A Hyper-Heuristic Evolutionary Algorithm for Learning Bayesian Network Classifiers

Conference Paper · November 2014

DOI: 10.1007/978-3-319-12027-0_35

2 authors:

Alex G. C. De Sá
Federal University of Minas Gerais
19 PUBLICATIONS **49** CITATIONS

SEE PROFILE

Gisele L. Pappa
Federal University of Minas Gerais
116 PUBLICATIONS **1,010** CITATIONS

SEE PROFILE

A Hyper-heuristic Evolutionary Algorithm for Learning Bayesian Network Classifiers

Alex G. C. de Sá and Gisele L. Pappa

Computer Science Department, Universidade Federal de Minas Gerais,
Belo Horizonte, Brazil
{alexgcsa,glpappa}@dcc.ufmg.br

Abstract. Hyper-heuristic evolutionary algorithms (HHEA) are successful methods for selecting and building new heuristics or algorithms to solve optimization or machine learning problems. They were conceived to help answer questions such as: given a new classification dataset, which of the solutions already proposed in the literature is the most appropriate to solve this new problem? In this direction, we propose a HHEA to automatically build Bayesian Network Classifier (BNC) tailored to a specific dataset. BNCs are powerful classification models that can deal with missing data, uncertainty and generate interpretable models. The method receives an input a set of components already present in current BNC algorithms and a specific dataset. The HHEA then searches for the best combination of components according to the input dataset. Results show the customized algorithms generated obtain results of F-measure equivalent or better than other state of the art BNC algorithms.

Keywords: Hyper-heuristics, Automatic Algorithm Design, Bayesian Network Classifiers.

1 Introduction

Hyper-heuristic evolutionary algorithms are successful methods for selecting and building new heuristics or algorithms to solve optimization or machine learning problems [13]. They were conceived to help experts and practitioners in the following task: given a new classification data set or a new instance of the traveling salesman problem, which of the solutions already proposed in the literature is the most appropriate to solve this new problem?

The answer to this question can be given in two different levels: (i) by selecting existing heuristics/algorithms [3] or (ii) by building new heuristics/algorithms using components from different existing algorithms [14]. In the context of machine learning, the first level can be solved using meta-learning algorithms [5]. However, in this work we are in interested in a different approach: selecting components from existing classification algorithms to produce a potentially better and customized one. Given a new dataset, we build (novel) algorithms by selecting the most promising components previously proposed. We do that by using a hyper-heuristic evolutionary algorithm (HHEA), which is used to automatically build a specific type of classification algorithms: Bayesian Networks Classifiers.

Bayesian Networks Classifiers (BNCs) [11] are robust and precise statistical methods for data classification based on the theoretical foundations of Bayesian networks [6]. They produce a classification model that assumes cause-effect relations among all data attributes (including the class). They represent data using a directed acyclic graph, where each node maps an attribute and edges define probabilistic dependencies among them. Furthermore, each node is associated with a conditional probability table, which represents the network parameters.

BNCs are interesting methods for classification because (i) they encode the dependencies among all variables of the problem, and are ready to deal with lack of data; (ii) they learn causal relationships and, therefore, can be used to gain understanding about a problem domain and to predict the consequences of events; (iii) they can be interpreted by domain specialists.

The most well-known and simplest BNC approach is the Naïve Bayes algorithm [21]. In this case, the network considers independence among all predictive attributes (they are no edges connecting them) and all of them depend on the class attribute. The first graph-based BNC, in turn, was presented in the early nineties [7], and since then many others were proposed. However, determining which algorithm should be used in a specific dataset is an open problem [5]. In this direction, the proposed HHEA can help finding an appropriate algorithm to any given dataset.

BNCs are usually built in two phases: structure learning and parameters learning. The first phase is more tricky than the second, as after having the network structure, learning the parameters in a more straightforward process [17]. Hence, the proposed HHEA focus on the first phase, where different search methods, attribute selection and evaluation metrics, among other components, are combined to produce new, customized BNC algorithms.

