Evolving Decision-Tree Induction Algorithms with a Multi-Objective Hyper-Heuristic

Márcio P. Basgalupp
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos–SP, Brazil
basgalupp@unifesp.br

Rodrigo C. Barros
Faculdade de Informática
Pontifícia Universidade Católica do RS
Porto Alegre–RS, Brazil
rodrigo.barros@pucrs.br

Vili Podgorelec
Institute of Informatics, FERI
University of Maribor
Maribor, Slovenia
vili.podgorelec@um.si

ABSTRACT

Multi-objective optimization has been widely used in evolutionary computation for solving problems in which two or more conflicting objectives need to be optimized in a simultaneous fashion. This paper presents a multi-objective hyper-heuristic based on evolutionary algorithms that automatically designs complete decision-tree induction algorithms. Such algorithms are designed to generate decision trees that present an interesting trade-off between predictive performance and complexity. The proposed approach is tested over 20 UCI datasets, and it is compared with a single-objective hyper-heuristic as well as with traditional decision-tree induction algorithms. Experimental results show that the proposed approach can match the top predictive performance achieved by the baseline methods, without significant loss in model comprehensibility.

Categories and Subject Descriptors

I.2.6 [**Induction and Knowledge Acquisition**]: [Learning]

Keywords

Hyper-Heuristics, Multi-Objective Optimization, Machine Learning

1. INTRODUCTION

Evolutionary algorithms (EAs) have been extensively applied to supervised learning in the past decades, specially for the problem of data classification. Classification can be regarded as an optimization problem, where the goal is to find a function $\hat{f}$ that better approximates the unknown true function f responsible for mapping attribute values from the input space into discrete categories (labels), $f : \mathbf{X} \rightarrow Y$.

Among the several existing classification methods, decision-tree induction algorithms are widely employed by both researchers and practitioners, mainly due to the fact that decision trees are comprehensible classification models. Indeed, decision trees are directed acyclic graphs that can be easily read as a disjunction of conjunctions in the form of `if-then` classification rules. They are the preferred model in application domains in which understanding the reasons that lead to a certain prediction is as important as the prediction itself (e.g., medical diagnosis [19] and protein function prediction [15]).

Most decision-tree induction algorithms employ the top-down greedy strategy for building the trees (e.g., C4.5 [20] and CART [10]). Nonetheless, the top-down approach is prone to falling into local-optima since it locally optimizes a criterion for each node of the tree in a recursive fashion. For addressing this issue, researchers turned to EAs as a means to avoid local-optima by performing a robust global search on the space of candidate decision trees, usually achieving enhanced predictive performance at the expense of increased computational cost [3].

Recently, Barros et al. [2, 4, 5] have proposed a paradigm shift, and instead of evolving decision trees for individual datasets, they proposed the automatic design of decision-tree induction algorithms tailored to specific application domains such as software effort estimation [9], classification of molecular docking data [7], and cancer classification based on the levels of gene expression [6]. For such, the authors developed a single-objective hyper-heuristic EA that evolved a set of linear-genome-based individuals throughout a fixed number of generations, namely HEAD-DT. Even though the authors usually report good results in terms of predictive performance, the decision trees generated by the automatically-designed algorithms are often more complex than those generated by the top-down algorithms, and thus harder to interpret.

In this paper, we propose addressing the problem faced by HEAD-DT by incorporating a multi-objective strategy that accounts for both predictive performance and tree complexity, two conflicting objectives that should be simultaneously optimized if one desires effective and comprehensible decision trees. We name our proposed approach MOHEAD-DT, which stands for *Multi-Objective Hyper-Heuristic Evolutionary algorithm for Automatically Designing Decision-Tree induction algorithms*.

This paper is organized as follows. Section 2 briefly offers a background on hyper-heuristics and their application to machine learning. Section 3 details our proposed approach for evolving decision-tree induction algorithms. Section 4 describes the methodology that was employed prior to performing the experimental analysis, which is itself presented in Section 5. Finally, we end the paper with our

SAC'15 April 13-17, 2015, Salamanca, Spain.

http://dx.doi.org/10.1145/2695664.2695828

conclusions and suggestions for future work in Section 6.

2. HYPER-HEURISTICS

In the recent years, hyper-heuristics have emerged as a novel field within optimization and evolutionary computation. The main difference between metaheuristics and hyper-heuristics is that, whereas the first operate on a search space of solutions to a given problem, the latter operate on a search space of heuristics, which in turn generate solutions to the problem at hand. Notwithstanding, hyper-heuristics usually employ typical metaheuristics such as EAs as the search methodology to look for heuristics to a given optimization task [18].

To illustrate that rationale, let us compare a metaheuristic approach and a hyper-heuristic approach regarding decision-tree induction for marketing data. In the first approach, an EA is applied to evolve the best decision tree for a dataset of a particular enterprise. In the second approach, an EA is used to evolve the best decision-tree induction algorithm to be further applied to marketing datasets. Note that in the first case, the EA clearly works as a metaheuristic, because it searches for the best decision tree (solution) to the specific marketing dataset, and the ultimate goal is to achieve an accurate decision tree for this particular problem. In the second case, the EA works as a hyper-heuristic, because it searches for the best decision-tree induction algorithm (heuristic), which in turn generates decision trees (solutions) for different instances of marketing applications. The second approach is problem independent, since it generates a decision-tree induction algorithm that can be applied to several distinct marketing datasets.

