data from (UCI) machine learning repository initially describes the general technique before using it to show how it works. This case study employs the provided methods in an instructional and clear case study in order to provide a fundamental overview of the evolutionary approach to DT creation, in addition to outlining the most significant advancements and methodologies in this field. Numerous solutions have been found, with just minor differences between all of their features (accuracy, tree size, used attributes, etc.). They chose one of these methods because it balanced accuracy and complexity well (number of nodes, depth, and different attributes utilized). The test set of unforeseen events with 19 decision nodes is 85.63% accurate (leaves). It is fascinating to compare this result to that of other well-used categorization techniques. The UCI repository provides classification data (classification accuracy) for 16 different algorithms, 11 of which were employed in their experiment using the original train/test split. Their solution placed third with 85.63%, whereas FSS NB and NBTree, two modified Naive-Bayes approaches, obtained the best classification accuracy 84.95% and 84.90%, respectively. All the other methods earned lower scores than the k-Nearest Neighbor algorithm, which had the lowest accuracy 78.58 percent. The current classical DT induction processes achieved an accuracy of 85.54 percent when using C4.5 algorithm, on WEKA benchmark for classical, statistical DT induction. The J48 method induced a DT for another comparison (the source code of the C4.5 algorithm in Weka, which is used

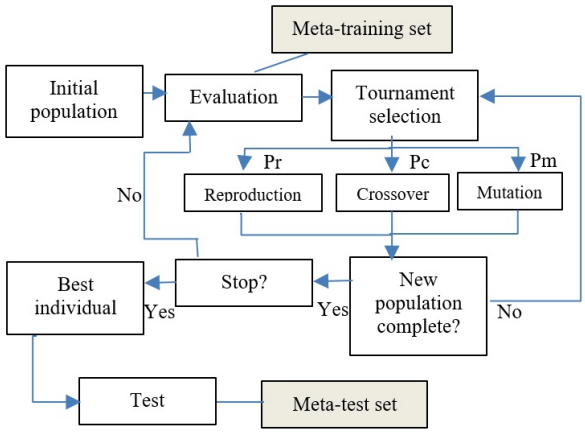

Figure 4. HEAD-DT evolutionary scheme [16].

widely). 564 decision nodes and 85.84% test-set accuracy their evolutionary-created DT is simpler (3% of traditionally produced DT) and almost as accurate (only a 0.21 percent difference). They may claim that the evolutionary process provides a better solution because inducing DTs produces a highly trustworthy (high accuracy) and simple (pruned) tree. In [16] suggest HEAD-DT, a hyper-heuristic EA, evolves top-down decision-tree's components induction techniques. A new paradigm DT research, in which an automatic design decision-tree induction algorithms learned a certain sort of classification data sets (or application domain). Hyper-heuristics are defined as an automated way for choosing or creating heuristics that address difficult computational search issues. In actuality, hyper-heuristics have the power to build new heuristics that are appropriate for a certain problem or class of problems on their own. An EA is used to combine human-designed heuristic components, or (building blocks). Hyper-heuristics were created to expand the number of search techniques available. Instead of utilizing a standard meta-heuristic technique, an EA is used to find the optimal decision-tree induction algorithm that can be successfully applied to a range of classification issues, increasing the generality level in the context of DTs. HEAD-DT is comparable to a traditional generational EA in which decision-tree induction techniques are coded as collections of top-down individuals. Figure 4 illustrates its evolutionary strategy. Use common EA operators, such as competition selection and genetic operators (mutation, crossover, and reproduction) that are mutually exclusive, and an established stopping condition that puts an end to evolution after a certain number of generations. Each gene in a HEAD-DT individual can have a value within a specific range because they are integer vectors (see Figure 5). Four gene categories were identified: Split, stopping criteria, missing value, and pruning genes are presented to summarize the design elements of a top-down decision-tree induction technique. HEAD-DT should design a new, maybe better DT algorithm for a certain application area. HEAD-performance DTs were compared to C4.5, CART,

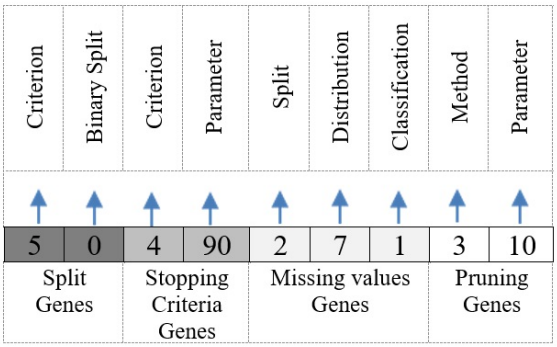

Figure 5. Linear-genome for evolving DT algorithms [16].