The proposed method works as follows. It receives a list of the main components of BNC algorithms, and uses an HHEA to encode these components. Given an input dataset, the method tests different combinations of components to that specific dataset. In order to evaluate the performance of the BNC generated, the algorithm is trained with a subset of the domain data available, and its accuracy assessed in a validation set. At the end of the evolutionary process, the best algorithm found is tested in data from the same application domain.

A preliminary version of this work was presented in Sá & Pappa [15], where the algorithm dealt with fewer BNC components (the search space of the evolutionary algorithm was very restricted), was tested in a limited number of datasets and used a different individual-components mapping scheme. The new individual representation proposed here is more robust and considers the dynamic nature of the components of BNC algorithms by using a real-coded mapping (in contrast with the previous static integer-coded representation). It is also important to emphasize that this work is one of the first efforts to consider the problem of generating a customized classification algorithm to a given dataset.

The algorithm was tested in 15 different datasets from different domains extracted from the UCI Repository [2], and compared to three popular BNCs: Naïve Bayes [21], Tree-Augmented Naïve Bayes (TAN) [11] and K2 [7]. Results showed that, in average, the values of F-measure obtained by HHEA are equivalent to those obtained by the algorithms already proposed in the literature.

However, looking carefully at each evolved algorithm in isolation, we observe they can produce BNC algorithms very different from the state of the art, and with F-measures equivalent or better.

The reminder of this paper is organized as follows. Section 2 reviews related work in the automatic evolution of algorithms with hyper-heuristics. Section 3 details the proposed method, while Section 4 presents and discusses the results obtained. Finally, Section 5 draws some conclusions and discusses directions of future work.

2 Related Work

The use of HHEAs to generate machine learning algorithms customized to datasets is an outgrowing research field. We focus on works which dealt with this problem with three different classification models: decision trees [3], artificial neural networks[18,22] and rule induction algorithms [14]. Besides, we review the work of Thornton *et al.* [19], where a different search method was used and considered different types of classification algorithms simultaneously.

Artificial neural networks were the first methods to be automatically evolved using evolutionary algorithms. As discussed by Yao [22], the methods can evolve new neural networks considering three levels of abstraction: (i) synaptic weights, (ii) topology design or (iii) selection of the learning rule. One work that combines the three aspects aforementioned in different ways is Floreano *et al.* [10], which proposed neuroevolution. One specific example of neuroevolution is NEAT (Neuroevolution of Augmentative Topologies) [18], which evolves simultaneously the topology and weights of the network.

Pappa & Freitas [14], in turn, proposed a grammar-based evolutionary algorithm to guide the process of automatically evolving a rule induction algorithm. They considered three different components: search method, rule evaluation and pruning. Different from previous work, a genetic programming algorithms was used to allow creating completely new algorithms, considering programming primitives such as loops and conditionals. The results showed the HHEA generated new algorithms competitive with the state of the art.

Regarding decision trees, Barros *et al.* [3] proposed HEAD-DT (Hyper-heuristic Evolutionary Algorithm for Automatically Designing Decision-Tree algorithms). The authors identified four components that can change significantly the results of decision tree algorithms: splitting criterion, stop criterion, pruning and the way missing values are treated. The evolutionary algorithms was then evolved to consider variations of these four main components, and results showed the automatically generated algorithms obtain better results of accuracy than other state of the art decision tree algorithms. Following this line, the work proposed here automatically evolves customized Bayesian Network algorithms for a given dataset.

Concerning methods not based on HHEAs, Thornton *et al.* [19] recently proposed Auto-WEKA to consider the problem of automatically selecting a learning algorithm and tuning its hyper-parameters. Auto-WEKA uses as input the feature selection algorithms and all classification algorithms implemented

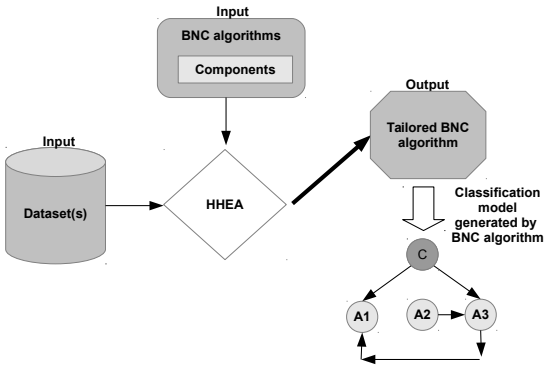

Fig. 1. HHEA scheme for evolving BNC algorithms.

in WEKA [21], being more generic than the aforementioned algorithms. The searching process used by Auto-WEKA is based on sequential model-based optimization.

