# Evolutionary Design of Decision-Tree Algorithms Tailored to Microarray Gene Expression Data Sets

Rodrigo C. Barros, Márcio P. Basgalupp, Alex A. Freitas, and André C. P. L. F. de Carvalho

***Abstract*—Decision-tree induction algorithms are widely used in machine learning applications in which the goal is to extract knowledge from data and present it in a graphically intuitive way. The most successful strategy for inducing decision trees is the greedy top-down recursive approach, which has been continuously improved by researchers over the past 40 years. In this paper, we propose a paradigm shift in the research of decision trees: instead of proposing a new manually designed method for inducing decision trees, we propose automatically designing decision-tree induction algorithms tailored to a specific type of classification data set (or application domain). Following recent breakthroughs in the automatic design of machine learning algorithms, we propose a hyper-heuristic evolutionary algorithm called hyper-heuristic evolutionary algorithm for designing decision-tree algorithms (HEAD-DT) that evolves design components of top-down decision-tree induction algorithms. By the end of the evolution, we expect HEAD-DT to generate a new and possibly better decision-tree algorithm for a given application domain. We perform extensive experiments in 35 real-world microarray gene expression data sets to assess the performance of HEAD-DT, and compare it with very well known decision-tree algorithms such as C4.5, CART, and REPTree. Results show that HEAD-DT is capable of generating algorithms that significantly outperform the baseline manually designed decision-tree algorithms regarding predictive accuracy and F-measure.**

***Index Terms*—Automatic algorithm design, classification, decision trees, evolutionary algorithms, hyper-heuristics, machine learning.**

## I. Introduction

CLASSIFICATION is a machine learning task that aims at building class distribution models by taking into account a set of instances characterized by predictive attributes. The outcome of such a model is used for assigning class labels to new instances that are described only by the values of their predictive attributes. The set of instances whose class distribution is known is called the training set. A classification algorithm usually performs two steps: induction and deduction [1]. In the induction step, it makes use of the training set to induce a model—abstract knowledge representation—which is, in turn, employed for classifying instances whose class information is unknown (deduction step).

Among the most well known classifiers are artificial neural networks, support vector machines (SVMs), decision rules, and decision trees. Decision trees, in particular, are widely used as a comprehensible classification model, since they can be easily represented in a graphical form and can also be represented as a set of classification rules, which generally can be expressed in natural language in the form of IF-THEN rules. They are often the preferred method in application domains in which understanding the reasons that lead to a certain prediction is equally or more important than the prediction itself. Examples of such domains include medical diagnostics [2] and protein function prediction [3].

Finding a (near)-optimal decision tree for a given data set is a challenging problem because the number of possible trees for a given data set grows exponentially with the number of attributes. It has been shown that constructing a decision tree with a minimal number of tests for classifying a new instance is an NP-complete problem [4]. It has also been proved that finding a minimal decision tree consistent with the training set is NP-hard [5], which is also the case for the task of finding the minimal decision tree equivalent to a given decision tree [6] and of building the optimal decision tree from decision tables [7]. The aforementioned studies indicate that finding the optimal decision tree for a given data set (using a brute-force approach that evaluates all possible decision trees) is feasible only for very simple problems. Therefore, heuristic procedures are necessary for finding a near-optimal decision tree in difficult problems.

Many different types of heuristic procedures have been proposed in the literature for building decision trees, such as hill-climbing bottom-up induction [8], hybrid induction [9], evolutionary induction [10]–[13], and ensemble of decision trees [14], to name just a few. In practice, the most used strategy, by far, is the heuristic greedy top-down induction strategy. This strategy has a low computational cost and often produces decision trees with a good tradeoff between accuracy and size, facilitating their interpretation by users.

In essence, a greedy top-down decision-tree induction algorithm starts by considering all instances at the root node,

Manuscript received January 8, 2013; revised September 13, 2013; accepted November 12, 2013. Date of publication November 20, 2013; date of current version November 26, 2014. This work was supported in part by the Brazilian research agency FAPESP under Grant 2009/14325-3 and in part by the Brazilian research agency CNPq.

R. C. Barros is with the Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre 90619-900, Brazil (e-mail: rodrigo.barros@pucrs.br).

A. C. P. L. F. de Carvalho is with the Department of Computer Sciences, University of São Paulo, São Carlos 13566-590, Brazil (e-mail: andre@icmc.usp.br).

M. P. Basgalupp is with the Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo, São José dos Campos 12231-280, Brazil (e-mail: basgalupp@unifesp.br).

A. A. Freitas is with the Computer Science Department, University of Kent, Canterbury CT2 7NF, U.K. (e-mail: a.a.freitas@kent.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2013.2291813

and then it recursively decides whether the set of instances in the current node should be split into subsets, or if no further splitting of the current set of instances is needed. This decision is made by greedily optimizing the value of a heuristic splitting criterion and also taking into account a stopping criterion, for deciding when tree growth should halt. Further improvements over this basic strategy include pruning tree nodes for enhancing the decision tree's capability of dealing with noisy data, and strategies for dealing with missing values.

A great number of approaches were proposed in the literature for implementing each of these design components of greedy top-down decision-tree algorithms. For instance, many different heuristic splitting criteria were proposed, as well as many different strategies for selecting multiple attributes for composing a tree node's test and for pruning a decision tree [15], [16]. It is clear that by improving these design components, we can obtain more robust top-down decision-tree induction algorithms.

This paper innovates in proposing a paradigm shift. Instead of manually improving the design components of a decision-tree algorithm as it has been done for the past 40 years, we propose making use of a hyper-heuristic evolutionary algorithm for optimally combining design components from decision-tree algorithms. By doing so, the hyper-heuristic automatically designs new decision-tree algorithms, which can be tailored to a particular type of data set, associated with a given application domain. We believe a hyper-heuristic is capable of providing a faster less-tedious—and at least equally effective—strategy for improving decision-tree algorithms for particular application domains, as suggested by the computational results of our experiments.

In this paper, we focus on the application domain of microarray gene expression data sets, but the basic idea of our approach is, of course, generic enough to be applicable to any other specific type of application domain. Microarray gene expression data sets contain data about the levels of expression of many thousands of genes, measured using high-throughput technology. This type of data set is an important and popular application of machine learning and data mining methods for at least two reasons. First, such data sets are believed to contain valuable hidden patterns that can be discovered by classification algorithms in order to help biomedical researchers better understand relationships between gene expression levels and diseases (such as cancer, the type of disease with which the data sets used in this paper are associated). Second, microarray gene expression data sets, in general, have a number of attributes (genes) much larger than the number of instances (samples of biological tissues), which constitutes a significant and interesting challenge for classification algorithms.

Note that this paper is a thorough extension of a previous conference paper [17], in which we first presented hyper-heuristic evolutionary algorithm for designing decision-tree algorithms (HEAD-DT). In terms of the functionality of HEAD-DT, the main difference between this paper and our previous conference is as follows. In [17], HEAD-DT searched for the best decision-tree algorithm for a single target data set. Hence, HEAD-DT's fitness function was a measure of a candidate decision-tree algorithm's accuracy on a single data subset, part of the original target data set. After a HEAD-DT run was over, the evolved decision-tree algorithm had its predictive accuracy evaluated on a separate hold-out subset of the same target data set. That is, the goal was to evolve a decision-tree algorithm, very specifically tailored to a single target data set. In contrast, in this paper, HEAD-DT has a different, and intuitively more difficult goal, namely evolving the best decision-tree algorithm for a type of data set associated with a given application domain (which includes many different data sets with similar structural or statistical characteristics). Hence, in this paper, HEAD-DT's fitness function is, in general, an overall measure of a candidate decision-tree algorithms's accuracy across a number of different data sets. After a HEAD-DT run is over, the evolved decision-tree algorithm has its predictive performance evaluated on a separate different set of data sets (of the same type as the data sets used during the HEAD-DT run).

In summary, in this paper, the decision-tree algorithm evolved by HEAD-DT is expected to have good generalization ability across many different data sets of a given type (the data set type in which a given user is interested), unlike our previous work in [17], where such generalization across data sets was not needed. In addition to this important difference, this paper also extends our previous conference paper in three ways.

1) It performs a more extensive set of computational experiments to evaluate HEAD-DT using 35 real-world microarray gene-expression data sets [18].
2) It extends the time complexity analysis of HEAD-DT with a discussion on the cost-effectiveness of automated versus manual algorithm design.
3) It reports the decision-tree components that were most frequently selected by HEAD-DT in order to create decision-tree algorithms customized to microarray data sets.

This paper is organized as follows. We present HEAD-DT in detail in Section II. In Section III, we perform a thorough experimental analysis, comparing the algorithms evolved by HEAD-DT with three well known top-down decision-tree induction algorithms, namely, C4.5 [19], classification and regression trees (CART) [20], and REPTree [21]. We discuss related work in Section IV, and present our conclusions and future work directions in Section V.

## II. HEAD-DT

HEAD-DT is a hyper-heuristic designed for automatically designing complete top-down decision-tree induction algorithms, providing an alternative to the manual design of such algorithms that is the current approach in the decision tree literature.

According to the definition in [22], a hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems. Indeed, hyper-heuristics can automatically generate new heuristics that are tailored to a given problem or type of problem. This is done by combining, through an evolutionary algorithm, components or building-blocks of human-designed heuristics. The motivation behind hyper-heuristics is to raise the level of generality at

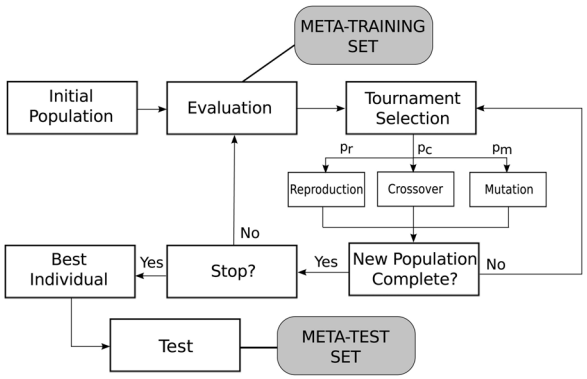



Fig. 1. HEAD-DT evolutionary scheme.

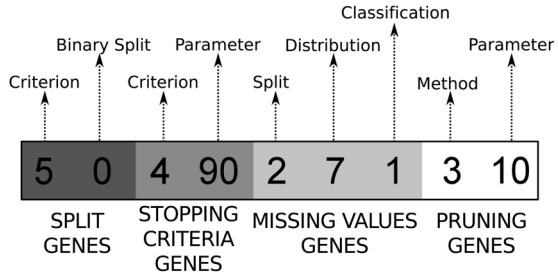



Fig. 2. Linear-genome for evolving decision-tree algorithms.

which search methodologies can operate. In the context of decision trees, instead of having an evolutionary algorithm searching for the best decision tree to a given problem (a regular meta-heuristic approach, e.g., [13], [23]), the generality level is raised by having an evolutionary algorithm searching for the best decision-tree induction algorithm that may be effectively applied to several different classification problems (a hyper-heuristic approach).

HEAD-DT can be seen as a regular generational evolutionary algorithm in which individuals are collections of building blocks of top-down decision-tree induction algorithms. In Fig. 1, we present its evolutionary scheme. We employ typical operators from evolutionary algorithms, such as tournament selection, mutually exclusive genetic operators (reproduction, crossover, and mutation), and a typical stopping criterion that halts evolution after a predefined number of generations.

Each individual in HEAD-DT is encoded as an integer vector (see Fig. 2), and each gene can take on a value in a predefined range of values. We divided the set of genes in four categories that represent the major types of building blocks (design components) of a top-down decision-tree induction algorithm, namely: 1) split genes; 2) stopping criteria genes; 3) missing value genes; and 4) pruning genes.

### A. Split Genes

The linear genome that encodes individuals in HEAD-DT holds two genes for the split component of decision trees. These genes represent the design component that is responsible for selecting the attribute to split the data in the current node of the decision tree. Based on the selected attribute, a decision rule is generated for filtering the input data in subsets (each subset being assigned to a new child node), and the process continues recursively.

To model this design component, we used two different genes. The first one, criterion, is an integer that indexes one of the 15 splitting criteria that we implemented: information gain [24], Gini index [20], global mutual information [25], G statistics [26], Mantaras criterion [27], hypergeometric distribution [28], Chandra–Varghese criterion [29], DCSM [30], $\chi^2$ [31], mean posterior improvement [32], normalized gain [33], orthogonal criterion [34], twoing [20], CAIR [35], and gain ratio [19].

The most widely used criteria are based on Shannon's entropy [36], a concept well known in information theory. Entropy has the interesting property of being a unique function that satisfies the four axioms of uncertainty. In essence, it represents the average amount of information when coding each class into a codeword with ideal length according to its probability. Examples of splitting criteria that are based on entropy are the global mutual information [25] and information gain [24]. The latter is employed by Quinlan in his ID3 system [24]. However, Quinlan points out that information gain is biased toward attributes with many values, and thus proposes a solution called gain ratio [19]. Gain ratio normalizes the information gain by the entropy of the attribute being tested. Several variations of the gain ratio have been proposed, such as the normalized gain [33].

A different type of split criteria are distance-based measures, which evaluate separability, divergency, or discrimination between classes. Such measures include the Gini index [20], the twoing criterion [20], and the orthogonality criterion [34]. We also implemented as options for HEAD-DT less popular split criteria such as CAIR [35] and mean posterior improvement [32], and the more recent Chandra–Varghese [29] and DCSM [30], to improve the flexibility of HEAD-DT, providing the system with a greater diversity of candidate methods for generating splits in a decision tree.

The second gene that controls the split component of a decision-tree algorithm is binary split. This is a binary gene that indicates whether the splits of a decision tree will be binary or multiway. In a binary tree, every parent node is split into two child nodes, which means that, in the case of nominal attributes with many categorical values, those values are divided into two subsets. In a multiway tree, the set of values of a nominal attribute is divided according to that set's cardinality—i.e., one child node is created for each categorical value of the selected attribute. Numeric attributes always partition the tree into two subsets, represented by tests $att \leq \Delta$ and $att > \Delta$, where $att$ is the selected split attribute and $\Delta$ is a threshold automatically determined by the decision-tree algorithm.

### B. Stopping Criteria Genes

The top-down induction of decision trees is recursively executed until a stopping criterion is satisfied. The linear genome in HEAD-DT holds two genes for representing this design component: criterion and parameter.

The first gene, criterion, chooses one out of five different strategies for stopping the tree growth, namely the following.

1) Reaching class homogeneity: when all instances that reach a given tree node have the same class label, there is

no reason to split the instances in that node any further. This can be the only stopping criterion used by the decision-tree algorithm, or it can be combined with the next four stopping criteria.
2) Reaching the maximum tree depth: a parameter tree depth can be specified to avoid that trees become too deep. Very deep trees are difficult to be interpreted by users, and they increase the degree of data fragmentation, possibly leading to overfitting. Range: [2, 10] levels.
3) Reaching the minimum number of instances for a nonterminal node: a parameter specifying such a minimum number can be used to mitigate the data fragmentation problem in decision trees. Range: [1, 20] instances.
4) Reaching the minimum percentage of instances for a nonterminal node: this parameter is conceptually similar to the previous one, but it specifies a percentage (rather than the actual number) of instances. Range: [1%, 10%] of the total number of instances in the training set.
5) Reaching an accuracy threshold within a node: a parameter accuracy reached can be specified so that the set of instances in a node is not split any further. In this strategy, the current node is declared a terminal node when the local accuracy is greater than or equal to a given accuracy threshold. By local accuracy we mean the relative frequency of the most frequent class among the instances belonging to that particular node. Range: {70%, 75%, 80%, 85%, 90%, 95%, 99%} accuracy.

The gene parameter dynamically adjusts a value in the range [0, 100] to the corresponding strategy. For example, if the strategy selected by the gene criterion is reaching the maximum tree depth, the following mapping function is executed:

$$param = (value \bmod 9) + 2. \quad (1)$$

This function maps from [0, 100] (variable value) to [2, 10] (variable param), which is the desired range of values for the parameter of strategy reaching the maximum tree depth. Similar mapping functions are executed dynamically to adjust the ranges of gene parameter.