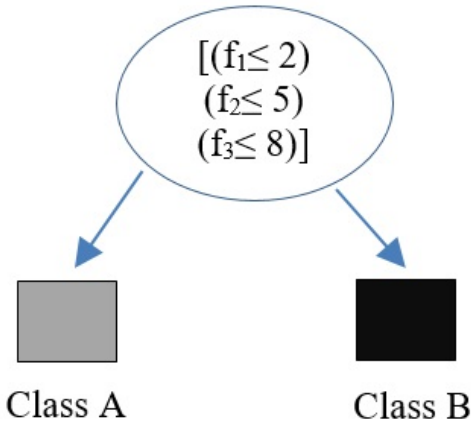

Figure 6. A multi-test split illustration that includes a number of univariate tests [17]

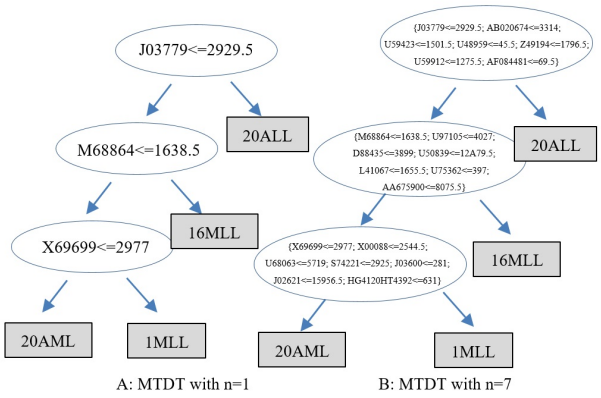

Figure 7. MTDT in a single node with N=1 and N=7 tests [17].

and REPTree utilizing microarray gene expressions and 35 real-world data sets. In terms of predicted accuracy and F-measure, HEAD-DT algorithms beat manually created decision-tree algorithms. In [17] Gives an introduction to the concept of multi-test DTs (MTDT), a new, strong language for representing *DT s*. A multi-test tree's structure is identical to that of a typical DT, such as *C*4.5. Every split in non-leaf nodes of a multi-test tree is referred to as a multi-test split and is made up of a collection of univariate tests. These simple tests are univariate and when combined demonstrate how our strategy differs significantly from conventional multivariate splits, such as oblique splits. All univariate test components have the same weight during classification, and the majority voting process determines how the MTDT splitting criterion is applied. A multi-test that separates the data in the node into Class A and Class B is depicted in Figure 6. The three different attribute tests in this multi-test are $[(f1 \leq 2), (f2 \leq 5), (f3 \leq 8)]$. In this example, the majority voting process resulted in at least two out of the three univariate tests being statistically significant. Figure 7 shows actual trees. Although both induced trees correctly classified cases from the training set, they performed very differently on the test set. Even in cases where there is no effect on later multi-tests and both alternate trees have identical primary tests, the suggested strategy is effective. Surrogates' effect on multi-test decision-making explains MTDT's good performance with N = 7. The surrogate tests $mt1,j(1 < j < N)$ in MTDT with $N = 7$ must outvote the primary tests mt1,1 in the nodes and correctly categorize 6 out of 15 cases. This increased the Armstrong dataset classification accuracy from 86% to 100%. This study offers a multi-test DT classification method for categorizing gene expression data. A unique splitting criterion was developed to improve classification accuracy and reduce DT underfit on specific types of data. The suggested method highly competitive with all tested competitors. It was determined through analysis of actual microarray data that the biological literature backed up the information learned through MTDT. By using this (white box) method, scientists can develop precise and biologically relevant classification models and find brand-new regularities in biological data. The crucial algorithmic traits of our strategy were uncovered by our extensive empirical inquiry and evaluated from a machine learning standpoint. In the article [18] a multi-objective evolutionary method EVO-Tree (EA for DT Induction) has been proposed. It gives new trend of grow a binary DTs for categorization, When more than one target needs to be optimized, various objectives are mapped into one multi-objective function using a predefined weight. A single solution is obtained in a single run using such a weighted aggregation. EVO-Tree uses an EA-based training approach that analyses a population of potential tree architectures that have been encoded into chromosomes that resemble trees. The misclassification rate and tree size are two objective factors that the training procedure seeks to minimize. Utilizing multiple open UCI data sets, evaluate the EVO-tree ability and contrast the outcomes with various cutting-edge classification algorithms. The size and accuracy of trees are chosen as the global metrics for the EVO-Tree algorithm. By removing parts of a classifier that might be based on false data, this method reduces the complexity of the final classifier without lowering predicted accuracy. Additionally, rather than focusing on how to