3 Evolving Algorithms for Learning Bayesian Network Classifiers

This section presents the proposed approach to automatically evolve algorithms for learning BNCs, illustrated in Figure 1. The method receives as input a dataset and a set of components identified from previously proposed BNC algorithms. Then, a HHEA is used to combine this components, outputting a BNC algorithm tailored to the domain of the input data.

In the HHEA, each individual represents a BNC algorithm (see Section 3.2), randomly generated from a combination of the available components (described in Section 3.1). During the evaluation of the individuals, a mapping between the individual and a BNC algorithm is performed (see Section 3.3).

Following, the individuals undergo uniform crossover and one-point mutation operations to generate a new population, and a elitist process copies the best individual to the next population. After a predefined number of generations, the best BNC algorithm generated is returned, and its associated model tested using a new set of data coming from the same domain used for training.

3.1 Components of BNC Algorithms

One of the most important steps in the conception of the HHEA is to identify a set of relevant components the search algorithm should explore to build a solution (BNC algorithm) tailored to a specific dataset.

As already mentioned, the process of learning BNCs occurs in two phases: structure learning and parameter learning. In the structure learning phase, the idea is to learn the causal relationships among the attributes of the input dataset,

i.e. which nodes (attributes) in the graph should be connected to each other. Different types of BNC algorithms will work in different ways here, and they can be score-based, constraint-based or hybrids [8]. Score-based methods use a score metric, such as the Entropy criterion, to guide the search process, which can be performed by virtually any search algorithm. Constraint-based approaches, in contrast, use a conditional independence test, such as the χ^2, to guide the graph construction. Hybrid method combine the two previous approaches.

The parameters learning phase, in turn, learns the Conditional Probability Tables (CPTs) for each node of the BNC. These tables are used to make estimations about the data. However, learning the parameters of a BNC is a relatively straightforward procedure when the network structure is defined with specific dependencies among the variables [17]. For this reason, this paper will focus on the structure learning phase, and will always consider the same parameter estimation method. It uses a simple estimator tuned according to a parameter α, which represents the initial count on each probability value and is used to estimate the probability tables. The value of α is also tuned by HHEA.

When learning the structure of the network, we can generate Naïve Bayes like structures, which assume all attributes are independent, or generate networks represented by graphs or trees. According to the first component choice, we determine the type of algorithm being generated (Naïve Bayes, score-based, constraint-base or hybrid) and, consequently, the type of model being generated (i.e. tree, graph, none).

Concerning this search algorithm, 12 different options can be performed: Naïve Bayes [21] (NB), *Tree Augmented Naïve Bayes (TAN) using Conditional Independence Tests* [11], *Inductive Causation Search* (ICS) [20], Hybrid ICS (ICS-H), *General Augmented Naïve Bayes* (GAN) [16], Hybrid GAN (GAN-H) Greedy Search of K2 algorithm (GSK2) [7], *Hill Climbing* (HC) [12], *Look Ahead in Good Directions Hill Climbing* (LAGDHC) [1], *Repeated Hill Climbing* (RHC) [12], Tabu Search (Tabu) [4] and *Simulated Annealing* (SA) [4].

Depending on the choice performed in the first level, a new set of parameters is chosen. Table 1 shows which components the search methods depend on. Note that here, due to lack of space, we do not specify all the values they can assume, but the search space of BNC algorithms is bounded to 4,960,000 possible solutions. For details about these components, see the works of Bouckaert et al. [4], Witten et al. [21] and Sacha [16].

In Table 1, each column represents the template of a different search algorithm, which is the first gene of the individual. Each line shows a different component, which might suit or not the chosen search procedure, which will describe now. The first line, *NB as initial structure*, says that the search starts considering no dependencies among predictive attributes. *Structure complexity*, in turn, defines whether a tree or a forest can be generated by an algorithm. *Arc reversal* says whether this operation should be used during the search. The *Markov Blanket Classifier (MBC)*, in turn, applies a correction on the BNC structure that excludes nodes outside the Markov Blanket of the class node [8]. *Feature selection* chooses the most appropriate subset of attributes that should be class-dependent.