Most of the hyper-heuristic research aims at solving typical optimization problems (e.g., production scheduling [21] and educational timetabling [16], just to name a few). Applications of hyper-heuristics to machine learning are much more recent than optimization applications. For instance, the work of Pappa and Freitas [17] proposes the evolution of complete rule induction algorithms through grammar-based genetic programming, whereas the work of Vella et al. [22] proposes the evolution of heuristic rules in order to select distinct split criteria in a decision-tree induction algorithm. The work of Basgalupp et al. [8] proposes a hyper-heuristic based on grammatical evolution to automatically design split criteria in decision-tree induction, generating novel criteria instead of selecting them as in [22]. Finally, the work of Barros et al. [4, 5, 6] proposes the evolution of complete decision-tree induction algorithms through a single-objective EA called HEAD-DT. In this paper, we extend HEAD-DT so it can deal with multiple conflicting objectives, generating algorithms capable of providing both accurate and compact decision trees.

3. HYPER-HEURISTIC DECISION-TREE INDUCTION

This section presents a multi-objective EA that evolves generic decision-tree induction algorithms following the greedy top-down approach. The proposed system is called MOHEAD-DT (Multi-Objective Hyper-Heuristic Evolutionary Algorithm for Automatically Designing Decision-Tree Algorithms), and it is a thorough extension of HEAD-DT [5, 6], which is presented in greater detail in the next subsection. MOHEAD-DT allows for a trade-off between predictive performance and model comprehensibility, the latter being of great importance in several application domains [15].

3.1 Background on HEAD-DT

Barros et al. [4, 5, 6] have proposed HEAD-DT (Hyper-Heuristic Evolutionary Algorithm for Automatically Designing Decision-Tree Algorithms), which is a hyper-heuristic single-objective EA capable of automatically designing complete top-down decision-tree induction algorithms. It can be regarded as an alternative to the manual design of such algorithms, in a process called *bias-fitting algorithmic generation*. HEAD-DT makes use of a regular generational single-objective EA in which individuals are collections of building blocks of top-down decision-tree induction algorithms, and typical operators from evolutionary computation are employed, 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.

The encoding of individuals in HEAD-DT is in the form of integer vectors, as depicted in Figure 1. Each gene takes on a value in a predefined range, which is later mapped into specific procedures/functions within the top-down greedy approach of decision-tree induction. The set of genes is divided in four categories, namely: split genes, stopping criteria genes, missing value genes, and pruning genes.

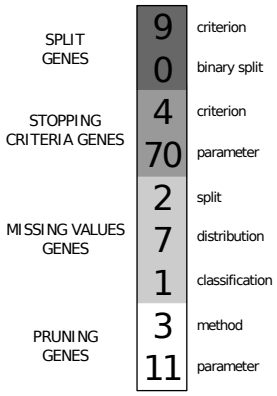

Figure 1: Individual representation in HEAD-DT.

3.1.1 Design Components

The integer vector 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. HEAD-DT makes use of two different genes to model this design component. The first one, **criterion**, is an integer that indexes one of the 15 splitting criteria that are implemented in HEAD-DT. The second gene that controls the split component of a decision-tree algorithm is **binary split**. It is a binary gene that indicates whether the splits of a decision tree are going to be binary or multi-way.

The top-down induction of decision trees is recursive and it continues until a stopping criterion is satisfied. HEAD-DT's genotype holds two genes for representing this design component: **criterion** and **parameter**. The first gene, **criterion**, selects among five different strategies

for stopping the tree growth, whereas gene **parameter** dynamically adjusts a value in the range $[0, 100]$ to the corresponding strategy.

The next design component of decision trees that is represented in HEAD-DT's genotype is the *missing value treatment*. Missing values may be an issue during tree induction and also during classification, and thus HEAD-DT uses three genes to represent missing values strategies in different moments of the induction/deduction process.

Finally, there are two genes for pruning strategies of top-down decision trees in HEAD-DT. The first gene, **method**, indexes one of five well-known approaches for pruning a decision tree, and also the option of not pruning at all. The second gene, **parameter**, is in the range $[0, 100]$ and its value is again dynamically mapped by a function according to the selected pruning method.

In its current implementation, HEAD-DT searches in the space of $21,319,200$ decision-tree induction algorithms, and it is $\approx 2,138$ times faster than the brute-force approach. Of course there are no theoretic guarantees that the (near-)optimal algorithm found by HEAD-DT is going to be the same global optimal algorithm provided by the brute-force approach, but its use is safely justified by the time saved during the process.

3.1.2 Evaluation Framework

In order to compute the fitness of each individual during the evolutionary process, HEAD-DT makes use of a *meta-training set*. A *meta-test set* is used to assess the quality of the evolved algorithm, which is the best individual produced by HEAD-DT. The search for good individuals during evolution can be performed under two distinct frameworks: (i) Evolution of a decision-tree algorithm tailored to one specific dataset at a time (specific framework); and (ii) Evolution of a decision-tree induction algorithm from multiple datasets (general framework).

