Quality Diversity Genetic Programming for Learning Decision Tree Ensembles

Stephen Boisvert[1] and John W. Sheppard[2]

[1] Johns Hopkins University, sboisve2@jhu.edu
[2] Montana State University, john.sheppard@montana.edu

Abstract. Quality Diversity (QD) algorithms are a class of population-based evolutionary algorithms designed to generate sets of solutions that are both fit and diverse. In this paper, we describe a strategy for applying QD concepts to the generation of decision tree ensembles by optimizing collections of trees for both individually accurate and collectively diverse predictive behavior. We compare three variants of this QD strategy with two existing ensemble generation strategies over several classification data sets. We then briefly highlight the effect of the evolutionary algorithm at the core of the strategy. The examined algorithms generate ensembles with distinct predictive behaviors as measured by classification accuracy and intrinsic diversity. The plotted behaviors hint at highly data-dependent relationships between these metrics. QD-based strategies are suggested as a means to optimize classifier ensembles along this performance curve along with other suggestions for future work.

Keywords: Genetic Programming · Decision Tree Ensemble · Quality Diversity.

1 Introduction

Decision tree induction is a supervised learning strategy in which labels for data points are predicted through a series of decisions modeled as nodes of a tree. That is, given a data point m consisting of a set of feature and value pairs, a tree τ attempts to predict $label(m)$, which is a value associated with m but unknown to the model. In order to perform the prediction $p(\tau, m)$, the model recursively applies tests to m, beginning at the root of the tree. Each test results in a decision that maps one or more input feature values to a choice of adjacent nodes. The corresponding branch is then traversed, and the process repeats until a leaf node is reached. The value of the ending leaf node then outputs $p(\tau, m)$.

At the cost of some simplicity, decision tree ensembles (or forests) can be formed in which the output labels of multiple decision trees are considered collectively in order to make a final prediction. That is, the ensemble prediction $p(T, m)$ combines $p(\tau, m), \forall \tau \in T$ to predict the value of $label(m)$. Using an ensemble necessarily implies some amount of dissimilarity of individual trees within the forest. Otherwise, the output of the complete ensemble would be equal to the output of any subset of that ensemble. Thus, there must be diversity within the ensemble population in order for the ensemble strategy to be effective.

Quality-Diversity (QD) algorithms are a class of population-based optimization algorithms that aim to output a collection of quantifiably high performing and diverse solutions [20]. While diversity is a common theme in evolutionary computation, the explicit diversity considered by QD algorithms is distinct from strategies that promote population diversity implicitly by increasing the use of stochastic methods. QD algorithms measure diversity by comparing the *behavioral characterizations* of individual solutions. Groups of similar individuals, called niches, are then identified, and a high performing representative for each behavioral niche is returned in the solution.

The concept of diversity and the application of decision tree formation are common subjects in the field of genetic programming [8] [4]. In this paper, we examine the qualities of decision tree ensembles formed from the output of a QD optimization algorithm. Specifically, we formulate a QD algorithm such that individuals in the population are complete decision trees with fitness and diversity calculated from prediction behavior over the training data. We then compare the accuracy and diversity of the resulting QD ensembles to the corresponding metrics of ensembles formed by alternative means under the hypothesis that the QD-generated ensembles will outperform the other studied algorithms in terms of both accuracy and diversity.

The remainder of the paper is organized as follows. In the following section, we review concepts in the areas of decision trees, ensemble classifiers, and QD algorithms. In Section 3, we present the specific algorithms used in this paper in detail. In Section 4 we describe the experimental configurations, analyzing the results in Section 5. Finally, in Section 6 we summarize our conclusions and then suggest future work.

2 Background

2.1 Decision Tree Diversity

Decision tree ensembles require some amount of diversity in order to behave differently than a single tree. Breiman shows a particular relationship between the misprediction probability or *generalization error* (PE^*) of an ensemble, the strength (s) of the ensemble, and prediction correlation ($\bar{\rho}$) between individuals within that ensemble [6]. Specifically, while introducing the well-known "Random Forest" algorithm, it is shown that:

$$PE^* \leq \bar{\rho}(1 - s^2)/s^2,$$

where $-1 \leq s \leq 1$ is defined as the expectation of the *margin function*, which describes the confidence of the ensemble in predicting correct classification labels, and $\bar{\rho}$ is described as the mean value of the correlation in *raw margin function* between individual classifiers in the ensemble. The full proof will not be discussed here, though Breiman finally suggests a ratio of $\bar{\rho}/s^2$ such that "the smaller it is, the better." That this, both higher strength and lower correlation of individuals will reduce the theoretical bound on this error. In this way, concepts of both fitness and diversity are shown to be reasonable objectives for selecting members of a classifier ensemble.

Given this formulation, a natural way to describe the diversity of a population of decision trees is to describe the correlation of their outputs for a given input set. Several strategies have been proposed for this purpose, including the averaged Q statistic [15], a Kappa statistic [11], classification overlap [7], ensemble ambiguity [14], and the percentage correct diversity measure (PCDM) [2]. While the latest is perhaps unique in being proposed specifically for the purpose of thinning decision tree forests, all of these strategies attempt to measure forest diversity by comparing the outputs of individual trees.