Table 1. Dependency relationship between search method and other components.

Components	Structure Learning											
	NB	**TAN**	**ICS**	**ICS-H**	**GAN**	**GAN-H**	**GSK2**	**HC**	**LAGDHC**	**RHC**	**Tabu**	**SA**
NB as initial structure	X						X	X	X	X	X	
Structure complexity					X	X						
Use arc reversal							X	X	X	X	X	
Markov Blanket Classifier		X	X	X	X	X	X	X	X	X	X	X
Feature selection		X		X	X	X	X	X	X	X	X	X
Scoring metrics				X	X	X	X	X	X	X	X	X
Accuracy estimation					X	X	X	X		X	X	X
Independence test		X	X	X		X						
Maximal cardinality			X	X								
Number of Parents							X	X	X	X	X	
# of look ahead steps									X			
# of operations									X			
Tabu list length											X	
Initial temperature and Δ												X
Iterations										X	X	X
Parameters Learning												
α **from estimator**	X	X	X	X	X	X	X	X	X	X	X	X

In scoring-based methods, the *scoring metric* guides the search for the structure, and can be composed of local or global metrics. Global metrics can define different ways of model *accuracy estimation*. In constraint-based approaches, the *independence test* defines which test will be used, and *maximum cardinality* restricts the breadth of the search in methods based on conditional independence tests. The *number of parents* determines the maximum number of parents (dependencies) a node can have. The last five lines in the structure learning part of the table show parameters of different search methods: *# of look ahead steps* and *# of operations* are parameters of the LAGD search method, where at each lookahead step the best n operations are considered, and the number of operations defines n. We also have parameters for the *length of the Tabu* list and the *initial temperature* and Δ factor used to update the temperature during Simulated Annealing algorithms. Finally, *iterations* defines the number of executions of iterative search methods.

3.2 Individual Representation and Genetic Operators

After identifying the main components of BNCs, we created an appropriate representation for them. An individual is a real vector with 11 positions, and its values are within the $[0, 1]$ interval. Each individual position represent an algorithm component, and different algorithms might vary significantly in their number of components. Because of that, the individual representation is dynamic, and some individual positions might be non-functional.

The smallest individual is encoded when a Naïve Bayes search is chosen, when we have three active positions in the genome and the others are non-functional (according to Table 1, when a NB search is chosen (gene 1), we use a NB structure (gene 2) and optimize α (gene 3)). The biggest individual is generated by the Tabu Search, when the 11 positions are mapped to different components (see Table 1, column Tabu, for details).

It is important to mention that although the individual phenotype is dynamic, crossover and mutation operations are applied to the individual genotype (real-coded representation) to avoid problems of individuals with different sizes. The uniform crossover builds a uniformly distributed binary mask the size of the individual genotype. A mask value of one (1) means that exchanges will occur in that gene position while a value zero (0) does not modify the content of the corresponding gene. The one-point mutation is applied into one of the 11 possible genes. Each gene has the same probability of being selected, and the value of the selected gene is replaced by a value randomly chosen within its domain (which here varies from zero to one).

3.3 Mapping and Fitness Function

In order to evaluate how effective the generated algorithms are, the classifiers represented by each individual need to be built and run in a dataset to generate a BNC model. Figure 2 shows the whole process of evaluation of a given individual.

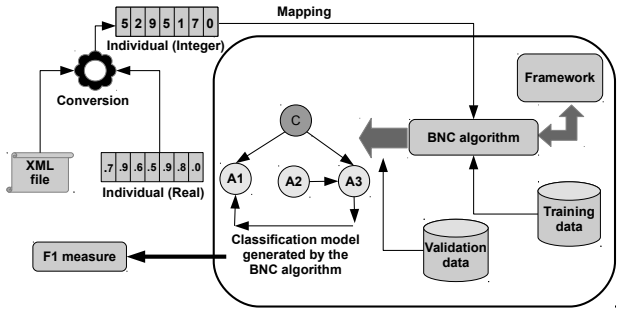

Fig. 2. Evaluation process of one individual.