### C. Missing Values Genes

The next design component of decision trees that is represented in the linear genome of HEAD-DT is the missing value treatment. Missing values can be a significant problem during the process of decision-tree induction and when using the induced tree to classify new instances. We make use of three genes to represent missing value strategies in different moments of the induction/deduction process. During tree induction, there are two moments in which we need to deal with missing values: splitting criterion evaluation (split gene), and instances distribution (distribution gene). During tree deduction (classification), we may also have to deal with missing values in the test set (classification gene).

The following strategies were implemented as possible solutions for the case of missing values during the split criterion evaluation in node $t$ based on attribute $a_i$.

1) ignoring all instances whose value of $a_i$ is missing [20], [37];
2) replacing all missing value symbols occurring in instances in node $t$ with the mode (for nominal attributes) or the mean/median (for numeric attributes) [38];
3) similar to the previous item, but computing the mode or the mean/median only among the instances in $t$ whose class attribute is the same as the instance whose $a_i$ missing value is being replaced [39];
4) weighting the splitting criterion value (for $a_i$ in $t$) by the proportion of missing values [40].

In order to decide which child node training instance $x_j$ should be assigned to, for a split in node $t$ based on $a_i$, we implemented the options:

1) ignoring instance $x_j$ [24];
2) assigning to $x_j$ the most common value of $a_i$ (mode or mean), regardless of the class [40];
3) assigning to $x_j$ the most common value of $a_i$ (mode or mean) considering the instances that belong to the same class as $x_j$;
4) assigning $x_j$ to all child nodes [37];
5) assigning $x_j$ to the child node with the largest number of instances [40];
6) weighting $x_j$ according to the partition probability [19], [41];
7) assigning instance $x_j$ to the most probable partition, considering the class of $x_j$ [39].

Finally, for classifying an unseen test instance $x_j$, considering a split in node $t$ over $a_i$, we implemented the options:

1) exploring all branches of $t$ and combining the results [42];
2) taking the route to the most probable partition (largest subset of instances);
3) halting the classification process and assigning instance $x_j$ to the most frequent class among instances in node $t$ [40].

### D. Pruning Genes

Pruning is important to improve decision tree accuracy in many noisy data sets [20], [24], [43]. It helps avoid over-fitting to the training set and reduce the size of a decision tree (which can make it simpler to be interpreted by users).

HEAD-DT holds two genes for this design component. The first gene, method, indexes one of the five well known approaches for pruning a decision tree shown below and the option of not pruning at all:

1) reduced error pruning (REP) [43];
2) pessimistic error pruning (PEP) [43];
3) minimum error pruning (MEP) [44], [45];
4) cost-complexity pruning (CCP) [20];
5) error-based pruning (EBP) [19].

The second gene, parameter, is in the range [0, 100] and its value is again dynamically mapped by a function according to the pruning method selected.

REP differs from the other methods mainly in using a pruning set (a subset of the original training set) to evaluate the quality of a given subtree. PEP, MEP, and EBP use different approaches to compute an estimate of the classification error associated with a given subtree. CCP differs from the other

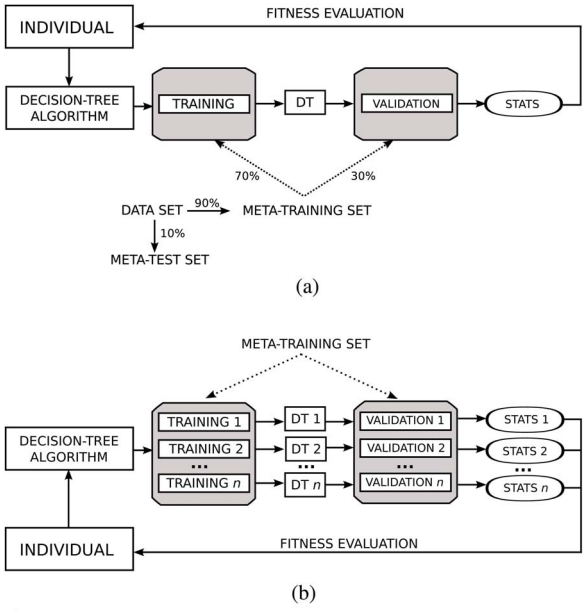



Fig. 3. Fitness evaluation schemes. (a) Fitness evaluation from one data set in the meta-training set. (b) Fitness evaluation from multiple data sets in the meta-training set.

methods in using a pruning criterion that is based on finding a good trade-off between the classification error of a given subtree and its size. For details, the reader is referred to the original references cited above.

### *E. Fitness Evaluation*

In order to compute the fitness of each individual (candidate decision-tree induction algorithm) during the evolutionary process, HEAD-DT employs a meta-training set. In contrast, the meta-test set is used to assess the quality of the evolved decision-tree induction algorithm, which is the best individual produced by HEAD-DT, as shown in Fig. 1. Note that there is no overlapping of instances between the meta-training and meta-test sets, which allows us to measure the generalization ability of the evolved decision-tree induction algorithm, as usual in machine learning. Broadly speaking, there are two distinct approaches for creating the meta-training and test sets, namely:

1) evolving a decision-tree induction algorithm tailored to only one specific data set (the specific approach);
2) evolving a decision-tree induction algorithm from multiple data sets (the general approach).

The specific approach was explored in our previous work [17]. In this approach, the meta-training and meta-test sets are the conventional training and test data obtained from a given data set for machine learning experiments [see Fig. 3(a)]. Note that, in this case, the training and test sets are subsets of the same original data set. More precisely, the training and test sets are described by exactly the same set of predictor attributes, but they have different sets of instances (with no overlapping, as mentioned earlier).

In the general approach, which is addressed for the first time in this paper, we have multiple data sets comprising the meta-training set, and possibly multiple (but different) data sets comprising the meta-test set [see Fig. 3(b)]. In this approach, each data set is described by a different set of predictive attributes, so each data set corresponds to a different classification problem. This approach can be employed with two different purposes.

The first approach is automatically designing a decision-tree algorithm that performs reasonably well in a wide variety of data set types. In other words, the evolved algorithm will be applied to data sets with very different structural characteristics (i.e., very different numbers and types of predictor attributes) and/or from very distinct application domains (e.g., data sets from medicine, finance, marketing, etc.). For this scenario, the user chooses these distinct data sets that will be part of the meta-training set, in the hope that evolution is capable of generating an algorithm that performs well in a wide range of data sets. Pappa [46] calls this strategy evolving robust algorithms. Note that this is the strategy normally used in the manual design of decision-tree algorithms by human machine learning researchers.

The second approach is designing a decision-tree algorithm that is tailored to a particular application domain or to a specific data distribution profile describing a well defined type of data set. In this scenario, which is the new scenario addressed in this paper, the meta-training set is comprised of data sets that share similarities, and so the evolved decision-tree algorithm will be tailored to solving a specific type of classification problem. Unlike the previous strategy, in this case, we have to define a similarity criterion for creating specialized algorithms, which is not trivial. We highlight here some possible similarity criteria: 1) choosing data sets that share the same application domain (e.g., gene expression data); 2) choosing data sets with provenance resemblance (e.g., data sets generated from data collected by a specific sensor or set of sensors); and 3) choosing data sets with structural resemblance (e.g., data sets with statistically similar features and/or with similar geometrical complexity [47], [48]).

In Fig. 3(b), we can observe how the fitness evaluation of a decision-tree induction algorithm constructed from multiple data sets occurs. First, a given individual is mapped into its corresponding decision-tree induction algorithm. Next, each data set from the meta-training set is partitioned into a training set and a validation set—typical values are 70% for training and 30% for validation [21]. The term validation set is used here instead of test set to avoid confusion with the meta-test set, and also due to the fact that we are using the measure of accuracy of a candidate decision-tree induction algorithm on those validation sets to guide the evolutionary search for a better decision-tree algorithm. The same cannot be done with test sets, which are exclusively used for assessing the predictive performance of an evolved decision-tree algorithm (which is output as the result of an entire run of HEAD-DT).

After dividing each data set from the meta-training set into training and validation, we induce a decision tree for each training set available. For evaluating the predictive performance of these decision trees, we use the corresponding validation sets. Statistics regarding the predictive performance and the size of each decision tree are recorded (e.g., accuracy,

F-measure, precision, recall, total number of nodes/leaves), and can be used individually or combined as the fitness function of HEAD-DT. The simple average is probably the most intuitive way of combining the values per data set, but other possible solutions are the median of the values or the harmonic mean. Depending on the data sets used in the meta-training set, the user may decide to give greater weight of importance to a more difficult-to-solve data set than to an easier one, and hence a weighted scheme may be a good solution when combining the data set values. Finally, one may think of any multiobjective optimization strategy for dealing with all values simultaneously [49] (e.g., Pareto dominance or lexicographic analysis).

In this paper, we use as a fitness function the average F-measure of the decision trees generated by a given individual for each data set in the meta-training set. The well known F-measure (also known as F-score or F1 score) is the harmonic mean of precision and recall, as

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2}$$

$$precision = \frac{tp}{tp + fp} \tag{3}$$

$$recall = \frac{tp}{tp + fn} \tag{4}$$

$$fmeasure = 2 \times \frac{precision \times recall}{precision + recall} \tag{5}$$

$$Fitness = \frac{1}{n} \sum_{i=1}^{n} fmeasure_i \tag{6}$$

where $tp$ ($tn$) is the numbers of true positives (negatives) in the validation set, $fp$ ($fn$) is the numbers of false positives (negatives) in the validation set, $fmeasure_i$ is the F-measure obtained in data set $i$, and $n$ is the total number of data sets in the meta-training set.

These equations are directly applicable in the case of binary classification problems, i.e., the case where a data set has only two classes: positive and negative. Nevertheless, they can be trivially extended to multiclass problems. For instance, we can compute the value of a measure for each class—assuming each class to be the positive class in turn, and considering all the other classes as the negative class—and then compute a (weighted) average of the per-class measure.

Although the accuracy measure (2) is still a very popular measure of predictive performance, it is important to notice that accuracy tends to be a misleading measure in data sets with a very unbalanced class distribution. For instance, suppose we are classifying a data set whose class distribution is very skewed: 10% of the instances belong to the positive class and 90% to the negative class. An algorithm that always classifies instances as belonging to the negative class would achieve 90% of accuracy, even though it never predicts the positive class. In this case, assuming that the positive class is equally important to (or even more so than) the negative class, we would prefer an algorithm with a somewhat lower accuracy measure, but which correctly predict some instances as belonging to the rare positive class.

Having in mind that most data sets used in our experiments have very unbalanced class distributions, the average F-measure is a more suitable fitness function than, say, the average accuracy, since it is well known that the F-measure copes much better with unbalanced class-distribution problems than the accuracy measure.

## III. Experimental Analysis

In this section, we present a thorough experimental analysis for assessing the relative performance of the algorithms designed by HEAD-DT. In Section III-A, we present the baseline algorithms that will be compared with HEAD-DT. Section III-B details the 35 real-world gene expression data sets employed in the analysis. Section III-C describes the methodology used for selecting the meta-training and meta-test sets. Section III-D introduces the statistical tests that will be used for analyzing whether there are significant differences among the performance of the algorithms. Section III-E shows our strategy for optimizing the evolutionary parameters of HEAD-DT, as well as the selected parameters. Sections III-F and III-G present the results and a discussion of the experimental analysis respectively. Section III-I comments on the theoretic and empirical time complexity of HEAD-DT, and it also discusses the cost-effectiveness of automated algorithm design. Finally, Section III-J depicts an example of automatically designed decision-tree algorithm.

### A. Baseline Algorithms

The most well known and employed top-down algorithms for decision-tree induction are CART [20] and C4.5 [19]. CART was developed in [20]. It generates binary trees that are recursively created, and the worth of a split is measured by either the Gini index or the twoing criterion. It performs cost-complexity pruning, and further allows the creation of oblique decision splits. C4.5 was developed by Quinlan as an evolution of his prior work, ID3 [24]. It allows multiway splits instead of the typical binary strategy, and employs the gain ratio criterion for splitting nodes. It performs error-based pruning and has robust strategies for dealing with missing values. Its commercial version, C5.0, offers improvements such as multithread computation and tree ensemble generation (which are beyond the scope of this paper).

In this paper, we make use of both CART and C4.5 as the baseline algorithms in the experimental analysis. We employ their java versions available from the Weka machine learning toolkit [21] under the names of SimpleCART and J48. Moreover, we also compare HEAD-DT with the REPTree algorithm, which is a variation of C4.5 that employs reduced-error pruning, also available from the Weka toolkit.

For all baseline algorithms, we employ their default parameters, since they were carefully optimized by their respective authors throughout several years. Note that, in the supervised machine learning literature, the common approach is to find the optimal parameters for being used in a variety of distinct data sets, instead of optimizing the algorithm for each specific data set. Therefore, this is the approach we also take for HEAD-DT, as it will be presented in Section III-E.

TABLE I
SUMMARY OF THE 35 GENE EXPRESSION DATA SETS. FOR EACH DATA SET, WE PRESENT TYPE OF MICROARRAY CHIP, THE TOTAL NUMBER OF INSTANCES, TOTAL NUMBER OF ATTRIBUTES, IMBALANCED-CLASS RATIO (RATE BETWEEN OVER- AND UNDER-REPRESENTED CLASS), AND TOTAL NUMBER OF CLASSES

| | Data Set | Chip | # Instances | # Attributes | IR | # Classes |
|---|---|---|---|---|---|---|
| Parameter Optimisation | armstrong-v2 | Affy | 72 | 2193 | 1.40 | 3 |
| | bredel | cDNA | 50 | 1738 | 6.20 | 3 |
| | dyrskjot | Affy | 40 | 1202 | 2.22 | 3 |
| | garber | cDNA | 66 | 4552 | 10.00 | 4 |
| | golub-v2 | Affy | 72 | 1867 | 4.22 | 3 |
| | gordon | Affy | 181 | 1625 | 4.84 | 2 |
| | khan | cDNA | 83 | 1068 | 2.64 | 4 |
| | laiho | Affy | 37 | 2201 | 3.63 | 2 |
| | pomeroy-v2 | Affy | 42 | 1378 | 2.50 | 5 |
| | ramaswamy | Affy | 190 | 1362 | 3.00 | 14 |
| | su | Affy | 174 | 1570 | 4.67 | 10 |
| | tomlins-v2 | cDNA | 92 | 1287 | 2.46 | 4 |
| | yeoh-v1 | Affy | 248 | 2525 | 4.77 | 2 |
| | yeoh-v2 | Affy | 248 | 2525 | 5.27 | 6 |
| Experiments | alizadeh-v1 | cDNA | 42 | 1094 | 1.00 | 2 |
| | alizadeh-v2 | cDNA | 62 | 2092 | 4.67 | 3 |
| | alizadeh-v3 | cDNA | 62 | 2092 | 2.33 | 4 |
| | armstrong-v1 | Affy | 72 | 1080 | 2.00 | 2 |
| | bhattacharjee | Affy | 203 | 1542 | 23.17 | 5 |
| | bittner | cDNA | 38 | 2200 | 1.00 | 2 |
| | chen | cDNA | 179 | 84 | 1.39 | 2 |
| | chowdary | Affy | 104 | 181 | 1.48 | 2 |
| | golub-v1 | Affy | 72 | 1867 | 1.88 | 2 |
| | lapointe-v1 | cDNA | 69 | 1624 | 3.55 | 3 |
| | lapointe-v2 | cDNA | 110 | 2495 | 3.73 | 4 |
| | liang | cDNA | 37 | 1410 | 9.34 | 3 |
| | nutt-v1 | Affy | 50 | 1376 | 2.14 | 4 |
| | nutt-v2 | Affy | 28 | 1069 | 1.00 | 2 |
| | nutt-v3 | Affy | 22 | 1151 | 2.14 | 2 |
| | shipp-v1 | Affy | 77 | 797 | 3.05 | 2 |
| | pomeroy-v1 | Affy | 34 | 856 | 2.78 | 2 |
| | risinger | cDNA | 42 | 1770 | 6.33 | 4 |
| | singh | Affy | 102 | 338 | 1.04 | 2 |
| | tomlins-v1 | cDNA | 104 | 2314 | 2.67 | 5 |
| | west | Affy | 49 | 1197 | 1.04 | 2 |

### B. Data Sets

To assess the relative performance of the algorithms automatically designed by HEAD-DT, we use a set of 35 publicly available data sets from microarray gene expression data, described in [18].[1] Microarray technology enables expression level measurement for thousands of genes in parallel, given a biological tissue of interest. The data sets employed here are related to different types or subtypes of cancer. The classification task refers to labeling different samples (instances) according to their gene (attribute) expression levels. The main structural characteristics of the 35 data sets are summarized in Table I.