The value used to determine whether the predictions of given trees are equal is a key distinguishing characteristic of the listed metrics. Some, such as the Kappa statistic, compare the direct outputs of the trees. That is, they compare $p_{raw}(\tau, m)$ values, each of which is a predicted output label for a data point m using a decision tree τ. Others first convert the tree predictions to binary values, indicating whether each prediction was correct or incorrect. That is, they compare $p_{correct}(\tau, m)$ where

$$p_{correct}(\tau, m) = \begin{Bmatrix} 1, & p_{raw}(\tau, m) = label(m) \\ 0, & p_{raw}(\tau, m) \neq label(m) \end{Bmatrix} \tag{1}$$

In this paper, we use p_{raw} for diversity calculations using a simple count of the data points in a sample that result in different outputs between two trees. That is, the diversity metric $\mathit{\Delta}_{raw}$ between two trees τ_1, τ_2 over the points m in a data sample M is calculated as the sum of:

$$\delta_{raw}(\tau_1, \tau_2, m) = \begin{Bmatrix} 0, & p_{raw}(\tau_1, m) = p_{raw}(\tau_2, m) \\ 1, & p_{raw}(\tau_1, m) \neq p_{raw}(\tau_2, m) \end{Bmatrix}$$

such that:

$$\mathit{\Delta}_{raw}(\tau_1, \tau_2, M) = \sum_{m \in M} \delta_{raw}(\tau_1, \tau_2, m) \tag{2}$$

The diversity of a decision tree ensemble T is then computed as the average pairwise $\mathit{\Delta}_{raw}$ between individuals in the ensemble. That is:

$$\bar{\mathit{\Delta}}_{raw}(T, M) = \frac{\sum_{(\tau_1, \tau_2) \in T \times T, (\tau_1 \neq \tau_2)} \mathit{\Delta}_{raw}(\tau_1, \tau_2, M)}{size(T) \times (size(T) - 1)} \tag{3}$$

This formulation provides a relatively simple way to interpret a reported diversity measure as a distance metric but does not attempt to account for differences in population size or other properties of M such as label representation. Therefore, it is important to not compare the reported $\bar{\mathit{\Delta}}$ measures between data sets or experimental configurations.

2.2 Decision Tree Induction

Common strategies for forming decision trees implement a greedy heuristic approach for determining the test to apply at each node. The tree is formed root-to-leaf in a way that recursively maximizes a quantifiable *split criterion* over the training data. This split criterion measures the predictive power of that node in isolation. Examples of the split criterion include information gain, Gini impurity, and prediction accuracy, though studies suggest that the difference between these criteria often have minimal effect on the resulting trees [21].

The chosen decision is then applied to the training data, and the children at the branches of the tree are trained using their respective data subsets. This continues until some stopping condition is met, typically a level of accuracy or a data sample size. Alternatively post-pruning can be applied to prevent overfitting. Note that, for a given input M and a list of possible decisions, this algorithm is deterministic and will produce identical trees. Thus, some additional stochastic elements are needed when forming diverse decision tree ensembles.

One option for diversity injection is through the training data input to the greedy tree formation algorithm. One popular and effective method, called *bootstrap aggregating* or *bagging* [5], forms separate data sets for training different trees by sampling data points at random (with replacement) from the original training set [3]. That is, to form a training set M_{τ_1} used to train a tree τ_1, random data points are selected from the original M until $|M_{\tau_1}| = |M|$.

A related strategy called boosting provides weights to the selection options such that inaccurately classified points are more likely to be added to new training sets. A comparison suggests that bagging and boosting can each be more effective than the other, depending on the amount of noise in the data set [11].

Alternatively, the structures of the trees can be built in a stochastic manner. One option is to generate random trees without using information from the training data. That is, a tree is generated where each test is selected as a random feature and test value. Labels for leaf nodes are determined later by feeding in the training data to the randomized decisions and examining the distributions at the leaves [13]. This approach may result in unreachable nodes, however, which can either be removed or saved for online learning.

Also, unreachable nodes can be prevented by partially using the training data when forming decisions in the tree. One option is to select randomly from the best n discovered decisions according to the split heuristic [11]. Another option is to select a random set of attributes with a decision threshold chosen as the half-way point between two randomly selected training points [17]. A particular configuration of such trees are called a *Max-diverse Ensemble.*

2.3 Evolutionary Decision Tree Induction

Evolutionary computation has also been applied extensively to the stochastic formation of decision trees [4]. In particular, genetic programming approaches fit naturally to the tree structure of the decision tree classifiers. Also, more traditional evolutionary algorithms have been applied to relatively rigid tree structures. While discussing the full range and taxonomy of population-based implementations is outside the scope of this paper, we present the following information as relevant to the evolutionary component of our QD algorithm implementation.

When restricting the tree structure to binary trees of a maximum depth, each tree-encoding chromosome can be described with a fixed number of genes, where each gene represents a single node in the tree [1]. That is, the number of genes G to represent an individual is given by:

$$G = 2^{depth+1} - 1$$

where each gene can represent either a leaf-node or a sub-tree.