Initially, all individuals have a real-coded representation, where each chromosome position determines the use of a specific component option. We want to map each position of the individual to a BNC component, using an *XML* file that describes components and their dependencies. For that, the real-coded chromosome is converted into an integer-coded chromosome of the same size. In this conversion, the real number of a gene is multiplied by the maximum number of choices associated with that component, resulting in a rounded integer that indicates a component option in the *XML* file. For example, suppose the real value of the gene representing the search method is 0.35, and we know the number of components for search is equals to 12. In this case, the rounded integer resulting from this combination will be 4, which means the fourth component in the search method components list in the XML file will be selected. When the mapping process finishes, genes not used receive value -1. This process ensures that HHEA will not generate infeasible individuals.

Given the integer-coded individual, its chromosome is mapped to a BNC algorithm. To define the BNC algorithm according to the individual, the frameworks

jBNC [16] and WEKA [21] were used. It is noteworthy the jBNC features and algorithms (inducers) were included in WEKA, generating a robust framework for generating the algorithm.

In the next step, the algorithms built are run in a training set to induce a BNC model, which is then evaluated using a validation set. The fitness function is generated from the validation set, using the F-measure [21].

F-measure is the harmonic mean between precision and recall and is defined in Equation 1. It is an interesting metric because it accounts for different levels of class imbalance, and considers both the precision (which is the number of correctly classified examples over the total number of examples) and recall (which is the number of correctly classified examples in class c over all examples classified as belonging to c, regardless of their real class). In order to prevent overfitting, the training and validation sets are resampled every n generations.

$$F\text{-}measure = \frac{2 \cdot (Precision \cdot Recall)}{(Precision + Recall)} \tag{1}$$

4 Experimental Results

This section presents results obtained when testing HHEA in 15 datasets from the UCI (University of California Irvine) repository [2]. Table 2 shows the datasets and their main characteristics, including number of instances, attributes, classes and missing values.

Table 2. Datasets used in the experiments.

Datasets	#Instances	#Attributes	#Classes	Missing?
Balance Scale	625	4	3	No
Breast Cancer (W)	286	9	2	Yes
Car	1,728	6	4	No
CMC	1,473	9	3	No
Credit (A)	690	14	2	Sim
Diabetes	768	8	2	No
Ecoli	336	7	8	No
Glass	214	9	7	No
Haberman	306	4	2	Não
Heart (C)	303	12	3	Sim
Iris	150	5	3	No
Led Diplay Domain	2,880	7	10	No
Liver Disorders	345	7	2	No
Monks	432	6	2	No
Tic Tac Toe	958	10	2	No

All experiments were executed five times using a 5-fold cross-validation, and two types of experiments were performed. The first compares HHEA with a greedy local search (GS) method to evaluate whether it is a good method to evolve BNC algorithms. GS is a simple method that performs a local search, and works as follows. Taking into account the same BNC components considered by the HHEA, an initial random solution is generated, and for each component

choice, all possible values for that component are tested, and the best one chosen as the most appropriate. In the next iteration, having fixed the previous component, the same procedure is performed for the next one, until a complete solution (BNC) is generated.

The second experiments compare HHEA to three popular BNCs, namely Naïve Bayes (NB), Tree Augmented Naïve Bayes (TAN) and K2. The aforementioned classifiers were chosen because they assume different premises when building the BNC. While Naïve Bayes assumes independence between the attributes (the only relationship considered is among a single attribute and the class attribute), TAN builds a tree to represent relationships between them. K2, in turn, uses a graph to represent the attributes relationships, with no restrictions regarding the BNC structure. For all algorithms, the value of the parameter α was set to 0.5. K2 was configured to use a Bayesian scoring metric and, at most, three parents for each node on the generated BNC.

The parameters of the genetic algorithms were set in preliminary experiments performed using a grid search. The best configuration resulted in the following parameters: 35 individuals evolved for 35 generations, tournament size 2 and crossover and mutation probabilities of 0.9 and 0.1, respectively. The relatively low number of individuals and generations are due to the complexity of the solutions generated. Recall that each individual represents a full BNC algorithm, which will be trained and tested in a given dataset. The training and validation sets were resampled every five generations in order to avoid overfitting. These two sets are merged to form a complete training set when we compared the HHEA to other methods.