Previous studies showed through empirical evaluation that HEAD-DT is an effective approach for both specific [4, 5] and general frameworks. The latter was applied with the goal of evolving a decision-tree algorithm tailored to a particular domain, *e.g.*, flexible-receptor docking data [7], software effort prediction [9], and gene expression data classification [6]. In the specific framework, the part of the dataset that belongs to the meta-training set is further divided into training and validation, with typical values being 70% for training and 30% for validation. The term "validation set" is used instead of "test set" to avoid confusion with the meta-test set, and also due to the fact that the "knowledge" within these sets are used to reach for a better solution, and the same cannot be done with test sets. The single-objective fitness function previously employed by HEAD-DT in the specific framework was the F-Measure, which is computed from the decision tree generated by the individual and executed over the validation set.

3.2 Multi-Objective Optimization

This section introduces an extended version of the single-objective HEAD-DT, namely MOHEAD-DT. The main modifications were introduced in the fitness calculation step, in the selection and elitism procedures, and also in the process of selecting the best individual to be returned to the final user. Prior to the description of those modifications, we first briefly describe the multi-objective optimization problem as follows.

When many objectives are simultaneously optimized, there is no single optimal solution for the task at hand. Instead, one has to choose within a set of optimal solutions, each one taking into account a given trade-off among the objectives. The final user can then select its preferred solution from this set of high-quality solutions. Several approaches for handling multi-objective problems were proposed in the literature, and we comment on three of them: i) Weighted-Formula; ii) Pareto Dominance; and iii) Lexicographic Analysis.

The weighted-formula is by far the most common approach in data mining applications [14]. It transforms a multi-objective problem into a single-objective one by assigning numeric weights to each objective to be optimized and then combining the values of the weighted criteria into a single value through arithmetic operations such as addition and multiplication. It has the advantage of being conceptually easy to use and implement, and also of being computationally inexpensive for appointing the optimal solution. Nevertheless, one of its disadvantages is the "magic number" problem: the weights are usually assigned in an ad-hoc fashion, based on the intuition of the user about the relative importance of each objective. Researchers generally justify their options with sentences like "the weights were empirically determined". Another drawback of this approach is that it often mixes non-commensurable criteria, resulting in a meaningless value of an unknown unity, which goes against the trend that discovered knowledge should be understandable to the user.

The Pareto-dominance approach, in turn, does not help the user to select a single alternative from the set of optimal solutions. The concept of Pareto-dominance is as follows. A given solution S_1 dominates solution S_2 if, and only if [11]: (i) S_1 is not worse than S_2 in any of the objectives; and (ii) S_1 is strictly better than S_2 in at least one of the objectives.

Hence, the solutions which are not dominated by any other in an EA population are said to comprise the estimated Pareto front, which is an estimation of the true, unknown Pareto front. The best estimated Pareto front found by the EA is the set of solutions returned to the user, who can then select the best one according to his/her preference. Freitas [14] argues that the difference between the weighted-formula approach and the Pareto approach is the timing when the user has to make a subjective decision. For instance, in the weighted-formula approach, the user has to make a choice of the weight values for each criterion, which is in practice a subjective decision made *a priori*, before the learning algorithm is run. By contrast, in the Pareto approach the user makes a subjective choice *a posteriori*, after he/she has seen all non-dominated solutions returned by the learning algorithm. We can notice that it is much more comfortable to analyze trade-offs associated with the different solutions produced by the algorithm and choose one based on our preference instead of making, in general, an uninformed decision by choosing weights before the algorithm is run.

The lexicographic analysis (or lexicographic ordering) intends to determine priorities among the objectives, and the best solution is the one that is significantly better according to a higher-priority objective. If there is no decision whether a solution is better than the other in a given objective, the next objective is chosen in order of priority. The lexicographic approach might be an interesting choice considering that it recognizes the non-commensurability of the different criteria, and it allows

the user to determine which criteria are more important without the need of identifying the correct weight of each measure. In addition, it preserves the simplicity of the weighted-formula approach and returns a single solution as the best one. Its disadvantage is the threshold values that must be defined *a priori*. Although defining thresholds can be critically appointed as a "magic number" problem, there is a commonplace approach to deal with this situation: statistics-oriented procedures [14]. For instance, standard-deviation based thresholds that allow us to reject a null hypothesis of insignificant difference between two criterion values with a certain degree of confidence. This is a statistically sound solution for this approach, even though the degree of confidence is a parameter that has to be chosen.

3.2.1 Novel Fitness Evaluation

When evolving a novel decision-tree induction algorithm, there are two objectives which should be simultaneously optimized: the predictive performance of the resulting decision tree, which should be maximized, and its complexity, which should be minimized. The advantage of simultaneously optimizing these two conflicting objectives is that we can implicitly control data overfitting, since larger and more complex models usually overfit the training data, losing performance when executed over unseen instances. Moreover, it is much easier to interpret a simpler model than a more complex one.

In MOHEAD-DT, the values of these two objectives are obtained after each individual is converted from genotype (integer vector) to phenotype (decision-tree induction algorithm). The resulting algorithm is then trained with the training part of the meta-training set, and its predictive performance is assessed according to how well it performs on the validation part of the meta-training set (see Figure 2). MOHEAD-DT aims at simultaneously optimizing two objectives: (i) F-Measure, which is a predictive performance measure that indicates a trade-off between precision and recall and is suitable for both balanced and imbalanced data; and (ii) number of nodes in the tree, which is a suitable estimation of model complexity for decision trees. The smaller the model, the more comprehensible it is, and also less-prone to overfitting[1].