induce a tree, one may focus on the conditions that a tree must meet (which impurity measure to select, how to prune, etc.). The authors in [19] create and put into practice a parallelization of evolutionary induction of DTs based on graphics processing units (GPUs). They use a programming paradigm for the Compute Unified Device Architecture, which provides general-purpose processing on a GPU (GPGPU). While the evaluation of the population's members is carried out in parallel, there is still a sequential execution part comprises selection and genetic operators. The data-parallel technique is used, distributing the components of a dataset across GPU cores. The assigned portion of the data is processed by each core. Finally, the output from each GPU core is combined, and the CPU is informed of the desired tree metrics. Experimental validation of the suggested approach's computational performance using synthetic and real-world datasets. Evolutionary induction of DTs assisted by GPGPU can be greatly sped up (even up to 800 times), which enables processing of much bigger datasets, as compared to the standard CPU version. This study suggests parallelizing DT's evolutionary induction using GPUs. They focus on classification, and they are particularly interested in evolutionary-induced univariate categorization trees. To their knowledge no study has been conducted on how to employ GPGPU to accelerate the evolution of DTs. Although the GPU computational model differs from the traditional CPU model, the method they use is comparable to the master-slave paradigm. EA phases are carried out by the CPU (master), which delegates the computation-intensive duties to the GPU. On its cores, which are essentially slaves, the GPU conducts the jobs in parallel. In this manner, the original sequential algorithm is maintained while preserving so-called global parallelism [20]. The proposed method is used in conjunction with a system known as a global DT (GDT). Finance [21] and medicine [22] are two real-world applications for the GDT solution paradigm. Its framework can be utilized for evolutionary classification induction [23] and regression trees [24]. The primary goals of this effort are to speed up the GDT system and enable effective DT evolution on massive data sets. For these reasons, the suggested parallelization makes successful use of the ability of contemporary GPUs to perform computationally heavy tasks like fitness computation and leaves the CPU in charge of the evolutionary flow management and communication chores. A hybrid MPI+OpenMP strategy was used in earlier attempts to parallelize the GDT solution [25].This is an alternative parallel paradigm. To expand the GDT system, they suggest GPU-based parallelization in this study. The required induction time can be reduced by more than two orders of magnitude with their technique, even on a standard PC with a medium-class graphics card. The studies done on the supplied fake and real-world datasets demonstrate that our method is quick, scalable, and capable of exploring big data. In [26], suggested method known as the Evolutionary Multi-Test Tree (EMTTree), which has the ability of structure self-adaptation with the data being examined at the time, is described. Their solution's better model stability and higher forecast accuracy are without a doubt its greatest strengths. The long tree induction time and a few adjustable input parameters are the EMTTree's minor flaws, which come from utilizing an evolutionary technique. The number of parameters that must be adjusted is small, and gene expression data are still scarce. One of the EMTTree framework's features are evolutionary tree induction as an alternative to top-down greedy algorithm. Using this comprehensive strategy, they were able to avoid the problematic pruning procedure and concurrently search for tree structure and multi-test splits, generating a novel algorithm in the process: The concept of gene clusters is formed, and a new dimension to the multi-test is provided by combining local optimizations with specialized EA to find the most uniform multi-tests, including the top-ranked genes: Every single-variable test in a multi-variable test has a unique fitness function that favours minimizing tree error over minimizing tree size, as is the typical strategy for DT. This fitness function contains information about the ability of genes to discriminate. They also include information on gene ranking and split resemblance to prevent the predictor from stuck in problem of underfitting/overfitting to data, especially in the bottom regions of the tree. Figure 8 depicts the flowchart for the EMTTree technique. According to a comprehensive series of computational studies employing 35 real-world data sets in gene-expression, one of the top DT-like classifiers for gene expression data at the moment seems to be the EMTTree method. Importantly, EMTTree keeps the predictive structures' patterns understandable while not significantly expanding the tree's overall complexity. In [27] Since feature indices are also encoded using real values and decoded using the finding the minimal operation, they provide a method that transforms a DT into a real-valued homogenous vector. With this method, a variety of optimization techniques, including differential evolution and evolution strategies can be used. Provide a novel method in this study for building axis-parallel DTs for classification issues utilizing EAs. Each node in axis-parallel trees divides the dataset in accordance with the following principle:

$$f(x) = \begin{cases} 1 & if a_i \leq t \\ 0 & otherwise \end{cases} \quad (1)$$

The two parameters that describe each node of the tree are the threshold value and the index of a feature. Consider the case when they have a real-valued vector with a fixed length and values between [0, 1]. This vector's two equally long halves, the first of which holds feature indices whereas the second encodes threshold values. Keep in mind that all object features fall between [0, 1]. If not, they Normalize the features. Finding the point in the first section of the vector with the least value allows to determine the index of a feature by figuring the remainder after integer division by the number of features. The threshold value for the pertinent node is represented by the value of the second component of the vector at this point. The following step is to identify a threshold value, the vector's next-smallest value, and the

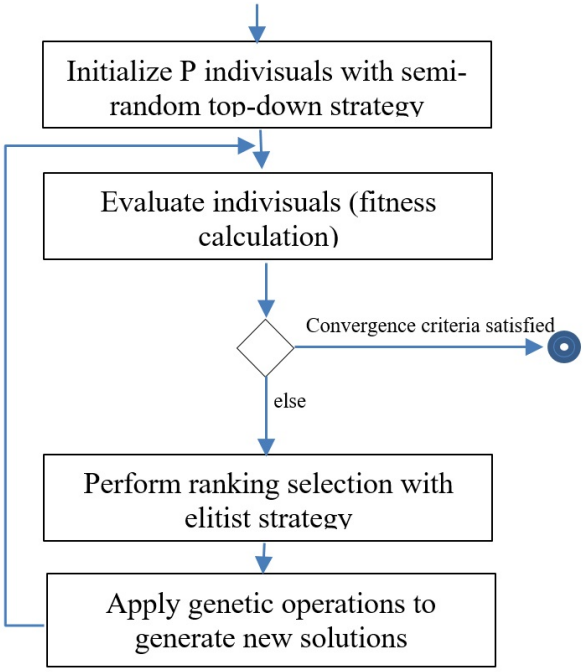

Figure 8. MTDT in a single node with N=1 and N=7 tests [26].