We use the statistical approach proposed by Demvšar [9] to compare the results. This approach aims to compare various algorithms in various datasets, relying on the adaptation of the Friedman test with a corresponding (Nemenyi) *post-hoc* test.

Table 3 shows the results of F-measure followed by standard deviations in the 15 selected datasets. Two versions of the HHEA are presented: HHEA represents the method described in Section 3, and HHEA-I (Initialized HHEA) introduces a simple modification to HHEA. In this version, the three state-of-art algorithms (NB, TAN and K2) are included into the initial population of the algorithm, to test whether it can faster improve over these three popular algorithms. Results in bold show the best absolute values obtained for each dataset

First, the *Demvšar* test [9] was performed considering the F-measure obtained by the two versions of HHEA and GS. The critical value of $F(k-1;(k-1)(N-1)) = F(2;28)$ for $\alpha = 0.05$ is 3.340. Since $F_F = 1.809$ and $F_F < F_{0.05}(2,28)$, the null hypothesis of similarity between classifiers is accepted, meaning there is no statistical evidence to show HHEA and GS present different results in terms of F-measure.

We applied the same statistical test to compare the versions of HHEA and the state-of-art methods. The critical value of $F(k-1;(k-1)(N-1)) = F(4;56)$ for $\alpha = 0.05$ is 2.537. Since $F_F = 4.650$ and, consequently, $F_F > F_{0.05}(4;56)$, the null hypothesis is rejected. In this case, we proceed with the Nemenyi *post hoc* test, and conclude that both versions of HHEA and state-of-art methods have similar classification behavior within the 15 datasets and the F-measure for statistical

Table 3. Results of F-measure for HHEA, HHEA-I, GS and selected state-of-art algorithms.

Datasets	HHEA	HHEA-I	GS	NB	TAN	K2
Balance Scale	0.715 (0.046)	0.693 (0.101)	0.691 (0.109)	**0.719 (0.048)**	0.703 (0.055)	0.710 (0.053)
Breast Cancer	0.701 (0.066)	0.701 (0.076)	0.701 (0.090)	**0.733 (0.038)**	0.670 (0.053)	0.719 (0.054)
Car	0.930 (0.086)	**0.972 (0.016)**	0.914 (0.067)	0.849 (0.030)	0.945 (0.019)	0.906 (0.020)
CMC	0.499 (0.041)	0.492 (0.046)	0.475 (0.063)	0.504 (0.044)	**0.507 (0.034)**	0.493 (0.038)
Credit (A)	0.853 (0.021)	0.859 (0.025)	0.847 (0.039)	**0.862 (0.036)**	0.842 (0.025)	0.845 (0.017)
Diabetes	0.733 (0.050)	0.723 (0.069)	0.740 (0.028)	0.737 (0.030)	**0.747 (0.020)**	0.741 (0.024)
Ecoli	0.777 (0.045)	0.801 (0.022)	0.782 (0.026)	**0.812 (0.014)**	0.798 (0.019)	0.807 (0.014)
Glass	0.599 (0.101)	0.636 (0.122)	0.600 (0.104)	**0.685 (0.040)**	0.658 (0.084)	0.682 (0.034)
Haberman	0.642 (0.087)	0.664 (0.092)	0.660 (0.091)	**0.679 (0.080)**	0.678 (0.122)	**0.679 (0.080)**
Heart (C)	0.823 (0.032)	0.824 (0.031)	0.817 (0.067)	**0.836 (0.038)**	0.835 (0.020)	**0.836 (0.024)**
Iris	0.921 (0.050)	0.932 (0.052)	**0.939 (0.033)**	0.932 (0.030)	0.926 (0.031)	0.926 (0.031)
Led	**0.735 (0.022)**	0.703 (0.145)	0.732 (0.023)	0.732 (0.023)	**0.735 (0.026)**	0.732 (0.023)
Liver Disorders	0.447 (0.103)	0.447 (0.103)	0.442 (0.101)	0.447 (0.113)	0.447 (0.113)	0.447 (0.113)
Monks	**0.352 (0.039)**	0.351 (0.040)	0.344 (0.046)	0.254 (0.051)	0.337 (0.024)	**0.352 (0.039)**
Tic Tac Toe	0.700 (0.020)	0.692 (0.045)	0.700 (0.020)	0.700 (0.022)	0.700 (0.022)	0.700 (0.022)

evaluation. Note that the data resampling applied every five generations can affect HHEA-I results, despite the use of elitism.