Microarray technology is generally available in two different types of platforms: single-channel microarrays (e.g., Affymetrix) or double-channel microarrays (e.g., cDNA). The type of microarray chip from each data set is in the second column of Table I. Measurements of Affymetrix arrays are estimates on the number of RNA copies found in the cell sample, whereas cDNA microarrays values are ratios of the number of copies in relation to a control cell sample. As in [18], all genes with expression level below 10 are set to a minimum threshold of 10 in the Affymetrix data. The maximum threshold is set at 16 000. This is because values below or above these thresholds are often said to be unreliable [50]. Still for the case of Affymetrix data, the following procedure is applied in order to remove uninformative genes: for each gene $j$ (attribute), compute the mean $m_j$ among the samples (instances). In order to get rid of extreme values, the first 10% largest and smallest values are discarded. Based on such a mean, every value $x_{ij}$ of gene $i$ and sample $j$ is transformed as follows: $y_{ij} = \log_2(x_{ij}/m_j)$. We then selected genes with expression levels differing by at least $l$-fold in at least $c$ samples from their mean expression level across the samples. With few exceptions, the parameters $l$ and $c$ were selected in order to produce a filtered data set with at least 10% of the original number of genes.[2] It should be noticed that the transformed data is only used in the filtering step. A similar filtering procedure was applied for the cDNA data, but without the log transformation. In the case of cDNA microarray data sets, genes with more than 10% of missing values were discarded. The remaining genes that still presented missing values had them replaced by its respective mean value.

Finally, note that we randomly divided the 35 data sets into two groups: parameter optimization and experiments. The 14 data sets in the parameter optimization group are used for tuning the evolutionary parameters of HEAD-DT. The remaining 21 data sets from the experiments group are used for evaluating the performance of the algorithms automatically designed by HEAD-DT.

### C. Building the Meta-Training and Meta-Test Sets

For selecting which data sets will be part of the meta-training and meta-test sets, we have adopted the following methodology. Randomly choose 1 data set from the available set of data sets to be part of the meta-training set. Then, execute HEAD-DT with the selected data set in the meta-training set and the remaining data sets in the meta-test set. For the next experiment, select two additional data sets that were previously part of the meta-test set, and move them to the meta-training set, which now will be comprised of three data sets. This procedure is repeated until we have nine data sets being part of the meta-training set.

Following this methodology, the 14 parameter optimization data sets are arranged in five different experimental configurations {#training sets, #test sets}: {1 x 13}, {3 x 11}, {5 x 9}, {7 x 7}, and {9 x 5}. Similarly, the 21 data sets in the experiments group are arranged in also five different experimental configurations {#training sets, #test sets}: {1 x 20}, {3 x 18}, {5 x 16}, {7 x 14}, and {9 x 12}. Table II presents the randomly selected data sets according to the configurations detailed above.

### D. Statistical Analysis

In order to provide some reassurance about the validity and nonrandomness of the obtained results, we present the results of statistical tests by following the approach proposed in [51]. This approach seeks to compare multiple algorithms on multiple data sets, and it is based on the use of the Friedman test with a corresponding post-hoc test. If the null hypothesis of similar performances is rejected, the Nemenyi post-hoc test

[1]Data sets available at http://algorithmics.molgen.mpg.de/Static/Supplements/CompCancer/datasets.htm

[2]The values of $l$ and $c$ for each data set can be found at http://algorithmics.molgen.mpg.de/Static/Supplements/CompCancer/datasets.htm

TABLE II
META-TRAINING AND META-TEST CONFIGURATIONS. DATA SETS WERE RANDOMLY SELECTED ACCORDING TO THE METHODOLOGY PRESENTED IN SECTION III-C

Data sets for parameter optimisation

| {1 x 13} | | {3 x 11} | | {5 x 9} | | {7 x 7} | | {9 x 5} | |
|---|---|---|---|---|---|---|---|---|---|
| Meta-Training | Meta-Test | Meta-Training | Meta-Test | Meta-Training | Meta-Test | Meta-Training | Meta-Test | Meta-Training | Meta-Test |
| | armstrong-v2 | | armstrong-v2 | | armstrong-v2 | | dyrskjot | bredel | dyrskjot |
| | dyrskjot | | dyrskjot | | dyrskjot | bredel | garber | pomeroy-v2 | garber |
| | garber | | garber | bredel | garber | pomeroy-v2 | golub-v2 | yeoh-v2 | golub-v2 |
| | golub-v2 | | golub-v2 | pomeroy-v2 | golub-v2 | yeoh-v2 | khan | gordon | ramaswamy |
| | gordon | bredel | gordon | yeoh-v2 | khan | gordon | ramaswamy | su | yeoh-v1 |
| | khan | pomeroy-v2 | khan | gordon | laiho | su | tomlins-v2 | armstrong-v2 | |
| bredel | laiho | yeoh-v2 | laiho | su | ramaswamy | armstrong-v2 | yeoh-v1 | laiho | |
| | pomeroy-v2 | | ramaswamy | | tomlins-v2 | laiho | | khan | |
| | ramaswamy | | su | | yeoh-v1 | | | tomlins-v2 | |
| | su | | tomlins-v2 | | | | | | |
| | tomlins-v2 | | yeoh-v1 | | | | | | |
| | yeoh-v1 | | | | | | | | |
| | yeoh-v2 | | | | | | | | |

Data sets for the experimental analysis

| {1 x 20} | | {3 x 18} | | {5 x 16} | | {7 x 14} | | {9 x 12} | |
|---|---|---|---|---|---|---|---|---|---|
| Meta-Training | Meta-Test | Meta-Training | Meta-Test | Meta-Training | Meta-Test | Meta-Training | Meta-Test | Meta-Training | Meta-Test |
| | alizadeh-v1 | | alizadeh-v1 | | alizadeh-v1 | | alizadeh-v1 | | alizadeh-v1 |
| | alizadeh-v2 | | alizadeh-v2 | | alizadeh-v2 | | alizadeh-v2 | | alizadeh-v2 |
| | alizadeh-v3 | | armstrong-v1 | | bhattacharjee | | bhattacharjee | | bhattacharjee |
| | armstrong-v1 | | bhattacharjee | | bittner | | chen | | chen |
| | bhattacharjee | | bittner | | chen | | chowdary | | golub-v1 |
| | bittner | | chen | | chowdary | | golub-v1 | | lapointe-v1 |
| | chen | | chowdary | | golub-v1 | | lapointe-v1 | | liang |
| | chowdary | | golub-v1 | | lapointe-v1 | | liang | | nutt-v1 |
| | golub-v1 | | lapointe-v1 | | liang | | nutt-v1 | | nutt-v2 |
| | lapointe-v1 | | liang | | nutt-v1 | | nutt-v2 | | shipp-v1 |
| nutt-v3 | lapointe-v2 | | nutt-v1 | | nutt-v2 | | shipp-v1 | | pomeroy-v1 |
| | liang | | nutt-v2 | | shipp-v1 | | pomeroy-v1 | | singh |
| | nutt-v1 | | shipp-v1 | | pomeroy-v1 | | singh | | |
| | nutt-v2 | | pomeroy-v1 | | risinger | | west | | |
| | shipp-v1 | | risinger | | singh | | | | |
| | pomeroy-v1 | | singh | | west | | | | |
| | risinger | | tomlins-v1 | | | | | | |
| | singh | | west | | | | | | |
| | tomlins-v1 | | | | | | | | |
| | west | | | | | | | | |

for pairwise comparisons is executed. The performance of two classifiers is significantly different if their corresponding average ranks differ by at least the critical difference given by the Nemenyi test.

### *E. Parameter Optimization*

Considering that HEAD-DT is a regular generational EA (as depicted in Fig. 1), the following parameters have to be chosen prior to evolution: 1) population size; 2) maximum number of generations; 3) tournament selection size; 4) elitism rate; 5) reproduction probability; 6) crossover probability; and 7) mutation probability.

For parameters 1–4, we have defined values commonly used in the literature of evolutionary algorithms for decision-tree induction [23], namely, 100 individuals, 100 generations, tournament between two individuals, and 5% of elitism. For the remaining parameters, since the selected individuals will undergo either reproduction, crossover, or mutation (mutually exclusive operators), we have created a single parameter $p$ that works as follows. We have fixed the reproduction rate in 5%, and set the crossover rate as $p$ and mutation as $1 - p - 0.05$. Hence, $p = 0.9$ means a 90% crossover probability and 5% mutation probability.

We have performed a tuning experiment varying $p$ within $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. For that, we have employed the 14 gene expression data sets belonging to the parameter optimization group to evaluate the best value of $p$. Note that the aim of this experiment is not to optimize the parameters for a particular data set; it is rather to find robust values that work well across the 14 tuning data sets. We then use the robust value of $p$ found in this experiment in the different set of 21 data sets used in our experimental analysis. This evaluates the generalization ability of $p$ across new data sets, unused for parameter tuning, as usual in supervised machine learning research.

In this tuning experiment, we followed the methodology presented in Section III-C, in which we vary the number of data sets in the meta-training set within $\{1, 3, 5, 7, 9\}$, resulting in the following configurations: {1 x 13}, {3 x 11}, {5 x 9}, {7 x 7}, and {9 x 5}. The exact data sets in each of these configurations are presented at the top of Table II.

The fitness function employed by HEAD-DT, as already mentioned, is the average F-measure (6) assessed on the validation sets of the data sets belonging to the meta-training set. To evaluate the predictive performance of the best algorithm evolved by HEAD-DT, we performed a ten-fold cross-validation procedure in each data set belonging to the meta-test set, recording the accuracy and F-measure achieved by each of the corresponding decision trees. Also, to mitigate the randomness effect of evolutionary algorithms, we average the results of five different runs of HEAD-DT, with a different random seed used to initialize the population in each run.

Table III presents the results of the parameter-tuning experiments. We present the average ranking of each version of HEAD-DT (HEAD-$p$) in the corresponding experimental

TABLE III

RESULTS OF THE TUNING EXPERIMENTS. HEAD-DT IS EXECUTED WITH DIFFERENT VALUES OF PARAMETER $p$ (HEAD-$p$). VALUES ARE THE AVERAGE PERFORMANCE (RANK) OF EACH HEAD-DT VERSION IN THE CORRESPONDING META-TEST SET OF TUNING DATA SETS, ACCORDING TO EITHER ACCURACY OR F-MEASURE. THE LOWER THE RANK, THE BETTER THE PERFORMANCE

| Configuration | Rank | HEAD-0.1 | HEAD-0.2 | HEAD-0.3 | HEAD-0.4 | HEAD-0.5 | HEAD-0.6 | HEAD-0.7 | HEAD-0.8 | HEAD-0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| {1 x 13} | Accuracy | 5.73 | 4.23 | 4.54 | 4.92 | 5.19 | 4.42 | 5.31 | 5.62 | 5.04 |
| | F-Measure | 6.15 | 4.85 | 4.31 | 5.00 | 5.12 | 4.12 | 4.75 | 5.62 | 5.08 |
| {3 x 11} | Accuracy | 4.91 | 4.41 | 4.59 | 4.45 | 4.68 | 5.27 | 5.64 | 5.64 | 5.41 |
| | F-Measure | 4.68 | 4.45 | 4.09 | 4.59 | 5.05 | 5.95 | 5.27 | 6.14 | 4.77 |
| {5 x 9} | Accuracy | 5.11 | 6.17 | 5.22 | 6.28 | 4.28 | 5.11 | 4.61 | 3.44 | 4.77 |
| | F-Measure | 4.72 | 6.06 | 4.72 | 6.06 | 4.00 | 5.67 | 4.89 | 3.89 | 5.00 |
| {7 x 7} | Accuracy | 6.07 | 7.43 | 4.71 | 6.50 | 4.86 | 3.36 | 3.79 | 3.50 | 4.79 |
| | F-Measure | 6.21 | 6.71 | 5.14 | 3.21 | 5.14 | 3.71 | 4.00 | 3.21 | 4.50 |
| {9 x 5} | Accuracy | 8.30 | 5.70 | 5.50 | 5.20 | 7.90 | 2.70 | 4.00 | 2.80 | 2.9 |
| | F-Measure | 7.10 | 5.60 | 5.60 | 4.90 | 6.70 | 3.50 | 4.50 | 4.20 | 2.90 |
| | Average | 5.90 | 5.56 | 4.84 | 5.11 | 5.29 | **4.38** | 4.68 | 4.41 | 4.52 |

TABLE IV

RESULTS OF THE {1 X 20} CONFIGURATION. AVERAGE ± STANDARD DEVIATION

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.77± 0.11 | 0.71± 0.16 | 0.68± 0.20 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.88± 0.06 | 0.92± 0.11 | 0.86± 0.15 | 0.84± 0.17 |
| alizadeh-2000-v3 | 0.71± 0.05 | 0.69± 0.15 | 0.71± 0.15 | 0.66± 0.13 |
| armstrong-2002-v1 | 0.90± 0.04 | 0.90± 0.07 | 0.89± 0.06 | 0.91± 0.07 |
| bhattacharjee-2001 | 0.90± 0.03 | 0.89± 0.10 | 0.89± 0.08 | 0.86± 0.06 |
| bittner-2000 | 0.61± 0.09 | 0.53± 0.18 | 0.49± 0.16 | 0.71± 0.22 |
| chen-2002 | 0.85± 0.04 | 0.85± 0.07 | 0.81± 0.07 | 0.79± 0.12 |
| chowdary-2006 | 0.95± 0.03 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.06 |
| golub-1999-v1 | 0.88± 0.03 | 0.86± 0.07 | 0.88± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.66± 0.08 | 0.78± 0.18 | 0.71± 0.14 | 0.63± 0.09 |
| lapointe-2004-v2 | 0.62± 0.05 | 0.68± 0.19 | 0.57± 0.11 | 0.61± 0.15 |
| liang-2005 | 0.89± 0.09 | 0.71± 0.14 | 0.76± 0.19 | 0.78± 0.18 |
| nutt-2003-v1 | 0.53± 0.08 | 0.54± 0.19 | 0.50± 0.17 | 0.40± 0.21 |
| nutt-2003-v2 | 0.84± 0.08 | 0.77± 0.21 | 0.87± 0.17 | 0.45± 0.27 |
| pomeroy-2002-v1 | 0.88± 0.08 | 0.84± 0.17 | 0.88± 0.16 | 0.73± 0.10 |
| risinger-2003 | 0.58± 0.15 | 0.52± 0.15 | 0.53± 0.19 | 0.59± 0.21 |
| shipp-2002-v1 | 0.91± 0.05 | 0.77± 0.12 | 0.84± 0.16 | 0.77± 0.05 |
| singh-2002 | 0.78± 0.04 | 0.74± 0.14 | 0.78± 0.12 | 0.69± 0.15 |
| tomlins-2006 | 0.59± 0.07 | 0.58± 0.20 | 0.58± 0.18 | 0.58± 0.10 |
| west-2001 | 0.89± 0.08 | 0.89± 0.11 | 0.84± 0.13 | 0.74± 0.19 |
| Average Rank | **1.75** | 2.45 | 2.70 | 3.10 |

(a) ACCURACY RESULTS.