accompanying feature index in the node. This procedure is continued until the entire vector has been utilized. A DT without leaves can be made by successively connecting the nodes while using the feature indices and their threshold values for each node. The training dataset and the majority rule are then used to add leaves to the DT. Consequently, they can build a DT from a real-valued vector and assess its properties. The population under the method of evolution strategies, in contrast to the method of differential evolution [28], consists of single individual:

$$p \sim x \tag{2}$$

The following relation gives the individual initialization:

$$x_j = x_j^{min} + r(x_j^{max} - x_j^{min}) \tag{3}$$

$r \in [0, 1]$: Uniformly random number distributed. Then, they move the individual in the gradient-approximating direction of the weighted total offsets.

$$x \leftarrow x + \alpha \frac{1}{n\delta} \sum_{i=1}^{n} f(x + \delta e_i) e_i \tag{4}$$

α and σ constants that user-specified. Bagging and boosting are two of the most commonly used methods for creating DT ensembles. A technique that employs a bagging approach is Random Forest, whereas a technique that uses a boosting approach is AdaBoost. They suggest using the previously published EAs to replace the traditional algorithms used in these methods to infer DTs. As a result, they now have two new methods: evolutionary boost (EvoBoost) and evolutionary random forest (EvoRF), which are analogs of AdaBoost and Random Forest, respectively. Additionally, they consider the (EvoEnsemble) method, which combines the vectors from each ensemble tree to represent the ensemble as a whole by having each population member act as a representative of the entire ensemble. This method, evolutionary ensemble, optimizes the entire ensemble simultaneously, so theoretically a better result should occur. On prominent datasets from the UCI repository, the suggested algorithms outperform conventional methods like CART, random forest, and AdaBoost in terms of quality, but they require more time to produce comparable results. This is because, in contrast to classical algorithms, techniques using EAs build trees and assess their quality multiple times during training. In [28], they suggested nonlinear DT (NLDT) to represent a nonlinear function of features at each non-terminal conditional node, which would be used to build a split-rule. The data will be split using each split-rule into two separate, non-overlapping subgroups. The tree is permitted to continue growing until one of the termination conditions is satisfied as these subsets are divided up in ever more hierarchical ways. In their opinion, by making nonlinear split rules available at conditional nodes, a more adaptable DT will be created (instead of the single-variable-based rules included in conventional ID3-based DTs). (Having fewer nodes). Second, a unique bilevel optimization approach is utilized to construct the split rule at a particular conditional node, treating the learning of the rule's structure and related coefficients as two hierarchically connected optimization tasks. Thirdly, the approach they propose is specialized for addressing classification issues, leading to a computationally efficient bilevel optimization procedure. Fourth, to ensure that the rules we obtain are likewise clear, they emphasize the growth of simple rule structures using our unique bilevel optimization method. For their proposed technique, Figure 9 depicts a nonlinear DT (NLDT) as an interpretable classifier. Conditional (or non-terminal) nodes and terminal leaf nodes make up the DT (DT). The results demonstrate the effectiveness of the bilevel-based NLDT strategy for classification problems, which reduces the size of the rules set to a level comparable to the standard DT (for example, CART) and is more straightforward than a single complex rule produced by an SVM algorithm. As a result, their nonlinear rules have a low level of complexity, making them easier to understand and useful for both basic classification tasks and improving classifier comprehension. In the paper [29], DT Improved by Multiple-Splits with EA for Discretization (DIMPLED), a unique EA, is offered as a method for global discretization. The suggested DIMPLED method maintains the proper amount of interpretability with a single DT while gradually enhancing the discretization technique for better performance. Additionally, when used with k-means clustering for DIMPLED enables a tree augmentation with a variety of splits that can be meaningful and understandable for practitioners. The entire structure is first presented, and then each individual phase is thoroughly explained. A synopsis of the suggested DIMPLED framework may be found in Figure 10. The proposed DIMPLED method preserves the

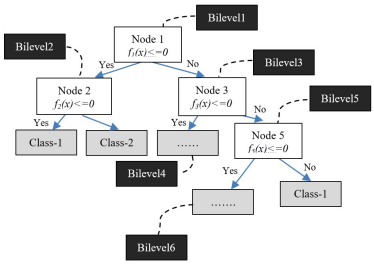

Figure 9. A Non-linear DT (NLDT) classifier is demonstrated for a two-class problem. A specific bilevel-optimization technique is used to evolve the split-rule function fi(x) for a given conditional node i [28].