The comparisons made so far consider the average results of 25 different algorithms (5 executions x 5 folds) tested over 15 different datasets. Next, we compare results produced by a single BNC algorithm, which should selected among the 25 to be used in a practical situation. We selected the algorithm with the best fitness value found in the validation set, used during the fitness calculation process, and the results are presented in Table 4.

Table 4. Solutions found by one algorithm produced by HHEA and HHEA-I when compared to three state of the art methods.

Base de Dados	HHEA	HHEA-I	NB	TAN	K2
Balance Scale	**0.729 (0.051)**	0.724 (0.046)	0.719 (0.048)	0.703 (0.055)	0.710 (0.053)
Breast Cancer	0.730 (0.051)	**0.733 (0.053)**	**0.733 (0.038)**	0.670 (0.053)	0.719 (0.054)
Car	0.979 (0.009)	**0.983 (0.009)**	0.849 (0.030)	0.945 (0.019)	0.906 (0.020)
CMC	**0.521 (0.038)**	0.517 (0.037)	0.504 (0.044)	0.507 (0.034)	0.493 (0.038)
Credit (A)	0.875 (0.012)	**0.879 (0.017)**	0.862 (0.036)	0.842 (0.025)	0.845 (0.017)
Diabetes	0.754 (0.020)	**0.759 (0.032)**	0.737 (0.030)	0.747 (0.020)	0.741 (0.024)
Ecoli	0.809 (0.041)	**0.821 (0.004)**	0.812 (0.014)	0.798 (0.019)	0.807 (0.014)
Glass	0.665 (0.107)	**0.724 (0.030)**	0.685 (0.040)	0.658 (0.084)	0.682 (0.034)
Haberman	0.676 (0.066)	**0.694 (0.095)**	0.679 (0.080)	0.678 (0.122)	0.679 (0.080)
Heart (C)	**0.854 (0.019)**	0.848 (0.023)	0.836 (0.038)	0.835 (0.020)	0.836 (0.024)
Iris	0.955 (0.017)	**0.962 (0.027)**	0.932 (0.030)	0.926 (0.031)	0.926 (0.031)
Led	**0.741 (0.021)**	0.740 (0.026)	0.732 (0.023)	0.735 (0.026)	0.732 (0.023)
Liver Disorders	0.447 (0.113)	0.447 (0.113)	0.447 (0.113)	0.447 (0.113)	0.447 (0.113)
Monks	0.386 (0.049)	**0.400 (0.021)**	0.254 (0.051)	0.337 (0.024)	0.352 (0.039)
Tic Tac Toe	0.700 (0.022)	0.700 (0.022)	0.700 (0.022)	0.700 (0.022)	0.700 (0.022)

The results are again compared using the same methodology described above. The critical value of $F(k-1;(k-1)(N-1)) = F(4;56)$ for $\alpha = 0.05$ is 2.537. With $F_F > F_{0,05}(4;56)$, the null hypothesis is rejected, and the Nemenyi test performed. The critical difference for this test is given by 1.575. Comparing the results by pairs, we conclude that HHEA is statistically better then TAN and no statistical difference between NB and K2 is found. HHEA-I, in turn, is statistically better than TAN and K2, but no evidence of difference between NB is found. As the methods generated vary a lot among each other, and the

search space is huge, isolated analysis of the algorithms is essential, and there are always solution better than those state of the art.

We evaluated the components present in the BNC algorithms produced for the *Car* dataset (Table 3). The generated BNCs use, in their majority, a Hill Climbing or Repeated Hill Climbing search method, combined with a global score function. Furthermore, we observed that the algorithms that impose fewer restrictions to the BNC (allowing, for example, a greater number of parents), perform feature selection and use lower α values for the parameter estimator (between 0.0 and 2.0) have better classification performance. This is an evidence that certain components should be prioritized during the search.