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.77± 0.11 | 0.67± 0.21 | 0.63± 0.26 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.89± 0.06 | 0.92± 0.10 | 0.84± 0.17 | 0.80± 0.19 |
| alizadeh-2000-v3 | 0.71± 0.05 | 0.65± 0.18 | 0.68± 0.17 | 0.63± 0.12 |
| armstrong-2002-v1 | 0.90± 0.04 | 0.90± 0.07 | 0.88± 0.06 | 0.91± 0.08 |
| bhattacharjee-2001 | 0.89± 0.02 | 0.87± 0.12 | 0.89± 0.08 | 0.85± 0.07 |
| bittner-2000 | 0.61± 0.09 | 0.49± 0.21 | 0.45± 0.19 | 0.66± 0.28 |
| chen-2002 | 0.85± 0.04 | 0.85± 0.07 | 0.81± 0.07 | 0.78± 0.13 |
| chowdary-2006 | 0.95± 0.03 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.07 |
| golub-1999-v1 | 0.88± 0.03 | 0.86± 0.07 | 0.87± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.64± 0.10 | 0.73± 0.20 | 0.68± 0.15 | 0.50± 0.13 |
| lapointe-2004-v2 | 0.62± 0.05 | 0.66± 0.20 | 0.58± 0.10 | 0.57± 0.15 |
| liang-2005 | 0.87± 0.10 | 0.67± 0.21 | 0.77± 0.22 | 0.71± 0.23 |
| nutt-2003-v1 | 0.53± 0.08 | 0.48± 0.19 | 0.46± 0.16 | 0.31± 0.22 |
| nutt-2003-v2 | 0.84± 0.08 | 0.73± 0.26 | 0.87± 0.17 | 0.34± 0.31 |
| pomeroy-2002-v1 | 0.87± 0.10 | 0.79± 0.22 | 0.85± 0.20 | 0.62± 0.14 |
| risinger-2003 | 0.56± 0.15 | 0.48± 0.19 | 0.53± 0.21 | 0.52± 0.25 |
| shipp-2002-v1 | 0.90± 0.05 | 0.75± 0.13 | 0.83± 0.17 | 0.70± 0.10 |
| singh-2002 | 0.78± 0.04 | 0.72± 0.18 | 0.77± 0.13 | 0.67± 0.18 |
| tomlins-2006 | 0.58± 0.08 | 0.56± 0.20 | 0.56± 0.20 | 0.53± 0.11 |
| west-2001 | 0.89± 0.08 | 0.89± 0.12 | 0.81± 0.17 | 0.71± 0.23 |
| Average Rank | **1.55** | 2.50 | 2.60 | 3.35 |

(b) F-MEASURE RESULTS.

configuration. For instance, when HEAD-0.1 makes use of a single data set for evolving the optimal algorithm ({1 x 13}), its performance in the remaining 13 data sets gives HEAD-0.1 the average rank position of 5.73 regarding the accuracy of its corresponding decision trees, and 6.15 regarding F-measure.

The Friedman and Nemenyi tests did not indicate any statistically significant differences among the nine distinct versions of HEAD-DT, either considering accuracy or F-measure, for any of the experimental configurations. Such a lack of significant differences indicates that HEAD-DT is robust to different values of $p$. For selecting the best value of $p$ to employ in the experimental analysis, we decided to average the results across the different configurations, and also between the two different evaluation measures, accuracy and F-measure. Hence, we calculated the average of the average ranks for each HEAD-$p$ version across the distinct configurations and evaluation measures, and the results are presented at the bottom of Table III. Notice how marginal are the differences among values of $p \geq 0.6$. HEAD-0.6 was then selected as the HEAD-DT version with the best $p$ value, bearing in mind that it presented the lowest average of the average ranks (4.38).

In the next section, we present the results of the experimental analysis performed over the 21 data sets in the experiments group, in which we compare HEAD-0.6 (hereafter called simply HEAD-DT) with the baseline algorithms.

### *F. Results*

Tables IV–VIII show average values of accuracy and F-measure achieved by HEAD-DT, CART, C4.5, and REPTree in configurations {1 x 20}, {3 x 18}, {5 x 16}, {7 x 14}, and {9 x 12}. At the bottom of each table, we present the average rank position (the average of the rank position in each data set) of each method. The lower the ranking, the better the method. For instance, a method capable of outperforming any other method in every data set would have an average rank position of 1.00 (first place).

The first experiment is regarding configuration {1 x 20}. Table IV shows the result for this configuration considering accuracy [see Table IV(a)] and F-measure [see Table IV(b)].

TABLE V
RESULTS OF THE {3 X 18} CONFIGURATION. AVERAGE ± STANDARD DEVIATION

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.74± 0.07 | 0.71± 0.16 | 0.68± 0.20 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.90± 0.07 | 0.92± 0.11 | 0.86± 0.15 | 0.84± 0.17 |
| armstrong-2002-v1 | 0.88± 0.01 | 0.90± 0.07 | 0.89± 0.06 | 0.91± 0.07 |
| bhattacharjee-2001 | 0.92± 0.03 | 0.89± 0.10 | 0.89± 0.08 | 0.86± 0.06 |
| bittner-2000 | 0.55± 0.09 | 0.53± 0.18 | 0.49± 0.16 | 0.71± 0.22 |
| chen-2002 | 0.87± 0.03 | 0.85± 0.07 | 0.81± 0.07 | 0.79± 0.12 |
| chowdary-2006 | 0.97± 0.01 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.06 |
| golub-1999-v1 | 0.89± 0.02 | 0.86± 0.07 | 0.88± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.70± 0.06 | 0.78± 0.18 | 0.71± 0.14 | 0.63± 0.09 |
| liang-2005 | 0.89± 0.08 | 0.71± 0.14 | 0.76± 0.19 | 0.78± 0.18 |
| nutt-2003-v1 | 0.67± 0.10 | 0.54± 0.19 | 0.50± 0.17 | 0.40± 0.21 |
| nutt-2003-v2 | 0.77± 0.08 | 0.77± 0.21 | 0.87± 0.17 | 0.45± 0.27 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.84± 0.17 | 0.88± 0.16 | 0.73± 0.10 |
| risinger-2003 | 0.53± 0.11 | 0.52± 0.15 | 0.53± 0.19 | 0.59± 0.21 |
| shipp-2002-v1 | 0.85± 0.05 | 0.77± 0.12 | 0.84± 0.16 | 0.77± 0.05 |
| singh-2002 | 0.80± 0.04 | 0.74± 0.14 | 0.78± 0.12 | 0.69± 0.15 |
| tomlins-2006 | 0.64± 0.03 | 0.58± 0.20 | 0.58± 0.18 | 0.58± 0.10 |
| west-2001 | 0.92± 0.04 | 0.90± 0.11 | 0.84± 0.13 | 0.74± 0.19 |
| Average Rank | **1.61** | 2.61 | 2.72 | 3.05 |

(a) ACCURACY RESULTS.

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.73± 0.07 | 0.67± 0.21 | 0.63± 0.26 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.91± 0.06 | 0.92± 0.10 | 0.84± 0.17 | 0.80± 0.19 |
| armstrong-2002-v1 | 0.88± 0.02 | 0.90± 0.07 | 0.88± 0.06 | 0.91± 0.08 |
| bhattacharjee-2001 | 0.91± 0.03 | 0.87± 0.12 | 0.89± 0.08 | 0.85± 0.07 |
| bittner-2000 | 0.53± 0.10 | 0.49± 0.21 | 0.45± 0.19 | 0.66± 0.28 |
| chen-2002 | 0.87± 0.03 | 0.85± 0.07 | 0.81± 0.07 | 0.78± 0.13 |
| chowdary-2006 | 0.97± 0.01 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.07 |
| golub-1999-v1 | 0.89± 0.02 | 0.86± 0.07 | 0.87± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.68± 0.08 | 0.73± 0.20 | 0.68± 0.15 | 0.50± 0.13 |
| liang-2005 | 0.87± 0.10 | 0.67± 0.21 | 0.77± 0.22 | 0.71± 0.23 |
| nutt-2003-v1 | 0.65± 0.10 | 0.48± 0.19 | 0.46± 0.16 | 0.31± 0.22 |
| nutt-2003-v2 | 0.77± 0.08 | 0.73± 0.26 | 0.87± 0.17 | 0.34± 0.31 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.79± 0.22 | 0.85± 0.20 | 0.62± 0.14 |
| risinger-2003 | 0.52± 0.11 | 0.48± 0.19 | 0.53± 0.21 | 0.52± 0.25 |
| shipp-2002-v1 | 0.84± 0.05 | 0.75± 0.13 | 0.83± 0.17 | 0.70± 0.10 |
| singh-2002 | 0.80± 0.04 | 0.72± 0.18 | 0.77± 0.13 | 0.67± 0.18 |
| tomlins-2006 | 0.63± 0.03 | 0.56± 0.20 | 0.56± 0.20 | 0.53± 0.11 |
| west-2001 | 0.92± 0.04 | 0.89± 0.12 | 0.81± 0.17 | 0.71± 0.23 |
| Average Rank | **1.61** | 2.61 | 2.56 | 3.22 |

(b) F-MEASURE RESULTS.

TABLE VI
RESULTS OF THE {5 X 16} CONFIGURATION. AVERAGE ± STANDARD DEVIATION

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.75± 0.05 | 0.71± 0.16 | 0.68± 0.20 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.91± 0.07 | 0.92± 0.11 | 0.86± 0.15 | 0.84± 0.17 |
| bhattacharjee-2001 | 0.94± 0.02 | 0.89± 0.10 | 0.89± 0.08 | 0.86± 0.06 |
| bittner-2000 | 0.54± 0.08 | 0.53± 0.18 | 0.49± 0.16 | 0.71± 0.22 |
| chen-2002 | 0.91± 0.02 | 0.85± 0.07 | 0.81± 0.07 | 0.79± 0.12 |
| chowdary-2006 | 0.97± 0.01 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.06 |
| golub-1999-v1 | 0.88± 0.02 | 0.86± 0.07 | 0.88± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.71± 0.06 | 0.78± 0.18 | 0.71± 0.14 | 0.63± 0.09 |
| liang-2005 | 0.89± 0.08 | 0.71± 0.14 | 0.76± 0.19 | 0.78± 0.18 |
| nutt-2003-v1 | 0.61± 0.10 | 0.54± 0.19 | 0.50± 0.17 | 0.40± 0.21 |
| nutt-2003-v2 | 0.78± 0.08 | 0.77± 0.21 | 0.87± 0.17 | 0.45± 0.27 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.84± 0.17 | 0.88± 0.16 | 0.73± 0.10 |
| risinger-2003 | 0.55± 0.08 | 0.52± 0.15 | 0.53± 0.19 | 0.59± 0.21 |
| shipp-2002-v1 | 0.87± 0.03 | 0.77± 0.12 | 0.84± 0.16 | 0.77± 0.05 |
| singh-2002 | 0.78± 0.03 | 0.74± 0.14 | 0.78± 0.12 | 0.69± 0.15 |
| west-2001 | 0.92± 0.03 | 0.90± 0.11 | 0.84± 0.13 | 0.74± 0.19 |
| Average Rank | **1.44** | 2.69 | 2.69 | 3.19 |

(a) ACCURACY RESULTS.

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.75± 0.05 | 0.67± 0.21 | 0.63± 0.26 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.91± 0.06 | 0.92± 0.10 | 0.84± 0.17 | 0.80± 0.19 |
| bhattacharjee-2001 | 0.94± 0.02 | 0.87± 0.12 | 0.89± 0.08 | 0.85± 0.07 |
| bittner-2000 | 0.53± 0.09 | 0.49± 0.21 | 0.45± 0.19 | 0.66± 0.28 |
| chen-2002 | 0.91± 0.02 | 0.85± 0.07 | 0.81± 0.07 | 0.78± 0.13 |
| chowdary-2006 | 0.97± 0.01 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.07 |
| golub-1999-v1 | 0.88± 0.02 | 0.86± 0.07 | 0.87± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.69± 0.08 | 0.73± 0.20 | 0.68± 0.15 | 0.50± 0.13 |
| liang-2005 | 0.87± 0.10 | 0.67± 0.21 | 0.77± 0.22 | 0.71± 0.23 |
| nutt-2003-v1 | 0.60± 0.09 | 0.48± 0.19 | 0.46± 0.16 | 0.31± 0.22 |
| nutt-2003-v2 | 0.78± 0.08 | 0.73± 0.26 | 0.87± 0.17 | 0.34± 0.31 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.79± 0.22 | 0.85± 0.20 | 0.62± 0.14 |
| risinger-2003 | 0.53± 0.08 | 0.48± 0.19 | 0.53± 0.21 | 0.52± 0.25 |
| shipp-2002-v1 | 0.86± 0.03 | 0.75± 0.13 | 0.83± 0.17 | 0.70± 0.10 |
| singh-2002 | 0.78± 0.03 | 0.72± 0.18 | 0.77± 0.13 | 0.67± 0.18 |
| west-2001 | 0.92± 0.03 | 0.89± 0.12 | 0.81± 0.17 | 0.71± 0.23 |
| Average Rank | **1.31** | 2.69 | 2.63 | 3.38 |

(b) F-MEASURE RESULTS.

Note that HEAD-DT is the best performing method with respect to both accuracy and F-measure, reaching an average rank of 1.75 and 1.55, respectively. HEAD-DT is followed by CART (2.45 and 2.50) and C4.5 (2.70 and 2.65), whose performances are quite evenly matched. REPTree is the worst-ranked method for either accuracy (3.10) or F-measure (3.35).

Table V presents the results for configuration {3 x 18}. In this experiment, HEAD-DT's average rank is again the lowest of the experiment: 1.61 for both accuracy and F-measure. That means HEAD-DT is often the best performing method (first place) in the group of 18 test data sets. CART and C4.5 once again present very similar average rank values, which is not surprising bearing in mind they are both considered the state-of-the-art top-down decision-tree induction algorithms. REPTree is again the worst-performing method among the four algorithms.

Table VI presents the results for configuration {5 x 16}. The scenario is quite similar to the previous configurations, with HEAD-DT leading the ranking, with average rank values of 1.44 (accuracy) and 1.31 (F-measure). These are the lowest rank values obtained by a method in any experimental configuration conducted in this analysis. HEAD-DT is followed by CART (2.69 and 2.69) and C4.5 (2.69 and 2.63), which once again present very similar performances. REPTree is at the bottom of the ranking, with 3.19 (accuracy) and 3.38 (F-measure).

Results for configuration {7 x 14} show a different picture for the first time. Table VII(a), which depicts the accuracy values of each method, indicates that HEAD-DT is outperformed by both CART and C4.5, although their average rank values are very similar: 2.14, 2.21, versus 2.36 for HEAD-DT. REPTree keeps its position as worst-performing method, with an average rank value of 3.29. However, Table VII(b) returns to the same scenario presented in configurations {1 x 20}, {3 x 18}, and {5 x 16}: HEAD-DT outperforming the baseline methods, with CART and C4.5 tied in the second place and REPTree in the last position.

The effect seen in Table VII(a), in which HEAD-DT did not outperform the baseline methods, has a straightforward explanation: HEAD-DT optimizes its generated algorithms according to the F-measure evaluation measure. Since accuracy may be a misleading measure (it is not suitable for data sets with unbalanced class distributions), and several of the gene-expression data sets are unbalanced (i.e., they have a large

TABLE VII

RESULTS OF THE {7 X 14} CONFIGURATION. AVERAGE ± STANDARD DEVIATION

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.79± 0.09 | 0.71± 0.16 | 0.68± 0.20 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.89± 0.05 | 0.92± 0.11 | 0.86± 0.15 | 0.84± 0.17 |
| bhattacharjee-2001 | 0.89± 0.03 | 0.89± 0.10 | 0.89± 0.08 | 0.86± 0.06 |
| chen-2002 | 0.84± 0.02 | 0.85± 0.07 | 0.81± 0.07 | 0.79± 0.12 |
| chowdary-2006 | 0.91± 0.04 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.06 |
| golub-1999-v1 | 0.84± 0.03 | 0.86± 0.07 | 0.88± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.61± 0.09 | 0.78± 0.18 | 0.71± 0.14 | 0.63± 0.09 |
| liang-2005 | 0.87± 0.08 | 0.71± 0.14 | 0.76± 0.19 | 0.78± 0.18 |
| nutt-2003-v1 | 0.68± 0.07 | 0.54± 0.19 | 0.50± 0.17 | 0.40± 0.21 |
| nutt-2003-v2 | 0.76± 0.07 | 0.77± 0.21 | 0.87± 0.17 | 0.45± 0.27 |
| pomeroy-2002-v1 | 0.70± 0.12 | 0.84± 0.17 | 0.88± 0.16 | 0.73± 0.10 |
| shipp-2002-v1 | 0.86± 0.03 | 0.77± 0.12 | 0.84± 0.16 | 0.77± 0.05 |
| singh-2002 | 0.79± 0.02 | 0.74± 0.14 | 0.78± 0.12 | 0.69± 0.15 |
| west-2001 | 0.80± 0.07 | 0.90± 0.11 | 0.84± 0.13 | 0.74± 0.19 |
| Average Rank | 2.36 | **2.14** | 2.21 | 3.29 |

(a) ACCURACY RESULTS.