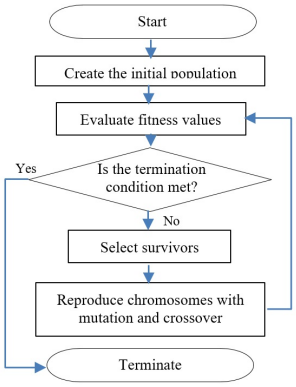

Figure 10. Overall Framework of DIMPLED [29].

appropriate degree of explicability with a single DT while gradually improving the evaluation technique for improved performance. When combined with k-means clustering for global, DIMPLED allows a tree to have many clear and relevant splits. The DT that was enhanced by DIMPLED performed better than single-decision-tree models (C4.5 and CART), which are routinely employed in practice, and it was compete with ensemble approaches, which use several DTs. Experimental findings employing two real-world sensor datasets served as proof of this. Although group techniques may occasionally produce somewhat better performances, the suggested method has a more obvious structure while maintaining a suitable performance level. The authors in [30] suggest a Baldwinian evolutionary strategy to improve the state-action function and the tree's structure at the same time. In contrast to Darwinian evolution and Lamarckian evolution, the evolutionary theory known as Baldwinian evolution holds that an individual's life experiences are not passed on to their descendants. The individual's knowledge may, nevertheless, represent an evolutionary advantage that changes the fitness landscape. They do this by employing an evolutionary approach to evolve the DT's structure and Q-learning to train the state-action function. In this method, they look for trees that divide the state-space in such way that, when the agent takes the best possible actions, the reward is maximized. The Grammatical Evolution is the EA that they employ (GE). Figure 11 displays a block diagram that demonstrates how the proposed approach functions inside. The evolutionary processes that are part of our method are depicted in blue, while the reinforcement-learning processes are shown in red. The findings indicate that the suggested method can produce DTs that are much more interpretable and outperforms the non-interpretable state-of-the-art. The results of this study also suggest that interpretable models may be competitive with cutting-edge approaches and that the generally accepted performance-interpretability trade-off is not necessarily true. This calls for encouraging study in the area. The authors in reference [31] seek to enhance the performance of decision trees (DTs) through the integration of an evolutionary algorithm (EA) with interpretable reinforcement learning (RL) techniques. As the evolutionary algorithm (EA) progresses, it refines the architecture of the decision tree (DT), while the reinforcement learning (RL) algorithm seeks to optimize the actions executed by the terminal nodes. The researchers utilize the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) algorithm to enhance the diversity of hybrid models across a feature space that encompasses both the complexity of the model and its variability in behavior. MAP-Elites, also known as MAP-Elites, is a quality-diversity algorithm that maintains a collection of optimal solutions that exhibit distinct characteristics based on a specified feature descriptor. The inclusion of a descriptor is an essential component of the MAP-Elites algorithm as it enables the categorization of individual solutions and facilitates the storage of diverse solutions inside the archive. It is imperative to recognize that the terminology employed to describe a solution need to be detached from the solution's level of effectiveness. Indeed, if the chosen attributes exhibit a strong correlation with the fitness measure, the illumination pattern would not hold substantial relevance. The researchers conducted experiments using two tasks, Cart Pole and Mountain Car, sourced from the Open AI Gym collection. They proceeded to compare the outcomes achieved by the MAP-Elites approach with those produced through the use of Genetic Algorithms (GE). The subsequent analysis focused on the outcomes of the two evolutionary algorithms (EAs) in relation to their performance and capacity to provide "illumination." This evaluation was conducted within the context of a feature space that was delineated by model complexity and behavioral variability. The researchers noted that in both challenges, MAP-Elites was able to identify solutions that effectively solved the task while also efficiently illuminating the feature space, in comparison to GE. Furthermore, it is worth noting that both evolutionary algorithms (EAs) generated models with low complexity, hence enhancing their interpretability. However, in the context of the Mountain Car challenge, it was seen that the ME algorithm identified that one specific action was not essential for successfully accomplishing the objective.

TABLE III. Illustrates the chronology of EAs for DT induction

Ref.	Year	Evolutionary Techniques	Dataset	Work Summary
[14]	2011	Genetic programming Generally evolve DT t-statistics mutual information	Kent ridge biomedical dataset	Comparison between GP and Generally evolve DT. Shows that GP has the opportunity to rank the attributes. This feature was utilized to cross-check the accuracy of forecasting.The performance of the Gaussian Process (GP) model, combined with feature selection, demonstrated exceptional results, achieving an average accuracy ranging from a minimum of 98.33% (when employing t-statistic for feature selection) to a maximum of 100% (when utilizing mutual information for feature selection). And the GATree algorithm, when combined with feature selection, demonstrated exceptional performance, with average accuracies ranging from a minimum of 85.00% (when employing t-statistic for feature selection) to a maximum of 88.33% (when utilizing mutual information for feature selection).
[15]	2012	EDTs	UCI medical datasets	Using (EDTs) for medical problem-solving gives more interpretability for individual decisions than black-box complex decisions in neural networks classifiers.
[11]	2013	Evolutionary approaches	UCI repository	This analysis aims to evaluate the evolutionary design process of decision trees (DTs) with respect to their key properties, such as accuracy, tree size, and utilized attributes. The selection of a particular method necessitates a careful consideration of striking a balance between accuracy and complexity. Among the available options, they opted for the one that exhibits the most favorable balance between complexity factors such as the number of nodes, depth, and the utilization of various qualities, and accuracy. The decision tree model consists of 19 decision nodes, also known as leaf nodes. When evaluated on a test set containing unseen cases, the model achieves an accuracy of 85.63%.
[16]	2013	HEAD-DT	Microarray gene expression data from 35 real-world datasets	Evolves hyper-heuristic top-down decision-tree induction design components in light of recent advances in autonomous machine learning algorithm design. It automatically creates categorization data set specific to decision-tree induction approaches (or application domains).The performance of HEAD-DT was evaluated using two distinct predictive performance measures, namely accuracy and F-measure. The results of the experimental analysis indicate that the decision-tree induction algorithms developed by HEAD-DT demonstrated superior performance compared to the three baseline approaches (CART, C4.5, and REPTree) in 9 out of 10 experiments. In relation to statistical significance, the predictive performance of HEAD-DT was shown to be considerably greater than that of all three baseline methods in two trials. Additionally, it was observed that HEAD-DT never exhibited a significantly poorer predictive accuracy compared to any of the three baseline methods in any experiment.