Regarding computational time, we cannot forget each individual is a classification algorithm, executed in a dataset to assess its fitness value. Some of the algorithms might execute a simulated annealing algorithm to find the BNC structure, while others might perform a greedy search. Hence, the running time of the HHEA depends on the BNC algorithms generated. In order to give the reader an idea of the real execution time, the fastest HHEA execution was in the *Haberman* dataset, where it took 118.5 seconds to run for all partitions of the 5-fold cross-validation. On the other hand, *Credit (A)* took the longest to run, with an average time of 16,114 seconds to execute in all folds.

5 Conclusions and Future Work

This work proposed a hyper-heuristic evolutionary algorithm to automatically evolve BNCs. The algorithm receives as input a dataset, and outputs a BNC tailored to that dataset. BNC algorithms are generated from a predefined set of components, which were identified analysing a set of BNC algorithms. By producing algorithms personalized to a dataset, we automate the process of choosing which algorithm is more suitable for that domain.

The method was tested in 15 UCI datasets, and the results showed that, in average, the values of F-measure obtained by HHEA are equivalent to those obtained by the algorithms already proposed in the literature. However, looking carefully at each evolved algorithm in isolation, we observe they can produce BNC algorithms very different from the state of the art, and with F-measures equivalent or better.

The next step for this work is to test the algorithm in other real world datasets. We known the UCI datasets are a great first testbed, but as many algorithm already proposed were fine tuned over them, improving their results with automatically generated algorithms might prove difficult. The use of the algorithm is much more productive in unknown scenarios. Furthermore, it would be interesting to look not only at the accuracy of the algorithms evolved, but also to the simplicity of the models they generate, as to a specialist simple models are more valuable than complex ones.

Acknowledgments

This work was partially supported by the following Brazilian Research Support Agencies: CNPq, CAPES and FAPEMIG.

References

1. M. Abramovici, M. Neubach, M. Fathi, and A. Holland. Competing fusion for Bayesian applications. In *Proc. of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 378–385, 2008.
2. A. Asuncion and D. Newman. UCI machine learning repository, 2007.
3. 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 (MIT)*, 21(4):659–684, 2013.
4. R. Bouckaert. *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, 1995.
5. P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer, 2008.
6. J. Cheng and R. Greiner. Learning Bayesian belief network classifiers: Algorithms and system. In *Proc. of the Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pages 141 – 151, 2001.
7. G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.
8. R. Daly, Q. Shen, and S. Aitken. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26(2):99 – 157, 2011.
9. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7(Jan):1–30, 2006.
10. D. Floreano, P. Durr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47 – 62, 2008.
11. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine*, 29(2 - 3):131 – 163, Nov. 1997.
12. A. S. Hesar, H. Tabatabaee, and M. Jalali. Structure learning of Bayesian networks using heuristic methods. In *Proc. of International Conference on Information and Knowledge Management (ICIKM 2012)*, 2012.
13. G. Pappa, G. Ochoa, M. Hyde, 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(1):3–35, 2014.
14. G. L. Pappa and A. A. Freitas. *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer, 2009.
15. A. G. C. Sá and G. L. Pappa. Towards a method for automatically evolving bayesian network classifiers. In *Proc. of the Conference Companion on Genetic and Evolutionary Computation Conference Companion*, pages 1505–1512, 2013.
16. J. P. Sacha. *New synthesis of bayesian network classifiers and cardiac spect image interpretation*. PhD thesis, 1999.
17. K. M. Salama and A. A. Freitas. Extending the ABC-Miner Bayesian classification algorithm. In *Proceeding of the Workshop on Nature Inspired Cooperative Strategies for Optimization*, pages 1–12, 2013.
18. K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
19. C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD*, pages 847–855, 2013.
20. T. Verma and J. Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proc. of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 323–330, 1992.
21. I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 2011.
22. X. Yao. Evolving artificial neural networks. *Proc. IEEE*, 87(9):1423–1447, 1999.