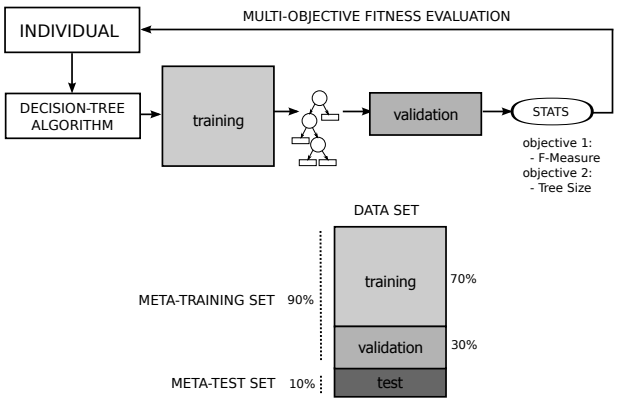

Figure 2: Fitness evaluation scheme.

For optimizing both objectives, MOHEAD-DT allows the user to choose between the Pareto dominance approach and the lexicographic analysis, according to his/her preference. Depending on the multi-objective strategy selected by the user, each individual is assigned a *pseudo* fitness value after the F-Measure and tree size of the trees generated by them have been computed.

In the case of the lexicographic analysis, the individuals are compared in a pairwise fashion, and the *pseudo* fitness value assigned to them is the total number of victories in all possible pairwise lexicographical comparisons. Let A be a given individual and A_i any other individual in the current population. A is said to be the winner of the lexicographical comparison regarding A_i if the difference between their i^{th} objective surpasses the objective's threshold Δ_i. The objectives are analyzed in a pre-defined order of priority, and in MOHEAD-DT F-Measure (predictive performance) has greater priority than tree size (model complexity). Since MOHEAD-DT analyzes two objectives, one needs to set two parameters, the objectives' thresholds Δ_1 and Δ_2.

Regarding the Pareto dominance approach, the *pseudo* fitness value is given by $n_{Dom} - n_{IsDom}$, in which n_{Dom} is the number of times the individual dominates other individuals in the population, and n_{IsDom} is the number of times the individual is dominated.

3.2.2 Selection and Elitism

For selection and elitism, MOHEAD-DT makes use of the *pseudo* fitness value of each individual much the same as it would do for a single-objective EA. We are aware that other multi-objective algorithms adopt more sophisticated strategies instead of using a single *pseudo* fitness values. For instance, NSGA-II [12] and SPEA2 [23] give preference to solutions found in the less-crowded regions of the solution space. MOHEAD-DT does not implement any mechanisms capable of enforcing the distribution of the solutions in different areas of the search space, but we later argue that its results show a good diversity of solutions in the Pareto front. A comparative analysis between the current implementation of MOHEAD-DT and the use of other multi-objective strategies such as NSGA-II and SPEA2 is left for future research.

As a final remark, note that, for the case of the Pareto strategy in MOHEAD-DT, we have to be able to treat one special case: individuals whose decision-tree induction algorithms produce models consisting of only one node. In other words, these individuals are generating trees that are predicting the most frequent class among the training instances for all new instances in the validation set. For these individuals, the optimal number of nodes is achieved (since we are minimizing tree size), but in this case the results obtained by the candidate algorithm is simply a trivial majority classifier. To avoid this situation, the number of nodes of a model with 1 node is set to 1,000.

4. EXPERIMENTAL METHODOLOGY

In Table 1, we present the public datasets from the UCI machine-learning repository [1] in order to assess the relative performance of the algorithms automatically designed by MOHEAD-DT. We compare the resulting decision-tree induction algorithms with the two most widely-used decision-tree induction algorithms: C4.5 [20] and CART [10]. For each dataset, we report the F-Measure and the size of the generated decision tree. The results are based on the average of 10-fold cross-validation runs.

[1]However, smaller models may eventually suffer from underfitting, and thus the need of optimizing predictive performance.

Table 1: UCI datasets that were employed in the experimental analysis.

dataset	# instances	# attributes	# numeric attributes	# nominal attributes	% missing	# classes
abalone	4177	9	7	1	0.00	28
anneal	898	39	6	32	0.00	6
arrhythmia	452	280	206	73	0.32	16
audiology	226	70	0	69	2.03	24
bridges_version1	107	13	3	9	5.53	6
car	1728	7	0	6	0.00	4
cylinder_bands	540	40	18	21	4.74	2
glass	214	10	9	0	0.00	7
hepatitis	155	20	6	13	5.67	2
iris	150	5	4	0	0.00	3
kdd_synthetic	600	62	60	1	0.00	6
segment	2310	20	19	0	0.00	7
semeion	1593	266	265	0	0.00	2
shuttle_landing	15	7	0	6	28.89	2
sick	3772	30	7	22	5.54	2
tempdiag	120	8	1	6	0.00	2
tep.fea	3572	8	7	0	0.00	3
vowel	990	14	10	3	0.00	11
winequality_red	1599	12	11	0	0.00	10
winequality_white	4898	12	11	0	0.00	10

Additionally, since MOHEAD-DT is a non-deterministic method, we averaged the results of 5 different runs by varying the random seed.

The baseline algorithms CART [10] and C4.5 [20] were set with their default parameters, which typically represent robust values that work well across different datasets. None of the algorithms, including HEAD-DT and MOHEAD-DT, had their parameters optimized to individual datasets.