[17]	2014	MTDT	Datasets of actual gene expression	The classification process for gene expression data is discussed. A unique splitting criterion was developed to improve classification accuracy and reduce DT underfit on specific types of data.All classifiers, including the MTDT algorithm, were utilized using the default parameter values on all datasets. The findings indicate that employing the Minimum Total Deviation Tree (MTDT) algorithm with a total of seven tests in a single node resulted in the highest average accuracy of 82.20% across all categorization tasks. In a general sense, it is evident that more intricate techniques such as Random Forest (RF), AdaBoost (ADA), and Bagging (BG) exhibit superior performance compared to conventional non-ensemble algorithms that produce simpler solutions. The MTDT approach successfully attained a high level of accuracy, while ensuring the preservation of complete classification rules by the utilization of uni-variate tests in multi-test splits. Based on the findings of the Friedman test, a statistically significant distinction (p-value of 0.0215) has been observed among the evaluated classifiers. According to the findings of Dunn's Multiple Comparison Test, there exists a statistically significant distinction in quality when comparing the MTDT (with a sample size of 7) to both the BF and j48 trees. The AD classifier was not included in the statistical analysis due to its inability to be applied to a dataset with multiple classes.
[18]	2014	EVO-Tree	Public UCI data sets	An evolutionary method with many objectives is suggested to develop binary DTs for categorization. The overall high accuracy of the EVO-Tree algorithm demonstrates its strong capability in handling datasets characterized by a limited number of features in comparison to a large number of characteristics. The EVO-Tree training technique has demonstrated superior efficacy in extracting valuable information and generating decision rules from challenging datasets, in comparison to traditional decision tree algorithms. It is believed that the usefulness of employing evolutionary algorithms (EA) lies in their ability to optimize tree structures, hence preventing the occurrence of local minima. This is particularly advantageous as classical decision tree induction procedures are prone to such limitations.
[26]	2019	Evolutionary Multi-Test Tree (EMTTree)	35 freely accessible datasets for gene expression, The information relates to various cancer subtypes or kinds.	(EMTTree) may help molecular biology uncover biomarkers, gene-gene interactions, and high-quality predictions, which significantly improved the DT's accuracy and stability and kept predicted structural patterns understandable by not increasing tree complexity. EMTTree has demonstrated superior performance in classification accuracy over a range of over 20 datasets. On average, it has achieved a classification accuracy that surpasses state-of-the-art solutions by more than 6% and outperforms the latest algorithm (HEAD-DT) specifically created for gene expression data by 4%.

[19]	2017	GPU-based parallelization of DT evolutionary induction	The experimental validation, carried out using synthetic datasets of various sizes	The objective is to enhance the efficiency of the GDT system and enable effective induction of DTs through evolutionary processes. To do this, it is proposed to implement parallelization techniques that leverage the computing capabilities of modern GPUs. This approach would allow the GPUs to handle computationally intensive tasks such as fitness calculation, so freeing up the CPU to perform other responsibilities.The researchers in this study exclusively emphasized the temporal performance of the GDT system, therefore omitting the inclusion of results pertaining to classification accuracy. Nevertheless, the GDT system successfully generated trees with optimal architectures and achieved nearly flawless accuracies (99-100%) for all simulated datasets that were examined. The experimental results obtained from both simulated and real-life datasets demonstrate that our proposed approach exhibits high efficiency and scalability. Specifically, our solution is capable of effectively handling large-scale data, as seen by the successful execution of tree induction on a dataset comprising 20,000,000 instances within a time frame of 20 minutes. In contrast, the sequential technique would require almost 8 days to do the same task.
[27]	2020	Building an ensemble of DTs	Several well-known datasets from the UCI collection	Encoding a DT as a real-valued- homogeneous vector since feature indices are also real numbers, and decoding by finding the minimum. This method supports differential evolution, evolution strategies, and other optimization methods. This method demonstrates superior quality compared to traditional approaches such as CART, random forest, and AdaBoost. However, it is worth noting that in order to attain these elevated performance levels, the proposed algorithms require a greater amount of computational time in comparison to the classical algorithms.
[28]	2021	DIMPLED	CNC and pasteurizer	Increases discretization performance while maintaining interpretability with a single DT. DIMPLED allows a tree to have many meaningful splits when used with k-means clustering for global discretization. The findings indicate that while CART and DIMPLED have comparable interpretability capabilities, DIMPLED demonstrated superior performance in both training and test accuracies compared to CART. Moreover, the DIMPLED framework has the capability to discover the underlying factors that contribute to a particular issue, as well as the relationships and connections between these factors. The DIMPLED method demonstrates superior interpretability in comparison to other tree-based algorithms. Furthermore, it can be observed that DIMPLED demonstrates the ability to enhance the decision tree through the employment of an evolutionary process for global discretization. This is evidenced by the tree's notable superiority in terms of performance when compared to the C4.5 and CART algorithms. Moreover, the model and its discretized properties possess a high degree of transparency and interpretability, hence enhancing the comprehensibility of manufacturing systems.