In our experiments, we share the gene encoding used by [22] with $node = \langle t, label, P, L, R, C, size \rangle$, where t is an identifying node number, P is a pointer to the parent node, L and R are pointers to the left and right children, $size$ indicates the height of the contained sub-tree, and C is a set of registers. If the node is an internal node, $C[0]$ represents the feature index used in the branching decision at that node while $C[1]$ represents the threshold of the decision. That is, the decision nodes use tests in the form of

$$feature_{C[0]} < C[1]$$

for real-valued features, or

$$feature_{C[0]} = C[1]$$

for discrete-valued features. If the node is a leaf node, C will contain an array of values representing the weighted labels of training data that reach the node. This information is then used to predict labels of testing data.

Using such a representation, the crossover operator can be implemented such that one random gene (and corresponding sub-tree) from two parents are swapped as used by the "standard GP" [22]. Note that in our implementation, we limit the selected genes to ensure they are of the same height so as to maintain the complete structure of the trees.

The mutation operator can be implemented by modifying the C[0] and C[1] registers for decision nodes according to some fixed rate. In this paper, the mutation operator chooses a random element for C[0]. For the test targets, mutation either adds a random value from the normal distribution to C[1] or selects uniformly from the set of discrete values. With these encodings and operators defined, we can use our tree representation in any standard population-based genetic framework.

2.4 QD Algorithms

Quality Diversity algorithms aim to produce populations containing locally-optimal representatives of regions in a *behavioral characterization* (BC) space.

The BC space is a problem-specific description of possible solutions that provides a mechanism for calculating the similarity between solutions. For example, in [10], a set of BCs represent the final position of the end of a robotic arm after applying a series of angular rotations to joints in the arm. Thus, it is expected that many possible solutions may result in similar or perhaps identical BCs.

A foundational example of a QD algorithm is the *Novelty Search with Local Competition* (NSLC) algorithm, which uses a "Pareto-based multi-objective" strategy to optimize explicitly for both novelty and quality during search [16]. This algorithm was applied to evolve a set of virtual creatures in a simulated robotic environment. The creatures were evolved to have diverse properties with local competition, ensuring that each returned representative showed relative success in achieving the goal of movement.

Another exemplary QD algorithm is *Multi-dimensional Archive of Phenotypic Elites* (MAP-Elites), which seeks to divide the behavior space into a grid, then "illuminate" the quality of the regions through high performing, representative individuals [19]. It is also noted that this strategy can be used as an optimizer by returning the overall highest performing individual while maintaining the benefits of diversity during the search process.

These algorithms share a common goal of producing populations containing locally-optimal representatives of behavioral niches. However, their structures have distinct characteristics. For example, MAP-Elites divides the behavioral feature space into a *grid* with a representative for each space, while NSLC allows for a more flexible *archive* where acceptance into the collection is determined by comparison to nearest neighbors. Work to unify these examples into a common framework has abstracted these differences into types of *container operators* [10].

Additionally, the selection of individuals to insert into future generations has been abstracted as a *selection operator* [10]. Examples of selection operators include *uniform random selection* and *score-proportionate selection.* A typical example of the latter is to establish a score for each individual proportional to its fitness or diversity, where a higher score increases the chance of being selected. A suggested type of score called the *curiosity score* provides higher values to individuals that produce new offspring that are added successfully to the container [10].

In this paper, we use the predictive behavior over the training set as the behavioral characterization of each tree. This metric is related to the label purity at leaf nodes. Distances to nearest neighbors in the behavioral space are then calculated as the defined Δ_{raw} metric from Equation 2. Trees are added to an archive container with a minimum distance requirement between all individuals in the population. Newly generated trees can replace existing members of the container if they are sufficiently high-performing.

2.5 Ensemble Prediction

The final prediction of an ensemble is a combination of the predictions of the individual classifiers within that ensemble. The strategy chosen for this combination is critical to the effectiveness of the ensemble. Perhaps the simplest strategy is to apply a voting technique where the most frequent output among individuals is selected as the overall output [18]. These votes can also be weighted or ranked according to confidence of the prediction for one or more outputs [23]. More complex strategies, such as stacked generalization or meta-learning, apply a learning process to the combination [9].

For this paper, we use only the *sum rule* strategy for calculating ensemble output. In this strategy, each decision tree provides a confidence value for each possible output. For each decision, the confidence value is calculated as the percentage of the training label purity of the deciding leaf node. All confidence values are added over all trees in the ensemble, and the choice with the highest sum is chosen as the ensemble output [23]. However, we suspect that more intentionally complementary combination strategies could increase the effectiveness of any ensemble generation algorithm.

3 Implementation

3.1 Structure Restrictions

Our implemented algorithms all require the decisions within each internal tree node to be Boolean. That is, each test will either result in an evaluation of either *true* or *false*. If the test evaluates to *true*, the tree will return the prediction of its left subtree; otherwise, it will return the right subtree's prediction. In the case of real-valued features, each test will determine whether the value of a given feature in the input data point is below some threshold. For discrete-valued features, the test will determine whether the input feature value is equal to some test value.