HEAD-DT and MOHEAD-DT were employed using the specific framework. They were configured with the parameters described in Table 2, which are typical values for EAs in the context of decision trees [3]. The lexicographic thresholds (Δ_1 and Δ_2 in Section 3.2.1) for MOHEAD-L were set to 2% and 2 nodes for F-Measure and tree size, respectively.

Table 2: Parameter values for MOHEAD-L, HOEAD-P and HEAD-DT.

	MOHEAD-L	MOHEAD-P	HEAD-DT
population size	100	100	100
number of generations	100	100	100
tournament size	2	2	2
elitism rate	-	5%	5%
crossover rate	90%	90%	90%
reprodution rate	5%	5%	5%
mutation rate	5%	5%	5%

In order to provide some reassurance about the validity and non-randomness of the results, statistical tests are applied by following the approach proposed by Demšar [13]. These tests seek to compare multiple algorithms on multiple datasets, and are based on the use of the Friedman test with a corresponding post-hoc test. Let R_i^j be the rank of the j^{th} of k algorithms on the i^{th} of N datasets, the Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i R_i^j$. The original Friedman statistic and its adjusted less-conservative version are given by:

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

$$F_f = \frac{(N-1)\times\chi_F^2}{N\times(k-1)-\chi_F^2}, \quad (2)$$

where the adjusted version is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. If the null hypothesis of similar performance is rejected, then we proceed with the Nemenyi post-hoc test for pairwise comparisons. The performance of two classifiers is significantly different if their corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} \quad (3)$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$.

5. RESULTS

Table 3 shows the classification F-Measure and tree sizes $\pm$ the standard deviations from the trees induced by algorithms automatically designed by the two versions of MOHEAD-DT: MOHEAD-L (based on the lexicographic analysis) and MOHEAD-P (based on the Pareto dominance), as well as the results of the single-objective approach HEAD-DT.

Results show that HEAD-DT generates better algorithms in terms of predictive performance (F-Measure) in 15 out of the 20 datasets. MOHEAD-L generates the best algorithm in 6 datasets, whereas MOHEAD-P does it for two datasets.

To evaluate the statistical significance of the F-Measures results, we calculated the average Friedman rank for MOHEAD-L, MOHEAD-P, and HEAD-DT: 1.875, 2.625, and 1.5, respectively. The average rank suggests that HEAD-DT outperforms both multi-objective versions regarding predictive performance. The calculation of Iman's F statistic resulted in $F_f = 9.28$. Critical value of $F(k-1,(k-1)(n-1)) = F(2,38)$ for $\alpha = 0.05$ is 3.25. Since $F_f > F_{0.05}(2,38)$ $(9.28 > 3.25)$, the null-hypothesis is rejected. We proceed with a post-hoc Nemenyi test to find which method provides better results in a pairwise fashion. The critical difference is $CD = 0.74$. The differences between the average rank of HEAD-DT and the rank of the multi-objective versions MOHEAD-L and MOHEAD-P are 0.38 and 1.13, respectively, and the difference between MOHEAD-L and MOHEAD-P is 0.75.

Therefore, we can assert that both MOHEAD-L and HEAD-DT generate algorithms significantly better than

Table 3: Results for the comparison among the hyper-heuristics.

	F-Measure						Tree Size					
	MOHEAD-L		MOHEAD-P		HEAD-DT		MOHEAD-L		MOHEAD-P		HEAD-DT	
abalone.data	**0.23**	**0.01**	**0.23**	**0.01**	0.20	0.02	48.78	31.05	**21.54**	**4.91**	4068.12	13.90
anneal	0.97	0.01	0.97	0.01	**0.99**	**0.01**	**14.38**	**2.12**	15.92	1.66	55.72	3.66
arrhythmia	**0.66**	**0.06**	0.64	0.06	0.63	0.06	20.04	3.51	**13.20**	**2.65**	171.84	5.18
audiology	0.71	0.06	0.63	0.08	**0.79**	**0.07**	45.86	6.14	**19.32**	**3.44**	118.60	3.81
bridges-version1	**0.59**	**0.05**	0.55	0.05	0.56	0.12	20.56	3.91	**8.44**	**1.50**	156.88	14.34
car	0.95	0.02	0.91	0.01	**0.98**	**0.01**	58.68	9.71	**31.28**	**6.76**	171.92	4.45
cylinder-bands	0.70	0.04	0.69	0.03	**0.72**	**0.04**	40.56	20.22	**12.52**	**2.94**	211.44	9.39
Glass	0.67	0.08	0.63	0.08	**0.72**	**0.09**	26.72	4.99	**14.52**	**3.49**	86.44	3.14
hepatitis	0.75	0.07	0.78	0.08	**0.80**	**0.08**	11.32	5.24	**3.68**	**0.87**	71.80	4.77
iris	0.95	0.06	0.94	0.06	**0.95**	**0.05**	6.24	0.76	**5.60**	**0.60**	20.36	1.81
segment	0.95	0.01	0.94	0.02	**0.97**	**0.03**	38.32	6.89	**28.92**	**3.03**	26.16	2.45
semeion	**0.99**	**0.01**	0.97	0.02	0.97	0.01	19.64	1.51	**14.56**	**3.77**	132.76	3.48
shuttle-landing-control	0.99	0.01	0.97	0.02	**1.00**	**0.00**	19.64	1.51	**14.56**	**3.77**	19.00	0.00
sick	**0.98**	**0.00**	0.98	0.01	0.93	0.20	5.72	0.67	9.28	3.63	**5.64**	**1.69**
synthetic_control	0.88	0.04	0.85	0.02	**0.99**	**0.00**	28.00	4.55	**16.96**	**2.79**	153.70	8.89
tempdiag	**1.00**	**0.00**	0.99	0.01	**1.00**	**0.00**	5.00	0.00	**4.80**	**0.28**	5.32	1.04
tep.fea	0.60	0.02	**0.61**	**0.02**	**0.61**	**0.02**	**3.00**	**0.00**	7.20	1.62	18.84	1.97
vowel	0.79	0.03	0.67	0.07	**0.89**	**0.03**	206.12	32.63	**86.60**	**26.13**	361.42	5.54
winequality-red	0.56	0.03	0.55	0.02	**0.63**	**0.03**	74.12	50.74	**17.16**	**2.13**	796.00	11.22
winequality-white	0.55	0.02	0.51	0.03	**0.63**	**0.03**	653.80	147.32	**125.04**	**109.26**	2525.88	13.17