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.79± 0.09 | 0.67± 0.21 | 0.63± 0.26 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.91± 0.03 | 0.92± 0.10 | 0.84± 0.17 | 0.80± 0.19 |
| bhattacharjee-2001 | 0.89± 0.03 | 0.87± 0.12 | 0.89± 0.08 | 0.85± 0.07 |
| chen-2002 | 0.84± 0.02 | 0.85± 0.07 | 0.81± 0.07 | 0.78± 0.13 |
| chowdary-2006 | 0.91± 0.04 | 0.97± 0.05 | 0.95± 0.05 | 0.94± 0.07 |
| golub-1999-v1 | 0.83± 0.04 | 0.86± 0.07 | 0.87± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.60± 0.11 | 0.73± 0.20 | 0.68± 0.15 | 0.50± 0.13 |
| liang-2005 | 0.87± 0.08 | 0.67± 0.21 | 0.77± 0.22 | 0.71± 0.23 |
| nutt-2003-v1 | 0.65± 0.08 | 0.48± 0.19 | 0.46± 0.16 | 0.31± 0.22 |
| nutt-2003-v2 | 0.76± 0.07 | 0.73± 0.26 | 0.87± 0.17 | 0.34± 0.31 |
| pomeroy-2002-v1 | 0.69± 0.13 | 0.79± 0.22 | 0.85± 0.20 | 0.62± 0.14 |
| shipp-2002-v1 | 0.86± 0.03 | 0.75± 0.13 | 0.83± 0.17 | 0.70± 0.10 |
| singh-2002 | 0.79± 0.02 | 0.72± 0.18 | 0.77± 0.13 | 0.67± 0.18 |
| west-2001 | 0.80± 0.07 | 0.89± 0.12 | 0.81± 0.17 | 0.71± 0.23 |
| Average Rank | **2.07** | 2.21 | 2.21 | 3.50 |

(b) F-MEASURE RESULTS.

TABLE VIII

RESULTS OF THE {9 X 12} CONFIGURATION. AVERAGE ± STANDARD DEVIATION

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.79± 0.09 | 0.71± 0.16 | 0.68± 0.20 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.89± 0.05 | 0.92± 0.11 | 0.86± 0.15 | 0.84± 0.17 |
| bhattacharjee-2001 | 0.90± 0.02 | 0.89± 0.10 | 0.89± 0.08 | 0.86± 0.06 |
| chen-2002 | 0.88± 0.03 | 0.85± 0.07 | 0.81± 0.07 | 0.79± 0.12 |
| golub-1999-v1 | 0.87± 0.02 | 0.86± 0.07 | 0.88± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.70± 0.05 | 0.78± 0.18 | 0.71± 0.14 | 0.63± 0.09 |
| liang-2005 | 0.88± 0.08 | 0.71± 0.14 | 0.76± 0.19 | 0.78± 0.18 |
| nutt-2003-v1 | 0.67± 0.09 | 0.54± 0.19 | 0.50± 0.17 | 0.40± 0.21 |
| nutt-2003-v2 | 0.77± 0.07 | 0.77± 0.21 | 0.87± 0.17 | 0.45± 0.27 |
| shipp-2002-v1 | 0.86± 0.03 | 0.84± 0.17 | 0.88± 0.16 | 0.73± 0.10 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.77± 0.12 | 0.84± 0.16 | 0.77± 0.05 |
| singh-2002 | 0.82± 0.04 | 0.74± 0.14 | 0.78± 0.12 | 0.69± 0.15 |
| Average Rank | **1.58** | 2.67 | 2.33 | 3.42 |

(a) ACCURACY RESULTS.

| Data Set | HEAD-DT | CART | C4.5 | REP |
|---|---|---|---|---|
| alizadeh-2000-v1 | 0.78± 0.09 | 0.67± 0.21 | 0.63± 0.26 | 0.72± 0.19 |
| alizadeh-2000-v2 | 0.91± 0.03 | 0.92± 0.10 | 0.84± 0.17 | 0.80± 0.19 |
| bhattacharjee-2001 | 0.89± 0.02 | 0.87± 0.12 | 0.89± 0.08 | 0.85± 0.07 |
| chen-2002 | 0.88± 0.03 | 0.85± 0.07 | 0.81± 0.07 | 0.78± 0.13 |
| golub-1999-v1 | 0.87± 0.02 | 0.86± 0.07 | 0.87± 0.08 | 0.90± 0.10 |
| lapointe-2004-v1 | 0.68± 0.07 | 0.73± 0.20 | 0.68± 0.15 | 0.50± 0.13 |
| liang-2005 | 0.87± 0.09 | 0.67± 0.21 | 0.77± 0.22 | 0.71± 0.23 |
| nutt-2003-v1 | 0.66± 0.09 | 0.48± 0.19 | 0.46± 0.16 | 0.31± 0.22 |
| nutt-2003-v2 | 0.77± 0.07 | 0.73± 0.26 | 0.87± 0.17 | 0.34± 0.31 |
| shipp-2002-v1 | 0.86± 0.03 | 0.79± 0.22 | 0.85± 0.20 | 0.62± 0.14 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.75± 0.13 | 0.83± 0.17 | 0.70± 0.10 |
| singh-2002 | 0.82± 0.04 | 0.72± 0.18 | 0.77± 0.13 | 0.67± 0.18 |
| Average Rank | **1.42** | 2.67 | 2.42 | 3.50 |

(b) F-MEASURE RESULTS.

difference in the relative frequency of the most frequent and the least frequent classes in the data set), it seems fair to say that HEAD-DT also outperformed the baseline methods in configuration {7 x 14}, given that it generated algorithms whose F-measure values are better than the values achieved by CART, C4.5, and REPTree.

Table VIII shows the results for configuration {9 x 12}. Results are consistent with the previous configurations, in which HEAD-DT has the edge over the baseline methods, presenting the lowest average rank for both accuracy and F-measure (1.58 and 1.42). C4.5 and CART presented similar ranks, whereas REPTree occupied the last position in the ranking.

Finally, Fig. 4 presents the fitness evolution in HEAD-DT across a full evolutionary cycle of 100 generations. We present both mean and best fitness of the population at each generation, for all experimental configurations. Some interesting observations can be extracted from Fig. 4. For instance, note that when the meta-training set is comprised of a single data set [see Fig. 4(a)], HEAD-DT is capable of continuously increasing the fitness function value, and at the same time the population is reasonably heterogeneous (mean fitness value oscillates considerably). In the other extreme, when the meta-training set is comprised of nine data sets [see Fig. 4(e)], HEAD-DT has a harder time in optimizing the fitness values, and the population is reasonably homogeneous (mean fitness value does not oscillate so much). The explanation for such a behavior is that by increasing the number of data sets in the meta-training set, HEAD-DT has to find algorithms with a good performance trade-off within the meta-training set. In practice, modifications in the design of the algorithm that favor a given data set may very well harm another, and hence, it is intuitively harder to design an algorithm that improves the performance of the generated decision trees in several data sets than in a single one. Conversely, it is also intuitive to believe that a larger meta-training set leads to the design of a better all-around algorithm, i.e., an algorithm that is robust to the peculiarities of the data sets from the application domain.

### G. Discussion

The experimental analysis conducted in the previous section aimed at comparing the algorithms designed by HEAD-DT with three baseline algorithms: CART, C4.5, and REPTree. We measured the predictive performance of each algorithm according to accuracy and F-measure, which are the most well known criteria for evaluating classification algorithms. In order to verify whether the number of data sets used in the meta-training set had an impact on the evolution of algorithms, we employed a consistent methodology that incrementally added two data sets, randomly chosen from the set of available data sets, to the meta-training set, generating five experimental configurations with different numbers of data sets in the meta-training and meta-test sets {#meta-

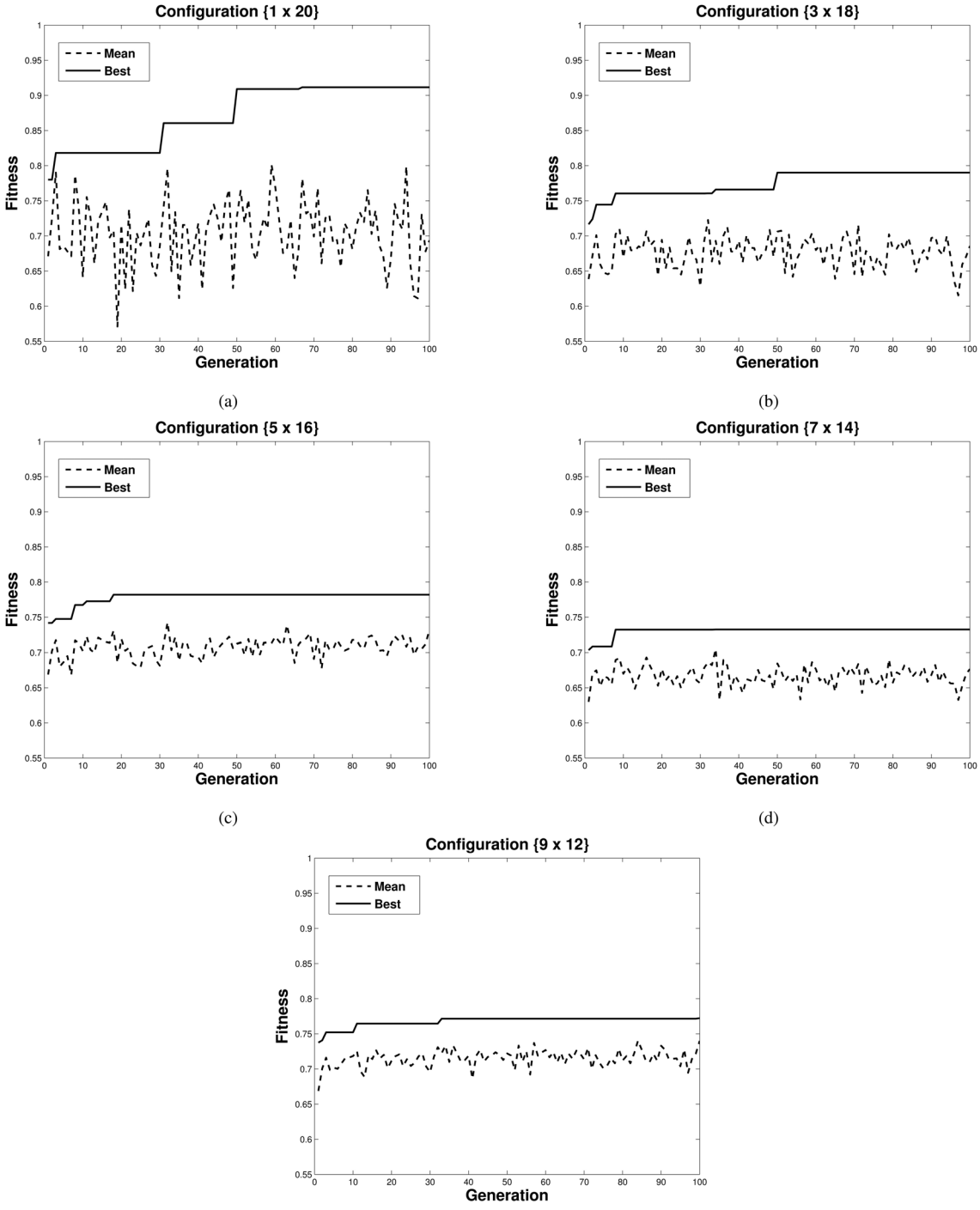



Fig. 4. Fitness evolution in HEAD-DT. Both the mean fitness and the best fitness of the population in a given generation are presented.

training sets, #meta-test sets}: {1 x 20}, {3 x 18}, {5 x 16}, {7 x 14}, and {9 x 12}. By analyzing the average rank obtained by each method in the previously mentioned configurations, we conclude the following.

1) HEAD-DT is consistently the best-performing method, nearly always presenting the lowest (best) average rank values among the four algorithms employed in the experimental analysis.
2) C4.5 and CART performances are quite similar, which is consistent with the fact that both algorithms are still the state-of-the-art top-down decision-tree induction algorithms.
3) REPTree, which is a variation of C4.5 that employs the reduced-error pruning strategy for simplifying the generated decision trees, is the worst-performing method of the lot. Its disappointing results seem to indicate that the reduced-error pruning strategy is not particularly suited to the gene-expression data sets, probably because it requires a separated validation set. Recall that the gene-expression data sets have very few instances when

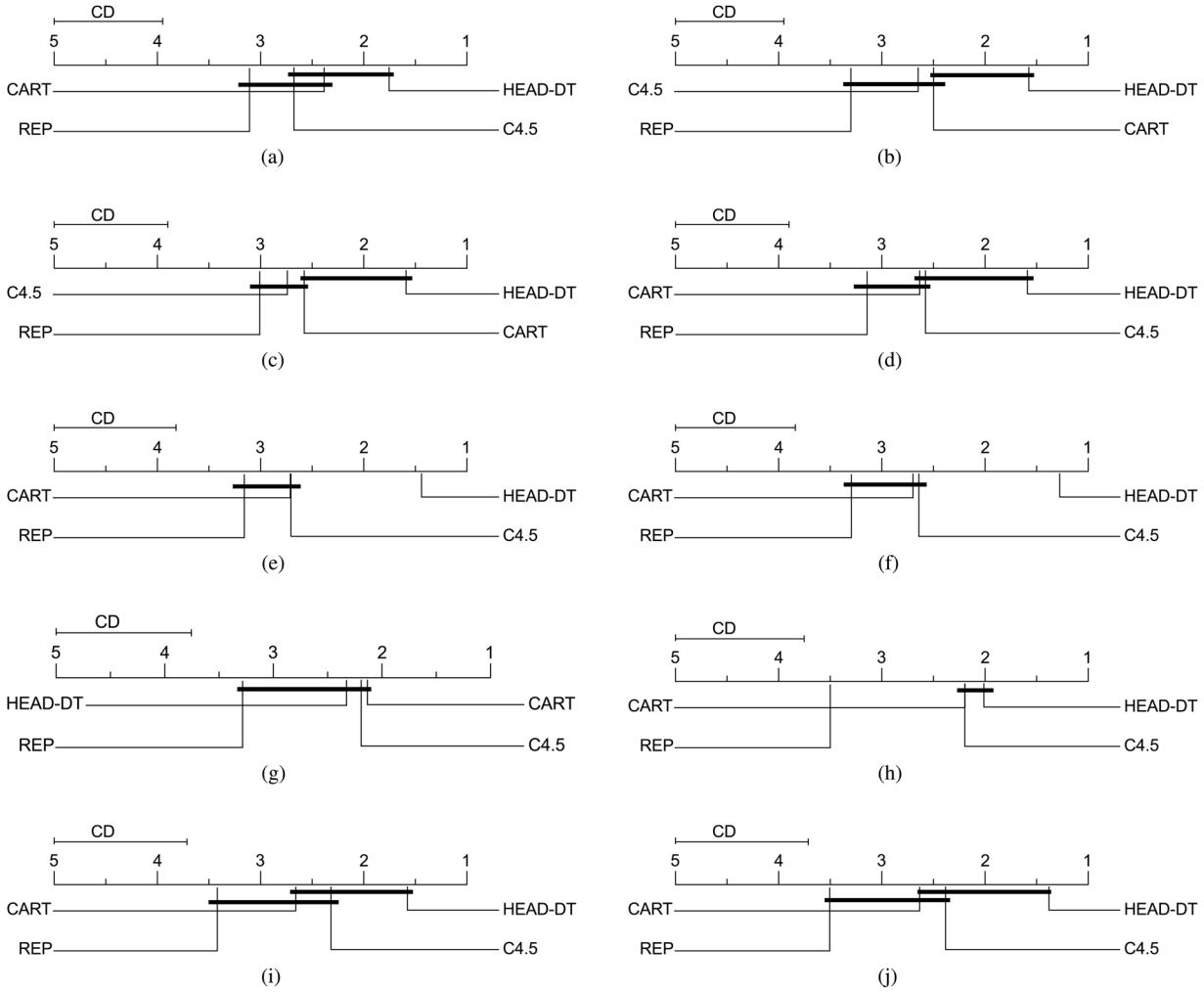



Fig. 5. Critical diagrams. (a) Accuracy rank for {1 x 20}. (b) F-measure rank for {1 x 20}. (c) Accuracy rank for {3 x 18}. (d) F-measure rank for {3 x 18}. (e) Accuracy rank for {5 x 16}. (f) F-measure rank for {5 x 16}. (g) Accuracy rank for {7 x 14}. (h) F-measure rank for {7 x 14}. (i) Accuracy rank for {9 x 12}. (j) F-measure rank for {9 x 12}.

compared to the currently very large databases from distinct application domains, and reducing their training set size for producing a validation set has certainly harmed REPTree's overall performance.