3.2 Bagging

Our bagging implementation follows the description in Section 2.2 using a standard greedy algorithm. That is, to generate an ensemble $T_{bagging}$ of n binary decision trees $\{\tau_1, \tau_2, ..., \tau_n\}$, the algorithm first generates a set of n training sets $\{M_1, M_2, ..., M_n\}$, each of which contains elements selected randomly from the original training set M with replacement until $|M_i| = |M|$. Then:

$$\tau_i = GreedyHeuristic(M_i), \forall i \in \{1, 2, ..., n\}$$

The split criterion for each rule choice is calculated as the accuracy of predicting the labels of the elements in the M_{left} and M_{right} sub-groups as the majority label in each respective group. This essentially calculates the accuracy of treating each sub-group as a leaf, functioning as a measure of label purity. Recall from Section 2.2 that we would expect replacing accuracy with information gain or Gini impurity to have similar results. The greedy formulation terminates when the size of the input to $GreedyHeuristic$ is below a percentage of $|M_i|$.

3.3 Random Trees

The structures of the random trees examined in this paper are formed as complete, binary decision trees. To form each individual τ_{rnd} in an ensemble T_{rnd}, the algorithm picks a series of $2^{depth} - 1$ randomly generated training data features on which to base decisions for internal tree nodes. The threshold for each node is then determined root-to-leaf by calculating the mean value of the corresponding feature from two randomly chosen (with replacement) data points that traverse that node. For discrete-valued features, the corresponding feature value from a randomly chosen point is used for the equality condition. Up to 2^{depth} leaf nodes are labeled by feeding the training data M into the established trees and keeping a record of which $m \in M$ reach each node. Any decision nodes that are reachable by a number of points less than an established percentage of $|M|$ are converted into leaf nodes with their children nodes truncated. Note that such pruned trees will no longer be complete.

3.4 QD Trees

Our implementation of the QD algorithm for decision tree generation uses an *archive* container A where the pairwise distance between tree elements $\tau_i, \tau_j \in A$ is calculated as the diversity measure $\Delta_{raw}(\tau_i, \tau_j, M)$ from Section 2.1, Equation 2. Given a minimum distance threshold Δ_{min}, this container will allow the addition of a new element τ_{new} based on its comparison to its nearest two neighbors τ_{first} and τ_{second} where:

$$\tau_{first} = \arg\min\{\Delta_{raw}(\tau_{new}, \tau_i, M), \forall \tau_i \in A\}$$

and:

$$\tau_{second} = \arg\min\{\Delta_{raw}(\tau_{new}, \tau_i, M), \forall \tau_i \in A \setminus \{\tau_{first}\}\}$$

Given this, τ_{new} will be added to A without affecting other members if its nearest neighbor is farther away than the minimum threshold. If both of the two nearest neighbors of τ_{new} are closer than the minimum threshold, τ_{new} will not be added to the container. However, if the nearest neighbor is closer than the threshold while the second nearest neighbor is farther, τ_{new} may replace its nearest neighbor if its fitness is higher, where fitness is measured as accuracy in predicting the labels of the training data set. Note this has a similar effect as our greedy tree formation such that trees with higher label purity in leaf nodes will have higher fitness. This is shown more precisely in Algorithm 1.

The container is initialized by generating a fixed number of τ_{rnd} trees and adding them to the container in order, as long as the diversity conditions are met as described above. Note that the first two decision trees are always added successfully to the container. Next Δ_{min} is tuned to ensure that the container capacity $|A|$ does not increase above a given size during initialization. This helps to ensure that the nearest neighbor calculations remain in an achievable range of computational complexity during the algorithm's execution.

Algorithm 1 Conditional Archive Addition

```
1: procedure AddTreeToArchive($\tau_{new}$)
2:     $globals : Archive\ A,\ Data\ M,\ Constant\ \Delta_{min}$
3:     $\tau_{first} \leftarrow \tau_1$
4:     $\tau_{second} \leftarrow \tau_1$
5:     for $\tau_i$ in $A$ do
6:         if $\Delta_{raw}(\tau_{new}, \tau_i, M) < \Delta_{raw}(\tau_{new}, \tau_{first}, M)$ then
7:             $\tau_{second} \leftarrow \tau_{first}$
8:             $\tau_{first} \leftarrow \tau_i$
9:         else if $\Delta_{raw}(\tau_{new}, \tau_i, M) < \Delta_{raw}(\tau_{new}, \tau_{second}, M)$ then
10:            $\tau_{second} \leftarrow \tau_i$
11:    if $\Delta_{raw}(\tau_{new}, \tau_{first}, M) < \Delta_{min}$ then
12:        $A \leftarrow A + \tau_{new}$
13:    else if $\Delta_{raw}(\tau_{new}, \tau_{second}, M) < \Delta_{min}$ then
14:        if $fitness(\tau_{new}) > fitness(\tau_{first})$ then
15:            $A \leftarrow A + \tau_{new}$
16:            $A \leftarrow A \setminus \tau_{first}$
```

The algorithm then executes for a specified number of generations. At each generation, two parent trees are selected from the container (with replacement), where the selection probability for each tree is proportional to its current *curiosity score*. Recall that the curiosity score indicates how successful the offspring of that tree have been in recent history. Crossover is then applied to the parent trees as described in Section 2.3. Specifically, randomly selected sub-trees from each parent tree at the same height are swapped. Note that the crossover operation results in two children. Each child undergoes mutation where each decision node may have its tested feature modified and its threshold reinitialized according to some probability. Failing this, a sample from the normal distribution is added to real-valued thresholds while discrete-valued thresholds are modified to an alternate value according to a second sampling from mutation probability.