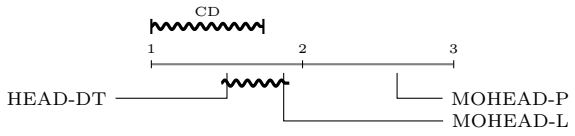

Figure 3: Critical diagram for the F-Measure of the hyper-heuristics.

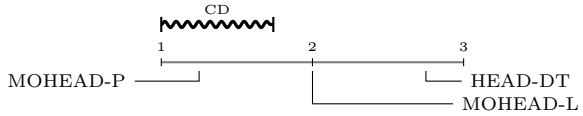

Figure 4: Critical diagram for the tree size of the hyper-heuristics.

those generated by MOHEAD-P. Nevertheless, it is important to note that there is no significant difference between MOHEAD-L and HEAD-DT in terms of F-Measure (see the critical diagram in Figure 3).

When analyzing the size of trees, we can observe that MOHEAD-P provided algorithms that generate smaller trees in 17 out of the 20 datasets. MOHEAD-L did it for 2 datasets, while HEAD-DT did it for only 1 dataset. Regarding the statistical analysis, the computed value of $F_f = 24.43$. Since $F_f > F_{0.05}(2, 38)$ $(24.43 > 3.25)$, the null-hypothesis is rejected. Once again, we proceed with a post-hoc Nemenyi test, and the critical difference is $CD = 0.74$. The differences between the average rank of HEAD-DT and the rank of the multi-objective versions MOHEAD-L and MOHEAD-P are 0.75 and 1.5, respectively, and the difference between MOHEAD-L and MOHEAD-P is 0.75. Hence, we can affirm that both MOHEAD-L and MOHEAD-P automatically design algorithms whose decision trees are significantly smaller than the ones generated by HEAD-DT. The critical diagram is depicted in Figure 4).

Since smaller trees are only preferable in scenarios where the predictive performance of the models is similar, the fact that MOHEAD-P designs algorithms whose trees are smaller than both MOHEAD-L and HEAD-DT should not be relevant. Indeed, the previous analysis clearly indicates that MOHEAD-P generates algorithms whose trees are outperformed by both MOHEAD-L and HEAD-DT regarding F-Measure with statistical significance. The Occam's razor assumption that among competitive hypotheses, the simpler is preferred, does not apply in this case. Notwithstanding, we can make use of the Occam's razor assumption when comparing MOHEAD-L with HEAD-DT, considering that there is no significant difference between them in terms of predictive performance, and that MOHEAD-L generates algorithms whose decision trees are significantly smaller than those generated by the algorithms designed by HEAD-DT.

Given that MOHEAD-L presents the best trade-off between predictive performance and model comprehensibility among the hyper-heuristics that automatically design decision-tree induction algorithms, we proceed to compare the results of MOHEAD-L with the baseline greedy top-down decision-tree induction algorithms, namely CART [10] and C4.5 [20].

Table 4 shows the classification F-Measure provided by C4.5, CART, and MOHEAD-L's automatically designed algorithm. It presents the average F-Measure over the 10-fold cross-validation runs ± the standard deviation (best absolute values in bold). Note that MOHEAD-L generates algorithms whose trees have superior predictive performance in 10 out of the 20 datasets. CART provides better trees in 5 datasets, and C4.5 in 9 datasets. Observe that there are cases in which all methods provided the same tree (tempdiag dataset), and also that there are cases in which both CART and C4.5 (but not MOHEAD-L's algorithm) induced the same tree (anneal and synthetic_control datasets).

To evaluate the statistical significance of the F-Measure results, we calculated the average rank for MOHEAD-L, CART, and C4.5: 1.825, 2.25 and 1.985, respectively. The average rank suggests that MOHEAD-L is the best performing method regarding predictive performance. The calculation of Iman's F statistic resulted in $F_f = 0.99$. Critical value of $F(2, 38)$ for $\alpha = 0.05$ is 3.25. Since $F_f < F_{0.05}(2, 38)$ $(0.99 < 3.25)$, the null-hypothesis is accepted. Even though the Friedman test does not indicate a significant difference among the methods, we can proceed to the pairwise Nemenyi's test, which is less conservative

Table 4: Results for the comparison among MOHEAD-L, CART, and C4.5.