For summarizing the average rank values obtained by each method in every experimental configuration, we gathered the rank values from Tables IV–VIII in Table IX. Values in bold indicate the best performing method according to the corresponding evaluation measure. Note how consistently HEAD-DT presents the lowest average rank.

The next step of this empirical analysis is to verify whether the differences in rank values are statistically significant. For this particular analysis, we employ the graphical representation suggested in [51], the so-called critical diagrams. In this diagram, a horizontal line represents the axis on which we plot the average rank values of the methods. The axis is turned so that the lowest (best) ranks are to the right since we perceive the methods on the right side as better. When comparing all the algorithms against each other, we connect the groups of algorithms that are not significantly different through a bold horizontal line. We also show the critical difference given by the Nemenyi test above the graph.

Fig. 5 shows the critical diagrams for all experimental configurations. Note that HEAD-DT outperforms C4.5, CART, and REPTree with statistical significance in configuration {5 x 16} for both accuracy [Fig. 5(e)] and F-measure [Fig. 5(f)]. The only scenario in which there were no statistically significant differences among all methods was regarding the accuracy measure in configuration {7 x 14} [Fig. 5(g)]. The straightforward explanation for this case is that HEAD-DT optimizes its solutions according to the F-measure, even at the expense of accuracy. In the remaining scenarios, HEAD-DT always outperforms REPTree with statistical significance, which is not the case of CART and C4.5. In fact, CART and C4.5 are only able to outperform REPTree with statistical significance in configuration {7 x 14} [Fig. 5(h)], suggesting once again that HEAD-DT should be preferred over any of the baseline methods.

Considering the results of the four algorithms for all the ten combinations of experimental configurations and performance measures as a whole, as summarized in Table IX, the decision-tree algorithm designed by HEAD-DT obtained the best rank among the four algorithms in nine out of the ten rows of Table IX. Clearly, if the four algorithms had the same

TABLE IX
RESULTS OF THE EXPERIMENTAL ANALYSIS. HEAD-DT IS EXECUTED WITH $p = 0.6$. VALUES ARE THE AVERAGE PERFORMANCE (RANK) OF EACH ALGORITHM IN THE CORRESPONDING META-TEST SET, ACCORDING TO EITHER ACCURACY OR F-MEASURE. THE LOWER THE RANK, THE BETTER THE PERFORMANCE

| Configuration | Rank | HEAD-DT | CART | C4.5 | REPTree |
|---|---|---|---|---|---|
| {1 x 20} | Accuracy | **1.75** | 2.45 | 2.70 | 3.10 |
| | F-Measure | **1.55** | 2.50 | 2.60 | 3.35 |
| {3 x 18} | Accuracy | **1.61** | 2.61 | 2.72 | 3.05 |
| | F-Measure | **1.61** | 2.61 | 2.56 | 3.22 |
| {5 x 16} | Accuracy | **1.44** | 2.69 | 2.69 | 3.19 |
| | F-Measure | **1.31** | 2.69 | 2.63 | 3.38 |
| {7 x 14} | Accuracy | 2.36 | **2.14** | 2.21 | 3.29 |
| | F-Measure | **2.07** | 2.21 | 2.21 | 3.50 |
| {9 x 12} | Accuracy | **1.58** | 2.67 | 2.33 | 3.42 |
| | F-Measure | **1.42** | 2.67 | 2.42 | 3.50 |
| | Average | **1.67** | 2.52 | 2.51 | 3.30 |

predictive performance (so that each algorithm would have a 25% probability of being the winner), the probability that the algorithm designed by HEAD-DT would be the winner in nine out of ten cases would be extremely small, so we can be confident that the results obtained by HEAD-DT are statistically valid as a whole.

Finally, bearing in mind that when evaluating an algorithm only in terms of predictive performance (ignoring the interpretability of the classification model), support vector machines (SVM) are usually considered the state-of-the-art classification algorithm for gene expression data, we compare the best HEAD-DT configuration ({5 x 16}) with the SVM implementation of Weka toolkit [21]—the SMO algorithm with default parameters. Note that decision trees and SVMs are very different types of algorithms, and the goal of this comparison is not to directly evaluate the effectiveness of HEAD-DT as a method to generate decision-tree algorithms—that evaluation is better performed by comparing the decision-tree algorithms automatically designed by HEAD-DT with state-of-the-art manually -designed decision tree algorithms, as reported in Section III-H. Rather, we report here a comparison of the HEAD-DT results with SVM results in order to give an idea about the level of predictive performance that could be obtained for the gene expression data sets used in our experiments. Table X presents the accuracy and F-measure results obtained from this comparison. Note that, as expected, SVM has the edge over decision trees in the gene expression data sets in terms of predictive performance. SVM outperforms HEAD-DT's automatically designed algorithm in 13 (12) out of 16 data sets regarding accuracy (F-measure). Nevertheless, recall that the classification models built by SVM have the disadvantage of being a black box that can be hardly interpreted by users, and in the context of bioinformatics this tends to be an important disadvantage [3], by comparison with interpretable decision-tree models.

### H. General Algorithms Versus Specific Algorithms

In the experimental analysis presented in the previous sections, we evaluated for the first time the so-called general approach, in which HEAD-DT evolves a decision-tree induction algorithm from multiple data sets. In this section, we investigate whether the general approach [Fig. 3(b)] is more effective than the specific approach [Fig. 3(a)], in which HEAD-DT designs one algorithm per data set [17].

We employed the Wilcoxon signed-ranks test [52] for evaluating the statistical significance of the results. The Wilcoxon signed-ranks test is the recommended statistical test to evaluate two classifiers in multiple data sets [51].

Tables XI–XV present the comparison results of the five configurations of the general approach with the corresponding data sets in the specific approach, both regarding accuracy and F-measure values. Below each table, we present the number of victories for each method (ties are omitted), and also the $p$-value returned by the Wilcoxon test. For rejecting the null hypothesis of performance equivalency between the two algorithms, the $p$-values should be smaller than the desired significance level $\alpha$.

By careful inspection of Tables XI–XV, note that the general approach obtained a statistically significantly better predictive performance (at the 95% confidence level) than the specific approach in eight out of ten groups of results (five configurations times two predictive performance measures). The only case in which the general approach did not outperform the specific approach with statistical significance was configuration {7 x 14} (Table XIV), although it still presents a greater number of victories (8 x 5 regarding accuracy and 8 x 4 regarding F-measure). Recall that configuration {7 x 14} was the only configuration in which HEAD-DT did not have a clear advantage over the baseline algorithms (C4.5, CART, and REPTree).

We identified three data sets in which the specific approach outperforms the general approach across different configurations: armstrong-2002-v1, bittner-2000, and risinger-2003. This fact may indicate that these data sets have slightly different structural characteristics than the other data sets, which was not captured by the automatically designed algorithm.

The results presented in this section indicate that the general approach, when applied to data sets from the same application domain, has an advantage over designing specific algorithms

TABLE X
HEAD-DT: CONFIGURATION {5 X 16} VERSUS SVM. AVERAGE ± STANDARD DEVIATION

| Data Set | HEAD-DT | SVM |
|---|---|---|
| alizadeh-2000-v1 | 0.75± 0.05 | 0.90± 0.17 |
| alizadeh-2000-v2 | 0.91± 0.07 | 1.00± 0.00 |
| bhattacharjee-2001 | 0.94± 0.02 | 0.95± 0.07 |
| bittner-2000 | 0.54± 0.08 | 0.88± 0.21 |
| chen-2002 | 0.91± 0.02 | 0.96± 0.06 |
| chowdary-2006 | 0.97± 0.01 | 0.97± 0.05 |
| golub-1999-v1 | 0.88± 0.02 | 0.96± 0.07 |
| lapointe-2004-v1 | 0.71± 0.06 | 0.87± 0.14 |
| liang-2005 | 0.89± 0.08 | 0.98± 0.08 |
| nutt-2003-v1 | 0.61± 0.10 | 0.74± 0.19 |
| nutt-2003-v2 | 0.78± 0.08 | 0.92± 0.18 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.83± 0.15 |
| risinger-2003 | 0.55± 0.08 | 0.81± 0.18 |
| shipp-2002-v1 | 0.87± 0.03 | 0.93± 0.07 |
| singh-2002 | 0.78± 0.03 | 0.91± 0.14 |
| west-2001 | 0.92± 0.03 | 0.84± 0.16 |
| Number of victories | 2 | 13 |

(a) ACCURACY RESULTS

| Data Set | HEAD-DT | SVM |
|---|---|---|
| alizadeh-2000-v1 | 0.75± 0.05 | 0.88± 0.22 |
| alizadeh-2000-v2 | 0.91± 0.06 | 1.00± 0.00 |
| bhattacharjee-2001 | 0.94± 0.02 | 0.94± 0.08 |
| bittner-2000 | 0.53± 0.09 | 0.86± 0.25 |
| chen-2002 | 0.91± 0.02 | 0.96± 0.06 |
| chowdary-2006 | 0.97± 0.01 | 0.97± 0.05 |
| golub-1999-v1 | 0.88± 0.02 | 0.96± 0.07 |
| lapointe-2004-v1 | 0.69± 0.08 | 0.83± 0.18 |
| liang-2005 | 0.87± 0.10 | 0.98± 0.08 |
| nutt-2003-v1 | 0.60± 0.09 | 0.71± 0.21 |
| nutt-2003-v2 | 0.78± 0.08 | 0.90± 0.22 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.80± 0.18 |
| risinger-2003 | 0.53± 0.08 | 0.79± 0.20 |
| shipp-2002-v1 | 0.86± 0.03 | 0.93± 0.08 |
| singh-2002 | 0.78± 0.03 | 0.91± 0.14 |
| west-2001 | 0.92± 0.03 | 0.84± 0.16 |
| Number of victories | 2 | 12 |

(b) F- MEASURE RESULTS

TABLE XI
GENERAL APPROACH: {1 X 20} CONFIGURATION VERSUS SPECIFIC APPROACH. AVERAGE ± STANDARD DEVIATION

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.77± 0.11 | 0.72± 0.18 |
| alizadeh-2000-v2 | 0.88± 0.06 | 0.83± 0.14 |
| alizadeh-2000-v3 | 0.71± 0.05 | 0.66± 0.13 |
| armstrong-2002-v1 | 0.90± 0.04 | 0.91± 0.09 |
| bhattacharjee-2001 | 0.90± 0.03 | 0.89± 0.06 |
| bittner-2000 | 0.61± 0.09 | 0.64± 0.23 |
| chen-2002 | 0.85± 0.04 | 0.81± 0.10 |
| chowdary-2006 | 0.95± 0.03 | 0.96± 0.06 |
| golub-1999-v1 | 0.88± 0.03 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.66± 0.08 | 0.65± 0.17 |
| lapointe-2004-v2 | 0.62± 0.05 | 0.60± 0.12 |
| liang-2005 | 0.89± 0.09 | 0.76± 0.18 |
| nutt-2003-v1 | 0.53± 0.08 | 0.51± 0.25 |
| nutt-2003-v2 | 0.84± 0.08 | 0.59± 0.36 |
| pomeroy-2002-v1 | 0.88± 0.08 | 0.79± 0.18 |
| risinger-2003 | 0.58± 0.15 | 0.65± 0.17 |
| shipp-2002-v1 | 0.91± 0.05 | 0.77± 0.17 |
| singh-2002 | 0.78± 0.04 | 0.75± 0.15 |
| tomlins-2006 | 0.59± 0.07 | 0.62± 0.06 |
| west-2001 | 0.89± 0.08 | 0.83± 0.15 |
| Number of victories | 15 | 5 |
| Wilcoxon $p$-value: | 0.01069 | |

(a) ACCURACY RESULTS

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.77± 0.11 | 0.68± 0.23 |
| alizadeh-2000-v2 | 0.89± 0.06 | 0.80± 0.16 |
| alizadeh-2000-v3 | 0.71± 0.05 | 0.61± 0.16 |
| armstrong-2002-v1 | 0.90± 0.04 | 0.91± 0.09 |
| bhattacharjee-2001 | 0.89± 0.02 | 0.89± 0.06 |
| bittner-2000 | 0.61± 0.09 | 0.61± 0.25 |
| chen-2002 | 0.85± 0.04 | 0.81± 0.10 |
| chowdary-2006 | 0.95± 0.03 | 0.96± 0.06 |
| golub-1999-v1 | 0.88± 0.03 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.64± 0.10 | 0.60± 0.18 |
| lapointe-2004-v2 | 0.62± 0.05 | 0.57± 0.11 |
| liang-2005 | 0.87± 0.10 | 0.77± 0.20 |
| nutt-2003-v1 | 0.53± 0.08 | 0.42± 0.27 |
| nutt-2003-v2 | 0.84± 0.08 | 0.58± 0.38 |
| pomeroy-2002-v1 | 0.87± 0.10 | 0.77± 0.20 |
| risinger-2003 | 0.56± 0.15 | 0.61± 0.20 |
| shipp-2002-v1 | 0.90± 0.05 | 0.78± 0.16 |
| singh-2002 | 0.78± 0.04 | 0.74± 0.18 |
| tomlins-2006 | 0.58± 0.08 | 0.58± 0.09 |
| west-2001 | 0.89± 0.08 | 0.82± 0.16 |
| Number of victories | 14 | 3 |
| Wilcoxon $p$-value: | $4.83\times10^{-4}$ | |

(b) F- MEASURE RESULTS

per data set. Since it does not use any subset of the test data sets during evolution, it seems the general approach is less-prone to data overfitting. In addition, it has the clear advantage of designing a single algorithm that later can be applied to several data sets, avoiding multiple executions of the costly evolutionary cycle.