The mutated children are then tested to determine if they should be added to the container. If the addition is successful, the curiosity scores of the parents are increased by 1. Otherwise, the curiosity scores of the parents are decreased by 0.5. For instance, the successful addition of one child and the failed addition of the other will result in a net curiosity score gain of 0.5 for each parent. After the set number of generations, the container is returned.

Finally, a decision tree ensemble T_{QD} of a fixed size n must be generated from the resulting container. Here, we test three different strategies. The *accuracy* strategy (QD Accuracy) chooses the n decision trees in the container with the highest fitness. The *diversity* strategy (QD Diverse) chooses n random trees from the container. The *hybrid* strategy (QD Hybrid) chooses $\frac{n}{2}$ trees with the highest fitness, then chooses the remaining trees randomly. Once chosen, each tree is pruned such that decisions reachable by a number of training samples less than a specified percentage of $|M|$ are replaced by leaf nodes. This is done to ensure sufficient data at the leaves to make a statistically reasonable prediction.

Table 1. Data Set Attributes

Data Set	# Used Points	# Used Features	# Classes	Missing Data?
Balance	626	4	3	No
Breast Cancer	699	9	2	Yes
Dermatology	366	34	6	Yes
Flags	194	28	8	No
Glass	214	9	6	No
Heart	270	12	3	No
Hayes-Roth	132	4	3	No
Ionosphere	351	34	2	No
Iris	150	4	3	No

4 Experiments

For our experiments, we execute the previously described algorithms and measure their classification performance over nine data sets selected from the UCI Machine Learning Repository [12]. Specifically, we compare the results of the QD Hybrid, QD Accuracy, and QD Diverse algorithms against those of the Bagging and Random algorithm baselines.

The data sets were selected for manageable data sizes, few missing attribute values and varying numbers of features (Table 1). For all used data sets, any missing attribute values are handled by removing the affected data points from consideration. Uniquely identifying attributes such as IDs and names are also removed from the data. Note that for this experiment, we predict the "religion" field of the Flags data set based on the other attributes.

All presented diversity and accuracy results are determined using 10-fold cross validation. That is, each point graphed in Figure 1 is a result of this validation, and 10 such results are used in our statistical test. Note that the cross validation folds are stratified by class, ensuring proportionate representation of all classes within each division. Note that these diversity and accuracy metrics are the same metrics used in the construction of the ensembles with the QD variants. The use of alternative metrics for additional insights into the effectiveness of the classifiers is an opportunity for future work.

Each of the algorithms includes a pre-pruning hyperparameter that specifies the percentage of training points that much be reachable by every decision used in the tree. For these experiments, that setting is kept at a constant 1%. Additionally, each algorithm is configured to generate ensembles such that $|T| = 50$. The depth-limit of the randomly-generated and QD-generated trees is kept at 4.

Each QD algorithm variant is executed for 10,000 generations unless stated otherwise. Note that two children are created in each generation. For most experiments, Δ_{min} is chosen at initialization such that no more than 350 of 3,000 randomly initialized trees are added successfully to the container. If the container size goes above this threshold during initialization, the algorithm restarts with a higher Δ_{min}. Note that reported diversity measures use the pairwise averaged $\bar{\Delta}$

Table 2. Median algorithm accuracy and diversity. **Bold** indicates $p < 0.05$

Data Set	Bagging		Random		QD Acc		QD Div		QD Hybrid	
	Acc	Div	Acc	Div	Acc	Div	Acc	Div	Acc	Div
Balance	0.842	16.68	**0.887**	**20.71**	0.871	17.38	**0.888**	20.17	**0.883**	18.63
Breast Cancer	0.958	3.08	**0.965**	**9.54**	**0.972**	4.80	0.963	**9.63**	**0.968**	7.40
Dermatology	**0.966**	2.26	0.775	19.80	0.949	16.21	0.863	**20.45**	0.942	18.92
Flags	0.478	10.31	0.419	11.12	**0.599**	11.40	0.481	**11.90**	0.587	**11.88**
Glass	0.621	7.92	0.658	10.80	**0.690**	9.11	0.621	**12.27**	**0.689**	10.17
Heart	0.652	7.82	**0.668**	8.95	**0.680**	8.37	**0.674**	**9.34**	**0.677**	9.05
Hayes-Roth	**0.781**	4.86	0.669	6.95	0.749	6.28	0.674	**7.07**	0.736	6.69
Ionosphere	**0.923**	2.80	0.832	9.50	0.891	8.38	0.849	**10.25**	0.886	9.33
Iris	**0.963**	0.44	0.953	3.43	0.953	1.05	0.947	**4.06**	0.953	2.72

value (Equation 3), indicating the average number of data points in the testing set that have different predicted labels between two trees in the ensemble.

The mutation rate is set to 2%. Note that for discrete valued features, the sampling for the mutation may occur twice where a second sampling for modifying the threshold value will occur if the sampling for modifying the feature fails. For real valued features, the threshold is always modified by adding a sample from the normal distribution, $\mathcal{N}(0, 0.1)$.