	F-Measure						Tree Size					
	MOHEAD-L		CART		C4.5		MOHEAD-L		CART		C4.5	
abalone.data	**0.23**	**0.01**	0.22	0.02	0.21	0.02	48.78	31.05	**44.40**	**16.00**	2088.90	37.63
anneal	0.97	0.01	**0.98**	**0.01**	**0.98**	**0.01**	**14.38**	**2.12**	21.00	3.13	48.30	6.48
arrhythmia	0.66	0.06	**0.67**	**0.05**	0.64	0.05	**20.04**	**3.51**	23.20	2.90	82.60	5.80
audiology	0.71	0.06	0.70	0.04	**0.75**	**0.08**	45.86	6.14	**35.80**	**11.75**	50.40	4.01
bridges-version1	**0.59**	**0.05**	0.44	0.06	0.52	0.10	20.56	3.91	**1.00**	**0.00**	24.90	20.72
car	**0.95**	**0.02**	0.93	0.97	0.93	0.02	**58.68**	**9.71**	108.20	16.09	173.10	6.51
cylinder-bands	**0.70**	**0.04**	0.54	0.07	0.42	0.00	40.56	20.22	4.20	1.03	**1.00**	**0.00**
Glass	0.67	0.08	0.67	0.10	**0.67**	**0.05**	26.72	4.99	**23.20**	**10.56**	44.80	5.20
hepatitis	0.75	0.07	0.74	0.07	**0.77**	**0.06**	11.32	5.24	**6.60**	**8.58**	15.40	4.40
iris	**0.95**	**0.06**	0.93	0.05	0.93	0.06	6.24	0.76	**6.20**	**1.69**	8.00	1.41
segment	**0.95**	**0.01**	0.88	0.03	0.90	0.04	38.32	6.89	**1.00**	**0.00**	37.80	4.34
semeion	**0.99**	**0.01**	0.95	0.01	0.96	0.09	**19.64**	**1.51**	78.00	8.18	80.60	4.97
shuttle-landing-control	**0.99**	**0.01**	0.93	0.01	0.95	0.02	**19.64**	**1.51**	34.00	12.30	55.00	8.27
sick	**0.98**	**0.00**	0.56	0.03	0.56	0.38	5.72	0.67	**1.00**	**0.00**	**1.00**	**0.00**
synthetic_control	0.88	0.04	**0.98**	**0.00**	**0.98**	**0.00**	**28.00**	**4.55**	45.20	11.33	46.90	9.41
tempdiag	**1.00**	**0.00**	**1.00**	**0.00**	**1.00**	0.00	**5.00**	**0.00**	**5.00**	**0.00**	**5.00**	**0.00**
tep.fea	0.60	0.02	0.60	0.02	**0.61**	**0.02**	**3.00**	**0.00**	13.00	2.83	8.20	1.69
vowel	0.79	0.03	0.81	0.03	**0.82**	**0.03**	206.12	32.63	**175.80**	**23.72**	220.70	20.73
winequality-red	0.56	0.03	**0.61**	**0.02**	0.60	0.03	**74.12**	**50.74**	151.80	54.58	387.00	26.55
winequality-white	0.55	0.02	0.57	0.02	**0.60**	**0.02**	**653.80**	**147.32**	843.80	309.01	1367.20	58.44

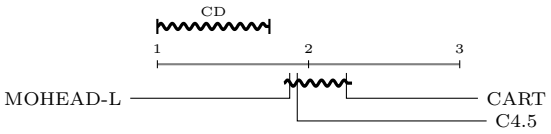

Figure 5: Critical diagram for the F-Measure of MOHEAD-L, CART, and C4.5.

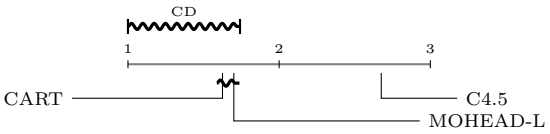

Figure 6: Critical diagram for the tree size of MOHEAD-L, CART, and C4.5.

than Friedman's. The critical difference is $CD = 0.74$. The difference between the average rank of MOHEAD-L and C4.5 is 0.1 and that between MOHEAD-L and CART is 0.425. Since there is no difference greater than 0.74, we can reinforce Friedman's conclusion that there is no significant difference among the three methods regarding F-Measure. The critical diagram is depicted in Figure 5.

In terms of model comprehensibility, results in Table 4 show that MOHEAD-L generates smaller trees in 10 out of the 20 datasets. CART also generates smaller trees in 10 datasets, and C4.5 in only three datasets. We calculated the average Friedman rank for MOHEAD-L, CART, and C4.5: 1.7, 1.625, and 2.675, respectively. Regarding the statistical analysis, the computed value of $F_f = 9.92$. Since $F_f > F_{0.05}(2, 38)$ $(9.92 > 3.25)$, the null-hypothesis is rejected. Once again, we proceed with the post-hoc Nemenyi's test, and the critical difference is $CD = 0.74$.

The difference between MOHEAD-L and CART is 0.075, much smaller than CD (0.74). On the other hand, the difference between MOHEAD-L and C4.5 is 0.975, and between CART and C4.5 is 1.05. Thus, we can assert that both MOHEAD-L's algorithms and CART generate decision trees significantly smaller than those generated by C4.5. It should be also stressed out that there are no significant differences between MOHEAD-L and CART in terms of tree size (see the critical diagram in Figure 6).