### *I. Algorithmic Time Complexity and the Cost-Effectiveness of Automated Versus Manual Algorithm Design*

The issue of time complexity of the proposed HEAD-DT can be analyzed from different perspectives, as follows. First, let us consider the conventional perspective for analyzing the time taken by a search algorithm. Regarding execution time, HEAD-DT is clearly slower than C4.5, CART, or REPTree. Considering that there are 100 individuals executed for 100 generations, there is a maximum (worst case) of 10 000 fitness evaluations of decision trees. We recorded the execution time of both breeding operations and fitness evaluation (one thread was used for breeding and other for evaluation). All experiments were executed in an Intel Xeon hexa-core E5645, 2.4 GHz, 48 GB RAM. Total time of breeding is absolutely negligible (a few milliseconds in a full evolutionary cycle), regardless of the data sets being used in the meta-training set, since breeding does not require access to the data sets. The bottleneck of HEAD-DT is fitness evaluation. The most time-consuming configuration, {9 x 12}, takes 11.62 h to be fully executed (one full evolutionary cycle of 100 generations). The faster configuration, {1 x 20}, takes only 5.60 min to be fully executed. Thus, the fitness evaluation time can vary quite a lot according to the number and type of data sets in the meta-training set. The computational complexity of algorithms such as C4.5 and CART is $O(m \times n \log n)$ ($m$ is the number of attributes and $n$ the data set size), plus a term regarding the specific pruning method. Considering that breeding takes negligible time, HEAD-DT time complexity is $O(i \times g \times m \times n \log n)$, where $i$ is the number of individuals

TABLE XII
GENERAL APPROACH: {3 X 18} CONFIGURATION VERSUS SPECIFIC APPROACH. AVERAGE ± STANDARD DEVIATION

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.74± 0.07 | 0.72± 0.18 |
| alizadeh-2000-v2 | 0.90± 0.07 | 0.83± 0.14 |
| armstrong-2002-v1 | 0.88± 0.01 | 0.91± 0.09 |
| bhattacharjee-2001 | 0.92± 0.03 | 0.89± 0.06 |
| bittner-2000 | 0.55± 0.09 | 0.64± 0.23 |
| chen-2002 | 0.87± 0.03 | 0.81± 0.10 |
| chowdary-2006 | 0.97± 0.01 | 0.96± 0.06 |
| golub-1999-v1 | 0.89± 0.02 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.70± 0.06 | 0.65± 0.17 |
| liang-2005 | 0.89± 0.08 | 0.76± 0.18 |
| nutt-2003-v1 | 0.67± 0.10 | 0.51± 0.25 |
| nutt-2003-v2 | 0.77± 0.08 | 0.59± 0.36 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.79± 0.18 |
| risinger-2003 | 0.53± 0.11 | 0.65± 0.17 |
| shipp-2002-v1 | 0.85± 0.05 | 0.77± 0.17 |
| singh-2002 | 0.80± 0.04 | 0.75± 0.15 |
| tomlins-2006 | 0.64± 0.03 | 0.62± 0.06 |
| west-2001 | 0.92± 0.04 | 0.83± 0.15 |
| Number of victories | 15 | 3 |
| Wilcoxon $p$-value: | 0.02081 | |

(a) ACCURACY RESULTS

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.73± 0.07 | 0.68± 0.23 |
| alizadeh-2000-v2 | 0.91± 0.06 | 0.80± 0.16 |
| armstrong-2002-v1 | 0.88± 0.02 | 0.91± 0.09 |
| bhattacharjee-2001 | 0.91± 0.03 | 0.89± 0.06 |
| bittner-2000 | 0.53± 0.10 | 0.61± 0.25 |
| chen-2002 | 0.87± 0.03 | 0.81± 0.10 |
| chowdary-2006 | 0.97± 0.01 | 0.96± 0.06 |
| golub-1999-v1 | 0.89± 0.02 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.68± 0.08 | 0.60± 0.18 |
| liang-2005 | 0.87± 0.10 | 0.77± 0.20 |
| nutt-2003-v1 | 0.65± 0.10 | 0.42± 0.27 |
| nutt-2003-v2 | 0.77± 0.08 | 0.58± 0.38 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.77± 0.20 |
| risinger-2003 | 0.52± 0.11 | 0.61± 0.20 |
| shipp-2002-v1 | 0.84± 0.05 | 0.78± 0.16 |
| singh-2002 | 0.80± 0.04 | 0.74± 0.18 |
| tomlins-2006 | 0.63± 0.03 | 0.58± 0.09 |
| west-2001 | 0.92± 0.04 | 0.82± 0.16 |
| Number of victories | 15 | 3 |
| Wilcoxon $p$-value: | $8.97\times10^{-3}$ | |

(b) F- MEASURE RESULTS

TABLE XIII
GENERAL APPROACH: {5 X 16} CONFIGURATION VERSUS SPECIFIC APPROACH. AVERAGE ± STANDARD DEVIATION

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.75± 0.05 | 0.72± 0.18 |
| alizadeh-2000-v2 | 0.91± 0.07 | 0.83± 0.14 |
| bhattacharjee-2001 | 0.94± 0.02 | 0.89± 0.06 |
| bittner-2000 | 0.54± 0.08 | 0.64± 0.23 |
| chen-2002 | 0.91± 0.02 | 0.81± 0.10 |
| chowdary-2006 | 0.97± 0.01 | 0.96± 0.06 |
| golub-1999-v1 | 0.88± 0.02 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.71± 0.06 | 0.65± 0.17 |
| liang-2005 | 0.89± 0.08 | 0.76± 0.18 |
| nutt-2003-v1 | 0.61± 0.10 | 0.51± 0.25 |
| nutt-2003-v2 | 0.78± 0.08 | 0.59± 0.36 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.79± 0.18 |
| risinger-2003 | 0.55± 0.08 | 0.65± 0.17 |
| shipp-2002-v1 | 0.87± 0.03 | 0.77± 0.17 |
| singh-2002 | 0.78± 0.03 | 0.75± 0.15 |
| west-2001 | 0.92± 0.03 | 0.83± 0.15 |
| Number of victories | 14 | 2 |
| Wilcoxon $p$-value: | 0.01309 | |

(a) ACCURACY RESULTS

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.75± 0.05 | 0.68± 0.23 |
| alizadeh-2000-v2 | 0.91± 0.06 | 0.80± 0.16 |
| bhattacharjee-2001 | 0.94± 0.02 | 0.89± 0.06 |
| bittner-2000 | 0.53± 0.09 | 0.61± 0.25 |
| chen-2002 | 0.91± 0.02 | 0.81± 0.10 |
| chowdary-2006 | 0.97± 0.01 | 0.96± 0.06 |
| golub-1999-v1 | 0.88± 0.02 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.69± 0.08 | 0.60± 0.18 |
| liang-2005 | 0.87± 0.10 | 0.77± 0.20 |
| nutt-2003-v1 | 0.60± 0.09 | 0.42± 0.27 |
| nutt-2003-v2 | 0.78± 0.08 | 0.58± 0.38 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.77± 0.20 |
| risinger-2003 | 0.53± 0.08 | 0.61± 0.20 |
| shipp-2002-v1 | 0.86± 0.03 | 0.78± 0.16 |
| singh-2002 | 0.78± 0.03 | 0.74± 0.18 |
| west-2001 | 0.92± 0.03 | 0.82± 0.16 |
| Number of victories | 14 | 2 |
| Wilcoxon $p$-value: | $4.18\times10^{-3}$ | |

(b) F- MEASURE RESULTS

and $g$ is the number of generations. In practice, the number of evaluations is much smaller than $i \times g$, since repeated individuals are not reevaluated. In addition, individuals selected by elitism and reproduction (instead of crossover) are also not reevaluated, saving computational time.

It should be noted, however, that the aforementioned conventional perspective for analyzing HEAD-DT is misleading in one way: it assumes that HEAD-DT is a conventional search algorithm, searching for an optimal solution to a single data set, which is not the case. In reality, as discussed earlier, HEAD-DT is a hyper-heuristic that outputs a complete decision-tree induction algorithm. The decision-tree algorithm automatically designed by HEAD-DT, as well as the manually designed decision tree algorithms C4.5, CART, and REPTree, are all complete decision-tree algorithms that can be reused over and over again to extract knowledge from different data sets. It seems reasonable to assume that the time taken by a single human researcher to design and implement a new decision-tree induction algorithm is on the order of at least several months. In this context, since HEAD-DT is effectively replacing the manual design of decision-tree algorithms with an automated approach for such a design, even if HEAD-DT took a couple of days to produce a decision-tree algorithm, that time would still be much smaller than the corresponding manual design time. That is, HEADT-DT is a much faster algorithm-design method than the manual design of decision-tree algorithms.

Playing the role of devil's advocate, one could perhaps present the following counter-argument: the previous discussion ignores the fact that HEAD-DT was itself designed by human researchers, a design process that also took several months! This is true, but even taking this into account, it can be argued that HEAD-DT is still much more cost-effective than the human design of decision-tree algorithms, as follows. First, now that HEAD-DT has been manually designed, it can be reused over and over again to automatically create

decision-tree algorithms tailored to any particular type of data set (or application domain) in which a given user is interested. In this paper, we focused on gene expression data sets, but HEAD-DT can be reused to create decision-tree algorithms tailored to, say, a specific type of financial data set or a specific type of medical data set, to mention just a few examples of application domains. Once HEAD-DT has been created, the cost associated with using HEAD-DT in any other application domain is very small, it is essentially the cost associated with the time to run HEAD-DT in a new application domain (say, a couple of days of a desktop computer's processing time).

In contrast, what would be the cost of manually creating a new decision-tree algorithm tailored to a particular type of data set or application domain? First, note that this kind of manual design is hardly found in the literature. This is presumably because few human researchers are experts on both decision-tree algorithms and the target application domain. Just for the sake of argument, though, let us make the (unrealistic) assumption that there are many application domains for which there is a researcher who is an expert in both that application domain and decision-tree induction algorithms. For each such application domain, it seems safe to assume again that the human expert in question would need on the order of several months to design a new decision-tree algorithm that is effective and really tailored to that application domain. In contrast, HEAD-DT could perform that algorithm design automatically in a couple of days.

In summary, given the very large diversity of application domains to which decision-tree algorithms can be applied, HEAD-DT's automated approach offers a much more cost-effective approach for designing decision-tree algorithms than the conventional manual design approach that is nearly always used in machine learning research. In this sense, HEAD-DT paves the way for the large-scale and cost-effective production of decision-tree algorithms that are tailored to any specific application domain or type of classification data set in which any given user is interested.

### *J. Example of an Evolved Algorithm*

For illustrating a novel algorithm designed by HEAD-DT, let us consider the {5 x 16} configuration, in which HEAD-DT managed to significantly outperform C4.5, CART, and REP-Tree for both accuracy and F-measure. The algorithm designed by HEAD-DT is presented in Fig. 6. It is indeed novel, since no algorithm in the literature combines components such as the DCSM criterion with PEP pruning. The main advantage of HEAD-DT is that it automatically searches for the most suitable combination of algorithmic components (with their own biases) that is tailored to the data set being investigated. It is hard to believe that a researcher would combine such a distinct set of components such as those in Fig. 6 to obtain better performance than traditional algorithms such as C4.5 and CART in the specific application domain of gene expression data sets.

The algorithm presented in Fig. 6 is one of the 25 algorithms automatically designed by HEAD-DT in the experimental analysis (five configurations executed five times each). Nevertheless, by close inspection of the 25 automatically generated

```
1) Recursively split nodes using the DCSM criterion;
2) Aggregate nominal splits in binary subsets;
3) Perform step 1 until class-homogeneity or the minimum
   number of 6 instances is reached;
4) Perform PEP pruning with 2 standard errors (SEs)
   to adjust the training error;
When dealing with missing values:
1) Calculate the split of missing values by performing
   unsupervised imputation;
2) Distribute missing values by assigning the instance to
   the largest partition;
3) For classifying an instance with missing values,
   explore all branches and combine the results.
```

Fig. 6. Algorithm designed by HEAD-DT for the {5 x 16} configuration of data sets.

algorithms, we observed that the algorithm in Fig. 6 is comprised of building blocks that were consistently favored regarding the gene expression application domain. For instance, the DCSM and Chandra-Varghese criteria—both created recently by the same authors [29], [30]—were selected as the best split criterion in 44% (11/25) of the algorithms designed by HEAD-DT. Similarly, the minimum number of instances stop criterion was selected in 68% (17/25) of the algorithms, with either six or seven instances as its parameter value. Finally, the PEP pruning was the favored pruning strategy in 56% (14/25) of the algorithms, with a very large advantage over the default strategies used by C4.5 (EBP pruning, selected in 8% (2/25) of the algorithms) and CART (CCP pruning, not selected by any of the automatically designed algorithms). This is evidence that the best components of decision-tree induction algorithms customized to microarray data sets are quite different from the default components of the traditional decision-tree algorithms, which justifies the idea of automatically designing a decision-tree algorithm for microarray data sets.

## IV. Related Work

To the best of our knowledge, no work to date attempts to automatically design complete decision-tree induction algorithms, with the exception of our previous conference paper [17], which has been significantly extended in this current paper (as discussed in the introduction), and of the biology-focused work in [53].

In this paper, we presented for the first time the novel strategy of designing a decision-tree algorithm from multiple data sets, and then apply the evolved algorithm in other data sets, though from the same type (the data set type in which a given user is interested), unlike our previous work in [17], where generalization across data sets was not needed. In addition to this important difference, this paper also extends [17] in three ways.

1) It performs a more extensive set of computational experiments to evaluate HEAD-DT.
2) It extends the time complexity analysis of HEAD-DT with a discussion on the cost-effectiveness of automated versus manual algorithm design.
3) It reports the decision-tree components that were most frequently selected by HEAD-DT in order to create decision-tree algorithms customized to microarray data sets. Furthermore, our work in [53] is focused on the

TABLE XIV
GENERAL APPROACH: {7 X 14} CONFIGURATION VERSUS SPECIFIC APPROACH. AVERAGE ± STANDARD DEVIATION

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.79± 0.09 | 0.72± 0.18 |
| alizadeh-2000-v2 | 0.89± 0.05 | 0.83± 0.14 |
| bhattacharjee-2001 | 0.89± 0.03 | 0.89± 0.06 |
| chen-2002 | 0.84± 0.02 | 0.81± 0.10 |
| chowdary-2006 | 0.91± 0.04 | 0.96± 0.06 |
| golub-1999-v1 | 0.84± 0.03 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.61± 0.09 | 0.65± 0.17 |
| liang-2005 | 0.87± 0.08 | 0.76± 0.18 |
| nutt-2003-v1 | 0.68± 0.07 | 0.51± 0.25 |
| nutt-2003-v2 | 0.76± 0.07 | 0.59± 0.36 |
| pomeroy-2002-v1 | 0.70± 0.12 | 0.79± 0.18 |
| shipp-2002-v1 | 0.86± 0.03 | 0.77± 0.17 |
| singh-2002 | 0.79± 0.02 | 0.75± 0.15 |
| west-2001 | 0.80± 0.07 | 0.83± 0.15 |
| Number of victories | 8 | 5 |
| Wilcoxon $p$-value: | 0.1531 | |

(a) ACCURACY RESULTS

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.79± 0.09 | 0.68± 0.23 |
| alizadeh-2000-v2 | 0.91± 0.03 | 0.80± 0.16 |
| bhattacharjee-2001 | 0.89± 0.03 | 0.89± 0.06 |
| chen-2002 | 0.84± 0.02 | 0.81± 0.10 |
| chowdary-2006 | 0.91± 0.04 | 0.96± 0.06 |
| golub-1999-v1 | 0.83± 0.04 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.60± 0.11 | 0.60± 0.18 |
| liang-2005 | 0.87± 0.08 | 0.77± 0.20 |
| nutt-2003-v1 | 0.65± 0.08 | 0.42± 0.27 |
| nutt-2003-v2 | 0.76± 0.07 | 0.58± 0.38 |
| pomeroy-2002-v1 | 0.69± 0.13 | 0.77± 0.20 |
| shipp-2002-v1 | 0.86± 0.03 | 0.78± 0.16 |
| singh-2002 | 0.79± 0.02 | 0.74± 0.18 |
| west-2001 | 0.80± 0.07 | 0.82± 0.16 |
| Number of victories | 8 | 4 |
| Wilcoxon $p$-value: | 0.06763 | |

(b) F- MEASURE RESULTS

TABLE XV
GENERAL APPROACH: {9 X 12} CONFIGURATION VERSUS SPECIFIC APPROACH. AVERAGE ± STANDARD DEVIATION

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.79± 0.09 | 0.72± 0.18 |
| alizadeh-2000-v2 | 0.89± 0.05 | 0.83± 0.14 |
| bhattacharjee-2001 | 0.90± 0.02 | 0.89± 0.06 |
| chen-2002 | 0.88± 0.03 | 0.81± 0.10 |
| golub-1999-v1 | 0.87± 0.02 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.70± 0.05 | 0.65± 0.17 |
| liang-2005 | 0.88± 0.08 | 0.76± 0.18 |
| nutt-2003-v1 | 0.67± 0.09 | 0.51± 0.25 |
| nutt-2003-v2 | 0.77± 0.07 | 0.59± 0.36 |
| shipp-2002-v1 | 0.86± 0.03 | 0.79± 0.18 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.77± 0.17 |
| singh-2002 | 0.82± 0.04 | 0.75± 0.15 |
| Number of victories | 12 | 0 |
| Wilcoxon $p$-value: | $4.88\times10^{-4}$ | |

(a) ACCURACY RESULTS

| Data Set | General | Specific |
|---|---|---|
| alizadeh-2000-v1 | 0.78± 0.09 | 0.68± 0.23 |
| alizadeh-2000-v2 | 0.91± 0.03 | 0.80± 0.16 |
| bhattacharjee-2001 | 0.89± 0.02 | 0.89± 0.06 |
| chen-2002 | 0.88± 0.03 | 0.81± 0.10 |
| golub-1999-v1 | 0.87± 0.02 | 0.86± 0.10 |
| lapointe-2004-v1 | 0.68± 0.07 | 0.60± 0.18 |
| liang-2005 | 0.87± 0.09 | 0.77± 0.20 |
| nutt-2003-v1 | 0.66± 0.09 | 0.42± 0.27 |
| nutt-2003-v2 | 0.77± 0.07 | 0.58± 0.38 |
| shipp-2002-v1 | 0.86± 0.03 | 0.77± 0.20 |
| pomeroy-2002-v1 | 0.92± 0.05 | 0.78± 0.16 |
| singh-2002 | 0.82± 0.04 | 0.74± 0.18 |
| Number of victories | 11 | 0 |
| Wilcoxon $p$-value: | $4.88\times10^{-4}$ | |

(b) F- MEASURE RESULTS

practical consequences of knowledge discovered by a HEAD-DT algorithm that was tailored to individual data sets from a particular application domain in biology (flexible-receptor molecular docking data), while the work described in this paper is focused on computer science issues.