5 Results

5.1 Algorithm Comparisons

Figure 1 plots the diversity and quality of the algorithms over the indicated data sets. Table 2 shows median accuracy and diversity metrics. Entries shown in bold are not significantly lower than the maximum value in the row according to the Wilcoxon Signed-Rank Test at the 95% confidence level.

5.2 Evolutionary Algorithm

Much of the niching behavior of the QD strategy can be attributed to the container, which enforces a level of diversity while allowing for quality improvements. Here, we investigate the change in algorithm behavior when the evolutionary components of the algorithm are replaced by the random tree generation strategy. Specifically, we examine the size of the container over time, the number of times an existing member of the container was replaced by a newly generated tree, and the average accuracy of the members of the container over the *training* data. As in other fitness metrics, the training data accuracy indicates a measure of the purity of the labels at the leaves of the decision trees. In this experiment, the number of generations is increased to 100,000, and Δ_{min} is increased by 25%. We show the results on the Glass data as a representative in Figure 2.

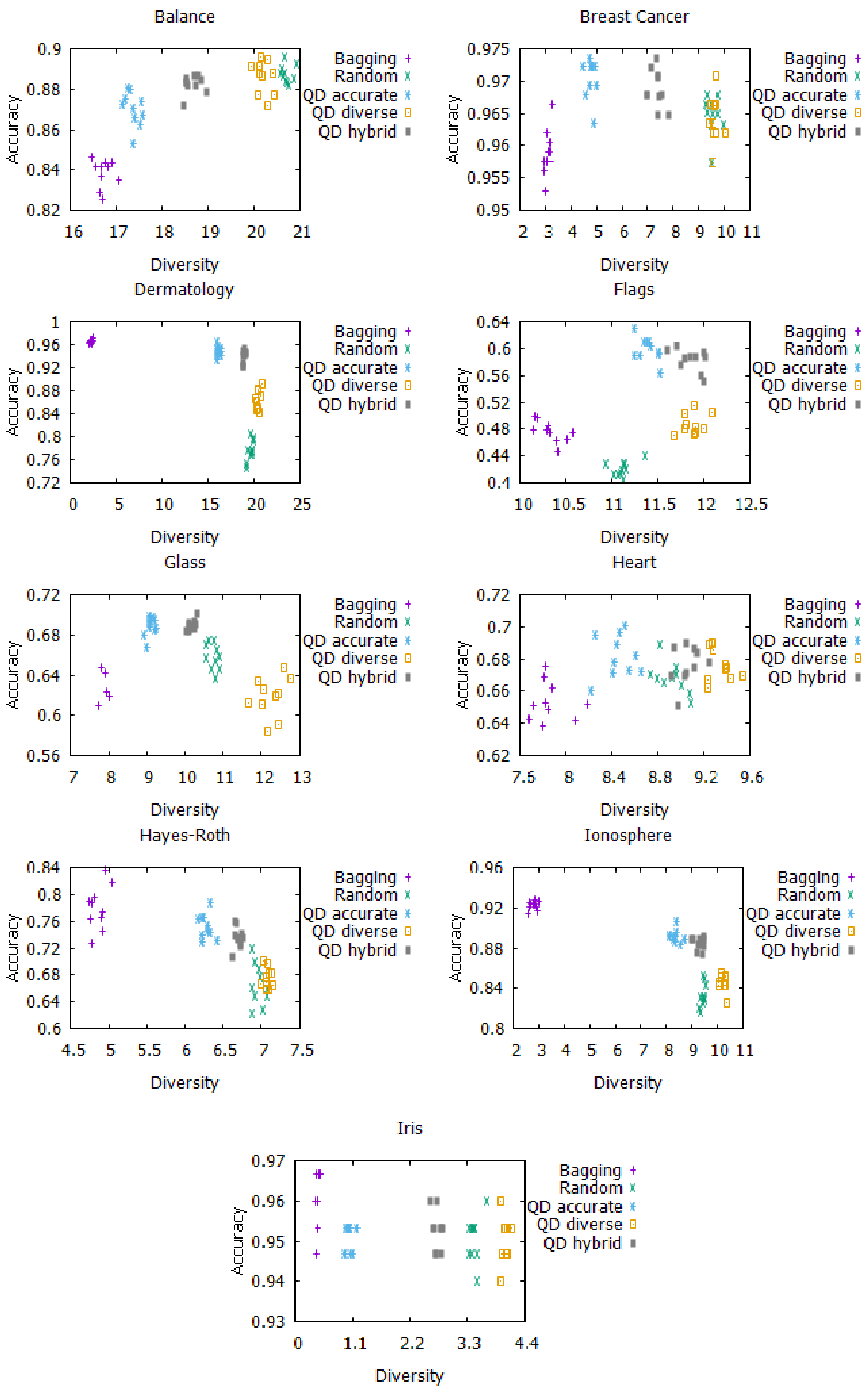

Fig. 1. Algorithm accuracy versus diversity for each of the nine data sets

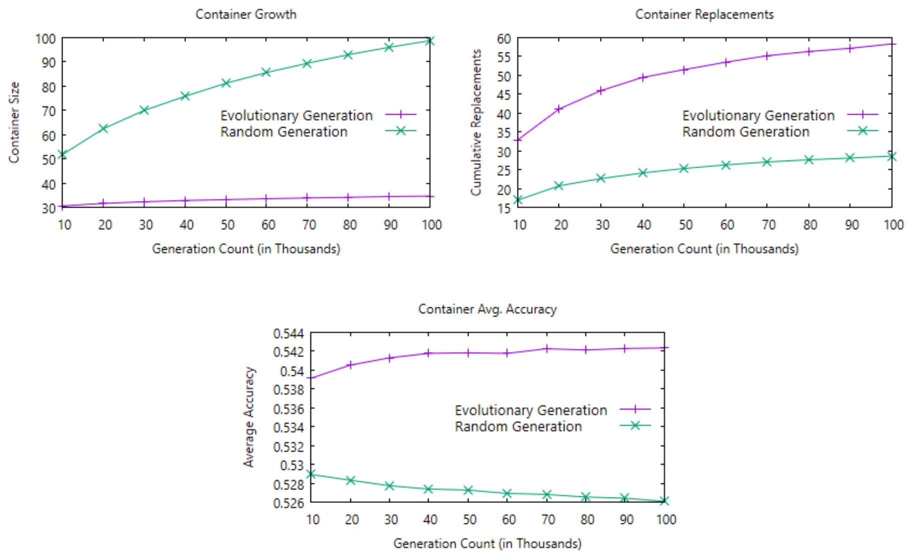

Fig. 2. Behavior of the QD algorithms vs. random generation on the Glass dataset

5.3 Binary Diversity

We examined the effect on performance of using the more common $p_{correct}$ (Equation 1) to calculate $\Delta_{correct}$ rather than Δ_{raw} (Equation 2) for the QD-hybrid algorithm, keeping all other settings constant. We found that the difference in accuracy was not statistically significant for the Glass, Ionosphere and Balance data sets tested. We consider these data sets to be representative of the accuracy/diversity relationship trends and label quantities of our data sets. Further experimentation with alternative diversity metrics, especially those less-directly tied to fitness, may prove interesting.