Since there is no significant difference between MOHEAD-L and CART, we can observe that, in terms of predictive performance (F-Measure) the average rank of MOHEAD-L (1.825) is better than CART's (2.25). In addition, MOHEAD-L generated better trees than CART in 11 out 20 datasets, whereas CART generated better trees in only 6 datasets, in terms of F-Measure. Also, considering the absolute values, the same superiority could not be observed when analyzing tree sizes, since MOHEAD-L's algorithms generated smaller trees in 9 out of 20 datasets and CART in 10 out 20. Even though the results are not conclusive in terms of which algorithm generates better decision trees (MOHEAD-L or CART), we believe they are encouraging since they show that MOHEAD-L's algorithms often generate more accurate decision trees than CART without significant loss in model comprehensibility.

6. CONCLUSIONS AND FUTURE WORK

This work presented a multi-objective evolutionary algorithm based hyper-heuristic capable of automatically designing top-down greedy decision-tree induction algorithms, namely MOHEAD-DT (Multi-Objective Hyper-Heuristic Evolutionary Algorithm for Automatically Designing Decision-Tree Algorithms). MOHEAD-DT is an extended version of HEAD-DT [4, 5], which is a single-objective hyper-heuristic that achieved promising results in generating decision-tree induction algorithms for application domains such as software effort prediction [9], gene expression classification [6], and flexible-receptor molecular docking data [7].

Comparisons between MOHEAD-DT and HEAD-DT show that the first is capable of designing decision-tree induction algorithms with competitive predictive performance and more compact models, following the Occam's razor science principle of preference for simpler solutions. Given that in many application domains interpretability is crucial and as important as predictive performance, these seem to be quite positive results.

When comparing MOHEAD-DT with two other traditional greedy top-down decision-tree induction

algorithms, similar conclusions were drawn, as it was often the case that the algorithms provided by MOHEAD-DT generated significantly smaller trees with similar predictive performance than the trees provided by C4.5 [20]. Even though the statistical analysis did not suggest significant differences between MOHEAD-DT and CART [10], we believe the lexicographic version of MOHEAD-DT is preferable because of its better results regarding predictive performance, measured in terms of F-Measure.

As future research, we believe that employing different multi-objective optimization techniques such as NSGA-II [12] and SPEA2 [23] are good opportunities for improving MOHEAD-DT's performance. In addition, we believe that objectives such as algorithmic complexity or execution time could be optimized in order to generate algorithms suitable for big data processing.

7. ACKNOWLEDGMENTS

The authors would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) for funding this research.

8. REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas. Towards the automatic design of decision tree induction algorithms. In *13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO 2011)*, pages 567–574, 2011.

[3] 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 Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(3):291–312, 2012.

[4] 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 *14th Genetic and Evolutionary Computation Conference (GECCO 2012)*, pages 1237–1244, 2012.

[5] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas. Automatic Design of Decision-Tree Algorithms with Evolutionary Algorithms. *Evolutionary Computation*, 21(4), 2013.

[6] R. C. Barros, M. P. Basgalupp, A. A. Freitas, and A. C. P. L. F. de Carvalho. Evolutionary Design of Decision-Tree Algorithms Tailored to Microarray Gene Expression Data Sets. *IEEE Transactions on Evolutionary Computation*, in press, 2014.

[7] R. C. Barros, A. T. Winck, K. S. Machado, M. P. Basgalupp, A. C. P. L. F. de Carvalho, D. D. Ruiz, and O. S. de Souza. Automatic design of decision-tree induction algorithms tailored to flexible-receptor docking data. *BMC Bioinformatics*, 13, 2012.

[8] M. P. Basgalupp, R. C. Barros, and T. Barabasz. A Grammatical Evolution Based Hyper-heuristic for the Automatic Design of Split Criteria. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO 2014)*, GECCO '14, pages 1311–1318, New York, NY, USA, 2014. ACM.

[9] M. P. Basgalupp, R. C. Barros, T. S. da Silva, and A. C. P. L. F. de Carvalho. Software effort prediction: a hyper-heuristic decision-tree based approach. In *28th Annual ACM Symposium on Applied Computing*, pages 1109–1116, 2013.

[10] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[11] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[12] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 849–858, London, UK, UK, 2000. Springer-Verlag.

[13] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[14] A. A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explor. Newsl.*, 6(2):77–86, 2004.

[15] 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*, 7:172–182, January 2010.

[16] G. Ochoa, R. Qu, and E. K. Burke. Analyzing the landscape of a graph based hyper-heuristic for timetabling problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 341–348, New York, NY, USA, 2009. ACM.

[17] G. L. Pappa and A. A. Freitas. *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer Publishing Company, Incorporated, 2009.

[18] G. L. Pappa, G. Ochoa, M. R. Hyde, A. A. Freitas, J. Woodward, and J. Swan. Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, 15:3–35, 2014.

[19] V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman. Decision trees: An overview and their use in medicine. *Journal of Medical Systems*, 26:445–463, 2002. 10.1023/A:1016409317640.

[20] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco, CA, USA, 1993.

[21] J. A. Vázquez-Rodríguez and S. Petrovic. A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics*, 16(6):771–793, Dec. 2010.

[22] A. Vella, D. Corne, and C. Murphy. Hyper-heuristic decision tree induction. *W CONF NAT BIOINSP COMP*, pages 409–414, 2010.

[23] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou, D. Tshalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimization, and Control*, pages 19–26. 2002.