The most related approach to this paper by other authors is hyper-heuristic decision tree [54]. It proposes an EA for evolving heuristic rules in order to determine the best splitting criterion to be used in nonterminal nodes of a decision tree. While this approach is a first step toward automating the design of decision-tree induction algorithms, it evolves a single component of the algorithm (the choice of splitting criterion), and thus should be further extended in order to be able to generate complete decision-tree induction algorithms.

Finally, a somewhat related approach is the one presented in [55]. Delibasic *et al.* [55] proposed a framework for combining decision-tree components, and test 80 different combinations of design components on 15 benchmark data sets. This approach is not a hyper-heuristic, since it does not present a heuristic to choose among different heuristics. It simply selects a fixed number of component combinations and tests them all against traditional decision-tree algorithms. We believe our strategy is more robust, since by using an evolutionary algorithm, we can search for solutions in a much larger search space. Currently, HEAD-DT's search space consists of more than 127 million different decision-tree induction algorithms.

## V. CONCLUSION

In this paper, we presented HEAD-DT, a hyper-heuristic algorithm that automatically designs top-down decision-tree induction algorithms. These algorithms have been manually improved over the last 40 years, resulting in a great number of approaches for each of their design components. Since the human manual approach for testing all possible modifications in the design components of decision-tree algorithms would be unfeasible, we believe the evolutionary search of HEAD-DT constitutes a robust and effective solution to that problem.

We performed a thorough experimental analysis in which the algorithms automatically designed by HEAD-DT were compared with the state-of-the-art decision-tree induction algorithms CART [20] and C4.5 [19], and also to a variation of C4.5 called REPTree [21], in 35 real-world microarray

gene expression data sets. We assessed the performance of HEAD-DT through two different predictive performance measures, accuracy, and F-measure. The experimental analysis has shown that HEAD-DT automatically generated decision-tree induction algorithms that outperformed the three (manually designed) baseline methods in nine out of ten experiments. In terms of statistical significance, HEAD-DT obtained significantly higher predictive performance than all three baseline methods in two experiments, while HEAD-DT never obtained a significantly lower predictive accuracy than any of the three baseline methods in any experiment. Furthermore, HEAD-DT's automated approach can be considered a much more cost-effective approach to the design of decision-tree algorithms tailored to a given type of data set (or application domain) than the manual algorithm design approach currently used in machine learning, as discussed earlier. Hence, HEAD-DT seems to arise as an effective hyper-heuristic method for future applications of decision trees.

As future work, we plan to develop a more sophisticated search system such as grammar-based genetic programming, and investigate whether it can outperform our current HEAD-DT implementation.

## REFERENCES


[1] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Reading, MA, USA: Addison-Wesley, 2005.

[2] V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman, "Decision trees: An overview and their use in medicine," *J. Med. Syst.*, vol. 26, no. 5, pp. 445–463, Oct. 2002.

[3] A. A. Freitas, D. C. Wieser, and R. Apweiler, "On the importance of comprehensible classification models for protein function prediction," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 7, no. 1, pp. 172–182, Jan. 2010.

[4] L. Hyafil and R. Rivest, "Constructing optimal binary decision trees is NP-complete," *Inform. Process. Lett.*, vol. 5, no. 1, pp. 15–17, 1976.

[5] T. Hancock, T. Jiang, M. Li, and J. Tromp, "Lower bounds on learning decision lists and trees," *Inform. Comput.*, vol. 126, no. 2, pp. 114–122, May 1996.

[6] H. Zantema and H. Bodlaender, "Finding small equivalent decision trees is hard," *Int. J. Found. Comput. Sci.*, vol. 11, no. 2, pp. 343–354, 2000.

[7] G. E. Naumov, "NP-completeness of problems of construction of optimal decision trees," *Sov. Phys. Doklady*, vol. 36, no. 4, pp. 270–271, 1991.

[8] R. C. Barros, R. Cerri, P. A. Jaskowiak, and A. C. P. L. F. de Carvalho, "A bottom-up oblique decision tree induction algorithm," in *Proc. 11th Int. Conf. Intell. Syst. Design Appl.*, 2011, pp. 450–456.

[9] B. Kim and D. Landgrebe, "Hierarchical classifier design in high-dimensional numerous class cases," *IEEE Trans. Geosci. Remote Sens.*, vol. 29, no. 4, pp. 518–528, Jul. 1991.

[10] M. Basgalupp, R. C. Barros, A. de Carvalho, A. Freitas, and D. Ruiz, "LEGAL-tree: A lexicographic multiobjective genetic algorithm for decision tree induction," in *Proc. 24th ACM SAC*, 2009, pp. 1085–1090.

[11] M. P. Basgalupp, A. C. P. L. F. de Carvalho, R. C. Barros, D. D. Ruiz, and A. A. Freitas, "Lexicographic multiobjective evolutionary induction of decision trees," *Int. J. Bioinspired Comput.*, vol. 1, nos. 1–2, pp. 105–117, 2009.

[12] R. C. Barros, M. Basgalupp, D. Ruiz, A. de Carvalho, and A. A. Freitas, "Evolutionary model tree induction," in *Proc. 25th ACM SAC*, 2010, pp. 1131–1137.

[13] R. C. Barros, D. D. Ruiz, and M. P. Basgalupp, "Evolutionary model trees for handling continuous classes in machine learning," *Inform. Sci.*, vol. 181, no. 5, pp. 954–971, Mar. 2011.

[14] L. Breiman, "Random forests," *Mach. Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] L. Breslow and D. Aha, "Simplifying decision trees: A survey," *Knowl. Eng. Rev.*, vol. 12, no. 01, pp. 1–40, 1997.

[16] F. Esposito, D. Malerba, and G. Semeraro, "A comparative analysis of methods for pruning decision trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 476–491, May 1997.

[17] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, "A hyper-heuristic evolutionary algorithm for automatically designing decision-tree algorithms," in *Proc. ACM GECCO*, 2012, pp. 1237–1244.

[18] M. Souto, I. Costa, D. de Araujo, T. Ludermir, and A. Schliep, "Clustering cancer gene expression data: A comparative study," *BMC Bioinformatics*, vol. 9, no. 1, p. 497, 2008.

[19] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth, 1984.

[21] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Mateo, CA, USA: Morgan Kaufmann, Oct. 1999.

[22] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "A classification of hyper-heuristics approaches," in *Handbook Metaheuristics* (Int. Series Oper. Res. Manage. Sci.), vol. 57, 2nd ed., M. Gendreau and J.-Y. Potvin, Eds. Berlin, Germany: Springer, 2010, ch. 15, pp. 449–468.

[23] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision tree induction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 291–312, May 2012.

[24] J. R. Quinlan, "Induction of decision trees," *Mach. Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[25] M. Gleser and M. Collen, "Toward automated medical decisions," *Comput. Biomed. Res.*, vol. 5, no. 2, pp. 180–189, 1972.

[26] J. Mingers, "Expert systems: Rule induction with statistical data," *J. Oper. Res. Soc.*, vol. 38, no. 1, pp. 39–47, 1987.

[27] R. L. De Mántaras, "A distance-based attribute selection measure for decision tree induction," *Mach. Learning*, vol. 6, no. 1, pp. 81–92, 1991.

[28] J. Martin, "An exact probability metric for decision tree splitting and stopping," *Mach. Learning*, vol. 28, no. 2, pp. 257–291, 1997.

[29] B. Chandra and P. P. Varghese, "Moving toward efficient decision tree construction," *Inform. Sci.*, vol. 179, no. 8, pp. 1059–1069, 2009.

[30] B. Chandra, R. Kothari, and P. Paul, "A new node splitting measure for decision tree construction," *Pattern Recognit.*, vol. 43, no. 8, pp. 2725–2731, 2010.

[31] J. Mingers, "An empirical comparison of selection measures for decision-tree induction," *Mach. Learning*, vol. 3, no. 4, pp. 319–342, 1989.

[32] P. C. Taylor and B. W. Silverman, "Block diagrams and splitting criteria for classification trees," *Statist. Comput.*, vol. 3, no. 4, pp. 147–161, Dec. 1993.

[33] B. Jun, C. Kim, Y.-Y. Song, and J. Kim, "A new criterion in selection and discretization of attributes for the generation of decision trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 1371–1375, Feb. 1997.

[34] U. Fayyad and K. Irani, "The attribute selection problem in decision tree generation," in *Proc. Nat. Conf. Artif. Intell.*, 1992, pp. 104–110.

[35] J. Ching, A. Wong, and K. Chan, "Class-dependent discretization for inductive learning from continuous and mixed-mode data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 7, pp. 641–651, Jul. 1995.

[36] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 1, pp. 379–423, 625–656, 1948.

[37] J. H. Friedman, "A recursive partitioning decision rule for nonparametric classification," *IEEE Trans. Comput.*, vol. TC-100, no. 4, pp. 404–408, Apr. 1977.

[38] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach. Learning*, vol. 3, no. 4, pp. 261–283, 1989.

[39] W. Loh and Y. Shih, "Split selection methods for classification trees," *Statist. Sinica*, vol. 7, pp. 815–840, 1997.

[40] J. R. Quinlan, "Unknown attribute values in induction," in *Proc. 6th Int. Workshop Mach. Learning*, 1989, pp. 164–168.

[41] I. Kononenko, I. Bratko, and E. Roskar, "Experiments in automatic learning of medical diagnostic rules," in *Proc. Intern. School Synthesis Expert's Knowl. Workshop*, Bled, Slovenia, Jozef Stefan Instit., Ljubljana, Yugoslavia, Aug. 1984.

[42] J. R. Quinlan, "Decision trees as probabilistic classifiers," in *Proc. 4th Int Mach Learn Work*, 1987.

[43] J. R. Quinlan, "Simplifying decision trees," *Int. J. Man-Mach. Stud.*, vol. 27, no. 3, pp. 221–234, 1987.

[44] T. Niblett and I. Bratko, "Learning decision rules in noisy domains," in *Proc. 6th Annu. Tech. Conf. Expert Syst.*, 1986, pp. 25–34.

[45] B. Cestnik and I. Bratko, "On estimating probabilities in tree pruning," in *Proc. EWSL*, 1991, pp. 138–150.

[46] G. L. Pappa, "Automatically evolving rule induction algorithms with grammar-based genetic programming," Ph.D. dissertation, School of Computing, Univ. Kent at Canterbury, Canterbury, U.K., 2007.
[47] T. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE Trans. Pattern Anal.*, vol. 24, no. 3, pp. 289–300, Mar. 2002.
[48] T. Ho, M. Basu, and M. Law, "Measures of geometrical complexity in classification problems," in *Data Complexity in Pattern Recognition*. London, U.K.: Springer, 2006.
[49] A. A. Freitas, "A critical review of multiobjective optimization in data mining: A position paper," *SIGKDD Explor. Newsl.*, vol. 6, no. 2, pp. 77–86, 2004.
[50] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Mach. Learn.*, vol. 52, nos. 1–2, pp. 91–118, Jul. 2003.
[51] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learning Res.*, vol. 7, pp. 1–30, Dec. 2006.
[52] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, no. 6, pp. 80–83, Dec. 1945.
[53] R. C. Barros, A. T. Winck, K. S. Machado, M. P. Basgalupp, A. de Carvalho, D. D. Ruiz, *et al.*, "Automatic design of decision-tree induction algorithms tailored to flexible-receptor docking data," *BMC Bioinformatics*, vol. 13, no. 1, p. 310, 2012.
[54] A. Vella, D. Corne, and C. Murphy, "Hyper-heuristic decision tree induction," in *Proc. World Congr. Nature Biologically Inspired Comput.*, 2009, pp. 409–414.
[55] B. Delibasic, M. Jovanovic, M. Vukicevic, M. Suknovic, and Z. Obradovic, "Component-based decision trees for classification," *Intell. Data Anal.*, vol. 15, pp. 1–38, Aug. 2011.

**Rodrigo C. Barros** received the B.Sc. degree from Universidade Federal de Pelotas, Pelotas, Brazil, the M.Sc. degree from Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil, and the Ph.D. degree from Universidade de São Paulo, São Carlos, Brazil, all in computer science, in 2007, 2009, and 2013, respectively.

He is an assistant professor in the Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, where he leads the business intelligence and machine learning research group (GPIN).

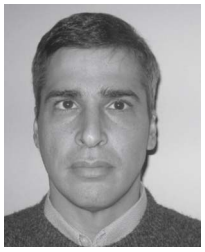
His research interests are machine learning, data mining, knowledge discovery, and biologically inspired computational intelligence algorithms.

**Márcio P. Basgalupp** received the B.Sc. degree in computer science from the Federal University of Pelotas, Pelotas, Brazil, in 2005, the M.Sc. degree in computer science from the Pontifical Catholic University of Rio Grande do Sul, Rio Grande do Sul, Brazil, in 2007, and the Ph.D. degree in computer science from the University of São Paulo, São Paulo, Brazil, in 2010.

He is currently an Associate Professor with the Department of Computer Sciences, Universidade Federal de São Paulo, São Paulo, Brazil. In 2010, he was a Post-Doctoral Researcher with the Norwegian University of Science and Technology, Trondheim, Norway, where he focused on biomedical data mining. His research interests include machine learning, data mining, and bioinspired computation.

**Alex A. Freitas** received the B.Sc. degree in computer science from the Faculdade de Tecnologia de São Paulo, São Paulo, Brazil, in 1989, the M.Sc. degree in computer science from the Federal University of São Carlos, São Carlos, Brazil, 1993, the Ph.D. degree in computer science from the University of Essex, Essex, U.K., in 1997, and the M.Phil. (the master's degree) in biological sciences from the University of Liverpool, Liverpool, U.K., in 2011.

He is currently a Professor of computational intelligence with the Computer Science Department, University of Kent, Kent, U.K. He has co-authored three research-oriented books on data mining and over 150 peer-reviewed papers in journals and conference proceedings. His research interests include data mining and machine learning (particularly classification) and their applications to biology (particularly the biology of aging) and pharmaceutical sciences.



**André C. P. L. F. de Carvalho** received the B.Sc. and M.Sc. degrees in computer science from the Universidade Federal de Pernambuco, Recife, Brazil, and the Ph.D. degree in electronic engineering from the University of Kent, Kent, U.K.

He is currently a Full Professor with the Department of Computer Science, University of São Paulo, São Paulo, Brazil. He has published around 100 journal and 250 conference refereed papers. His research interests include machine learning, data mining, bioinformatics, evolutionary computation, bioinspired computing, and hybrid intelligent systems.