5.4 Ensemble Size

Finally, in Figure 3, we briefly examine the change in behavior of ensemble classifiers of varying sizes. As an example, we replot the diversity and accuracy of class predictions on the Glass data set from ensembles consisting of 50, 25, 10 and 2 individual trees. All other parameters are held constant for this experiment.

6 Analysis

The effectiveness of the QD strategy is shown to be highly data-dependent, as is the correlation between ensemble diversity and accuracy. Data sets such as Breast Cancer, Flags, Glass, and Heart demonstrate a mid-range of diversity

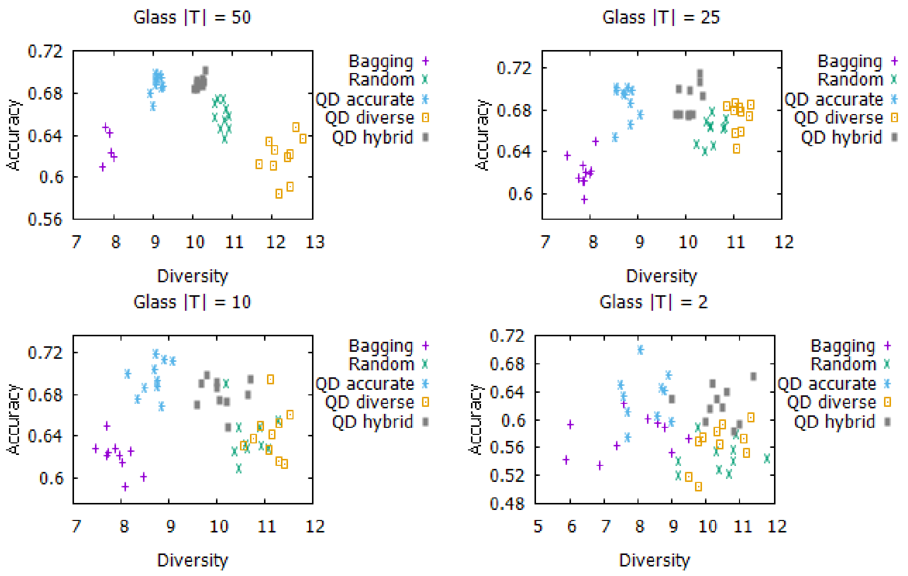

Fig. 3. Change in performance by ensemble size.

where accuracy tends to be higher. The Dermatology, Hayes-Roth, Ionosphere, and Iris data sets show a trend for less diverse ensembles to be more accurate. The Balance data set shows the reverse trend where more diverse ensembles tend to have higher accuracy. The QD-Diversity algorithm shows particularly promising results in creating diverse ensembles, shown here to be significantly less diverse than the top performer in only one data set. The QD-Hybrid variant demonstrates the possibility of tuning the behavior of the ensemble by more carefully selecting the individuals from the container output.

There are several cases where the local optimization of the QD algorithm is apparent from the plots. For example, in the Dermatology graph in Figure 1, we see the QD-Diverse ensemble outperform the Random ensemble, despite similar measures of diversity. Similar patterns are observed between QD-Hybrid and Random in the Ionosphere data set and between QD-Accurate and Random on the Flags data set. Indeed, Figure 2 suggests that a primary effect of combining the QD container with the evolutionary algorithm versus random generation is to optimize existing members of the container locally, thus increasing accuracy.