[29]	2020	NLDT (nonlinear DT)	There are: • 2 real-world classification tasks • 3 real-world optimization challenges • 4 custom test problems • 1 multi-class problem • 8 multi-objective problems totaling between 300 and 500 features.	Each non-terminal conditional node represents a nonlinear characteristic function for a split rule. Each split rule separates data into two sections that don't overlap. The tree grows hierarchically until one of the termination requirements is met. The effectiveness and practicality of the suggested method have been extensively shown by the achieved outcomes on well-established classification benchmark problems and a real-world single objective problem.
[30]	2023	Baldwinian evolutionary approach	Open AI Gym environments: • CartPole-v1 • MountainCar-v0 • LunarLander-v2	DT-based interpretable reinforcement learning. Two-level optimization that combines EAs and Q-learning. The authors put forth a two-tier optimization approach for the purpose of inducing decision trees in reinforcement learning environments. The trees acquired exhibit similar (or, superior) efficacy compared to cutting-edge methodologies, while possessing significantly greater interpretability.
[31]	2023	MAP-Elites algorithm	Open AI Gym library, • Cart Pole • Mountain Car	The application of an evolutionary algorithm (EA) in conjunction with interpretable Reinforcement Learning (RL) is employed to iteratively refine Decision Trees (DTs) by leveraging the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) framework. Evolutionary algorithms (EA) are responsible for the evolution of the decision tree (DT) structure, whereas reinforcement learning (RL) focuses on optimizing the actions performed by the terminal nodes of the tree. The MAP-Elites algorithm is a quality-diversity (QD) approach that preserves a repository of optimal solutions, utilizing a specified feature descriptor. The findings of the study indicate that the MAP-Elites algorithm shown superior efficiency in solving the tasks as compared to the Genetic Algorithm (GE). Additionally, both evolutionary algorithms (EAs) yielded models that exhibited low complexity and high interpretability. Nevertheless, it was found that in the context of the Mountain Car task, there existed an action that was deemed superfluous in achieving the objective.

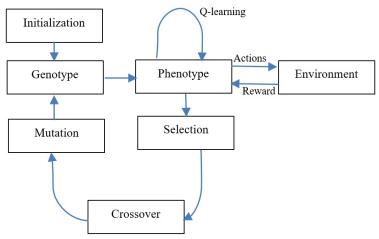

Figure 11. A diagram showing the suggested algorithm's internal structure. [30].

7. CONCLUSIONS

Decision trees are a popular classifier representation. In recent years, greedy divide-and-conquer algorithms for decision tree induction have grown less popular. Decision tree evolution has been studied extensively. This paper reviews the literature on evolutionary algorithms and decision trees. We also defined key steps of an evolutionary algorithm for decision tree induction and component evolution, which might help interested readers create their own EAs using a wide list of design options and techniques. The paper

addresses important factors for evaluating the proposed approaches, including processing time, accuracy, and required space. This comprehensive evaluation enables readers to understand the strengths and weaknesses of different methods and make informed decisions when applying them in practical scenarios.

References

[1] Z. Michalewicz and Z. Michalewicz, *GAs: Why Do They Work?* Springer, 1996.

[2] N. Tohidi, C. Dadkhah, and R. B. Rustamov, "Optimizing persian multi-objective question answering system," *International Journal on Technical and Physical Problems of Engineering (IJTPE)*, vol. 13, no. 46, pp. 62–69, 2021.

[3] R. Samsami, "Comparison between genetic algorithm particle swarm optimization and ant colony optimization techniques for nox emission forecasting in iran," *International Journal on "Technical and Physical Problems of Engineering"(IJTPE), International Organization of IOTPE*, pp. 80–85, 2013.

[4] H.-P. Schwefel, *Numerical optimization of computer models*. John Wiley & Sons, Inc., 1981.

[5] J. R. Koza, "G6. 1 classifying protein segments as transmembrane domains using genetic programming and architecture-altering operations."

[6] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial intelligence through simulated evolution. john willey & sons," *Inc., New York*, 1966.

[7] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evolutionary Computation*, vol. 4, no. 4, pp. 361–394, 1996.

[8] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 3, no. 2, pp. 124–141, 1999.

[9] M. Kamber and J. Pei, *Data mining*. Morgan kaufmann, 2006.

[10] A. A. Freitas, *Data mining and knowledge discovery with evolutionary algorithms*. Springer Science & Business Media, 2002.

[11] V. Podgorelec, M. Šprogar, and S. Pohorec, "Evolutionary design of decision trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 63–82, 2013.