Figure 3 demonstrates a trend for behavioral variance of the algorithms to decrease as ensemble size increases while maintaining distinct average behaviors. This suggests a substantial difference in the properties of the ensembles based on varying tree generation and selection methods.

7 Conclusion

We have presented a new application of QD algorithms in generating decision tree ensembles containing trees that are both fit and behaviorally diverse. Our experiments show that the benefit of diversity in such an ensemble is highly dependent on the data set. For data sets with very defined rules, aiming for more diverse ensembles may be counterproductive. However, for other data sets, the QD generation methods appear to offer effective strategies for tuning the performance of resulting ensembles. Furthermore, we have demonstrated that while the container plays an important role in the QD formulation by enforcing a degree of diversity, the evolutionary algorithm component also plays a key role by promoting local optimization to a greater degree than simple random tree generation. The QD strategies offer a promising method for exploring a locally optimized relationship between ensemble accuracy and diversity.

The parameters used in our experiments have been highly controlled in order to demonstrate the differences between the algorithms in generating ensemble members. Performance may be improved further by allowing for more complex and deeper tree structures. Further study of these algorithms at their respective peaks of performance may give more insight into the practical utility of QD-generated ensembles. More flexible genetic programming implementations may also promote quicker ensemble generation, allowing more opportunities to optimize container contents. Similarly, we have restricted the ensemble integration mechanism to highlight the difference between ensemble compositions. Using more advanced combination techniques could improve the performance of these ensembles by allowing for diverse "specializations." Long computation times led to using relatively simple data sets for our experiments. Though we demonstrate varying relationships between accuracy and diversity even among these data sets, using more complicated data may reveal additional insights. Finally, we have focused on using decision tree ensembles, but the concepts used in this paper may apply in a straightforward way to other types of classifiers. For example, it would be possible to replace the tree models with neural networks given appropriate crossover and mutation operators.

References

1. Bandar, Z., Al-Attar, H., McLean, D.: Genetic algorithm based multiple decision tree induction. In: Proc. 6th International Conference on Neural Information Processing (ICONIP). vol. 2, pp. 429–434 (1999)
2. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: A new ensemble diversity measure applied to thinning ensembles. In: International Workshop on Multiple Classifier Systems. pp. 306–316. Springer (2003)
3. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: A comparison of decision tree ensemble creation techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(1), 173–180 (2006)
4. Barros, R.C., Basgalupp, M.P., De Carvalho, A.C., Freitas, A.A.: A survey of evolutionary algorithms for decision-tree induction. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**(3), 291–312 (2011)

5. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
6. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
7. Brodley, C.E.: Recursive automatic bias selection for classifier construction. Machine Learning **20**(1-2), 63–94 (1995)
8. Burke, E.K., Gustafson, S., Kendall, G.: Diversity in genetic programming: An analysis of measures and correlation with fitness. IEEE Transactions on Evolutionary Computation **8**(1), 47–62 (2004)
9. Chan, P.K., Stolfo, S.J.: On the accuracy of meta-learning for scalable data mining. Journal of Intelligent Information Systems **8**(1), 5–28 (1997)
10. Cully, A., Demiris, Y.: Quality and diversity optimization: A unifying modular framework. IEEE Transactions on Evolutionary Computation **22**(2), 245–259 (2017)
11. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40**(2), 139–157 (2000)
12. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
13. Fan, W., Wang, H., Yu, P.S., Ma, S.: Is random model better? on its accuracy and efficiency. In: Third International Conference on Data Mining. pp. 51–58. IEEE (2003)
14. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Advances in Neural Information Processing Systems. pp. 231–238 (1995)
15. Kuncheva, L.I., Whitaker, C.J., Shipp, C.A., Duin, R.P.: Is independence good for combining classifiers? In: International Conference on Pattern Recognition. vol. 2, pp. 168–171. IEEE (2000)
16. Lehman, J., Stanley, K.O.: Evolving a diversity of virtual creatures through novelty search and local competition. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO). pp. 211–218. ACM (2011)
17. Liu, F.T., Ting, K.M., Fan, W.: Maximizing tree diversity by building complete-random decision trees. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 605–610. Springer (2005)
18. Merz, C.J.: Dynamical selection of learning algorithms. In: Learning from Data, pp. 281–290. Springer (1996)
19. Mouret, J.B., Clune, J.: Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909 (2015)
20. Pugh, J.K., Soros, L.B., Stanley, K.O.: Quality diversity: A new frontier for evolutionary computation. Frontiers in Robotics and AI **3**, 40 (2016)
21. Raileanu, L.E., Stoffel, K.: Theoretical comparison between the gini index and information gain criteria. Annals of Mathematics and Artificial Intelligence **41**(1), 77–93 (2004)
22. Tanigawa, T., Zhao, Q.: A study on efficient generation of decision trees using genetic programming. In: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO). pp. 1047–1052. ACM (2000)
23. Van Erp, M., Vuurpijl, L., Schomaker, L.: An overview and comparison of voting methods for pattern recognition. In: Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. pp. 195–200. IEEE (2002)