[12] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 2, pp. 121–144, 2009.

[13] R. C. Barros, M. P. Basgalupp, A. C. De Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 3, pp. 291–312, 2011.

[14] A. Kulkarni, B. N. Kumar, V. Ravi, and U. S. Murthy, "Colon cancer prediction with genetics profiles using evolutionary techniques," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2752–2757, 2011.

[15] P. Kokol, S. Pohorec, G. Štiglic, and V. Podgorelec, "Evolutionary design of decision trees for medical application," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 3, pp. 237–254, 2012.

[16] R. C. Barros, M. P. Basgalupp, A. A. Freitas, and A. C. De Carvalho, "Evolutionary design of decision-tree algorithms tailored to microarray gene expression data sets," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 873–892, 2013.

[17] M. Czajkowski, M. Grześ, and M. Kretowski, "Multi-test decision tree and its application to microarray data classification," *Artificial intelligence in medicine*, vol. 61, no. 1, pp. 35–44, 2014.

[18] D. Jankowski and K. Jackowski, "Evolutionary algorithm for decision tree induction," in *Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings 14*. Springer, 2014, pp. 23–32.

[19] K. Jurczuk, M. Czajkowski, and M. Kretowski, "Evolutionary induction of a decision tree for large-scale data: a gpu-based approach," *Soft Computing*, vol. 21, pp. 7363–7379, 2017.

[20] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE transactions on evolutionary computation*, vol. 6, no. 5, pp. 443–462, 2002.

[21] M. Czajkowski, M. Czerwonka, and M. Kretowski, "Cost-sensitive global model trees applied to loan charge-off forecasting," *Decision Support Systems*, vol. 74, pp. 57–66, 2015.

[22] M. Grześ and M. Kretowski, "Decision tree approach to microarray data analysis," *Biocybernetics and Biomedical Engineering*, vol. 27, no. 3, pp. 29–42, 2007.

[23] M. Kretowski and M. Grzes, "Evolutionary induction of mixed decision trees," in *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2008, pp. 3509–3523.

[24] M. Czajkowski and M. Kretowski, "Evolutionary induction of global model trees with specialized operators and memetic extensions," *Information Sciences*, vol. 288, pp. 153–173, 2014.

[25] M. Czajkowski, K. Jurczuk, and M. Kretowski, "A parallel approach for evolutionary induced decision trees. mpi+ openmp implementation," in *Artificial Intelligence and Soft Computing: 14th International Conference, ICAISC 2015, Zakopane, Poland, June 14-18, 2015, Proceedings, Part I 14*. Springer, 2015, pp. 340–349.

[26] M. Czajkowski and M. Kretowski, "Decision tree underfitting in mining of gene expression data. an evolutionary multi-test tree approach," *Expert Systems with Applications*, vol. 137, pp. 392–404, 2019.

[27] E. Dolotov and N. Zolotykh, "Evolutionary algorithms for constructing an ensemble of decision trees," in *Analysis of Images, Social Networks and Texts: 8th International Conference, AIST 2019, Kazan, Russia, July 17–19, 2019, Revised Selected Papers 8*. Springer, 2020, pp. 9–15.

[28] S. Jun, "Evolutionary algorithm for improving decision tree with global discretization in manufacturing," *Sensors*, vol. 21, no. 8, p. 2849, 2021.

[29] Y. Dhebar and K. Deb, "Interpretable rule discovery through bilevel optimization of split-rules of nonlinear decision trees for classifica-

tion problems," *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5573–5584, 2020.

[30] L. L. Custode and G. Iacca, "Evolutionary learning of interpretable decision trees," *IEEE Access*, vol. 11, pp. 6169–6184, 2023.

[31] A. Ferigo, L. L. Custode, and G. Iacca, "Quality diversity evolutionary learning of decision trees," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 425–432.

Maryam Hussein Bahar : Birthday: 24/10/1994 Birth Place: Najaf, Iraq Bachelor: Computer Science, Department of Computer, Faculty of Education for Women, University of Kufa, Najaf, Iraq, 2012 Master: Computer Science, Software Department, Collage of Information Technology, University of Babylon, Babil, Iraq, 2016 The Last Scientific Position: Assist. Lecturer Assist./Lecturer (Doctorate) Student, Software Department, Collage of Information Technology, University of Babylon, Babil, Iraq, Since 2020 Research Interests: Machin learning, Image processing, Data Mining Scientific Publications: 2 Papers .

Hadeel Noori Saad : Birthday: 3/11/1973 Birth Place: Najaf/Iraq Bachelor: Computer Science, Computer Science, Faculty of Science, University of Babylon, Babylon, Iraq, 1995 Master: Computer Science, Computer Science, Faculty of Science, University of Babylon, Babylon, Iraq, 2001 Doctorate: Computer Networks Iraqi Commission for Computers and Informatics Institute for Postgraduate Studies, Baghdad, Iraq, 2006 The Last Scientific Position: Prof., Department of Computer Science, Faculty of Education for Women, University of Kufa, Najaf, Iraq, Since 2001 Research Interests: Computer Networks, Machine Learning, Data Mining Scientific Publications: 7 Papers .