Towards a Method for Automatically Evolving Bayesian Network Classifiers

Alex G. C. de Sá
Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil
alexgcsa@dcc.ufmg.br

Gisele L. Pappa
Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil
glpappa@dcc.ufmg.br

ABSTRACT

When faced with a new machine learning problem, selecting which classifier is the best to perform the task at hand is a very hard problem. Most solutions proposed in the literature are based on meta-learning, and use meta-data about the problem to recommend an effective algorithm to solve the task. This paper proposes a new approach to this problem: to build an algorithm tailored to the application problem at hand. More specifically, we propose an evolutionary algorithm (EA) to automatically evolve Bayesian Network Classifiers (BNCs). The method receives as input a list of the main components of BNC algorithms, and uses an EA to encode these components. Given an input dataset, the method tests different combinations of components to that specific application domain. The method was tested in 10 UCI datasets, and compared to three classical BNCs and a greedy search algorithm. Results show that the current algorithms can indeed be improved, but that the EA is currently outperformed by the greedy search.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Bayesian Network Classifiers, Evolutionary Algorithms, Automatic Design

1. INTRODUCTION

Bayesian Networks (BNs) are powerful tools to knowledge representation and inference under conditions of uncertainty [8]. They usually represent data using a direct acyclic graph, where each node represents an attribute and edges represent probabilistic dependencies among attributes. BNs were first used in the context of classification with Naïve Bayes, which is considered a BN where all attributes are independent given the class node.

In contrast with Naïve Bayes, most BNs compute the conditional probability of one node given the values assigned to others (i.e., nodes are not independent), and can be used as classifiers that give the posterior probability distribution of the class node given the values of other attributes. Hence, after Naïve Bayes, many other Bayesian Network Classifiers (BNCs) were proposed in the literature, creating models represented by trees, forests and graphs [15, 23, 24].

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

Although many adaptations of classical BN algorithms for inference were made to work with a special attribute – i.e., the class attribute, there are no studies showing the advantages of one representation or algorithms over the other. This is the main goal of this paper: to select the best BNC components to a target dataset using an evolutionary algorithm (EA).

Different from traditional meta-learning algorithms [5], which focus on algorithms recommendation/selection, the approach proposed here performs generative meta-learning [3, 21]. In this case, instead of *selecting* an appropriate algorithm, the meta-learner *builds* a classifier targeting the dataset at hand. Figure 1 shows the method in a high-level of abstraction. It is important to note that the output of the EA is a BNC *algorithm*, which in turn can generate a BN to any dataset, although it is built tailored to the input dataset. Previously proposed approaches, in contrast, output the *BN* straight away, as showed by the dotted line in Figure 1.

The proposed method works as follows. It receives a list of the main components of BNC algorithms, and uses an EA to encode these components. Given an input dataset, the method tests different combinations of components to that specific application domain. 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 a measure which takes into account precision and recall used as a fitness function. At the end of the evolutionary process, the best algorithm is tested in data from the same application domain.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

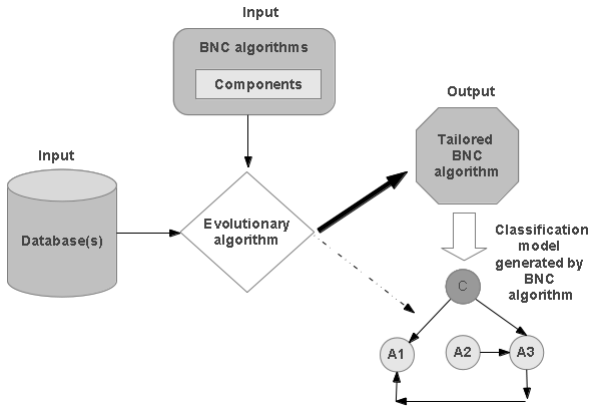


Figure 1: Method proposed to automatically evolve Bayesian Network Classifiers.

The algorithm was tested in 10 different datasets from different domains extracted from the UCI Repository [1], and compared to three popular BNCs: Naïve Bayes [18], Tree-Augmented Naïve Bayes (TAN) [15] and K2 [10]. The performance of the GA is also tested with different numbers of solution evaluation, and compared to a greedy search algorithm. Results showed that there is room for improvement in BNCs even with the current set of components, which is preliminary and can be extended with new components. In particular, the GA obtained results equal or better than the other algorithms in the great majority of datasets. However, a comparison with a greedy search showed that it currently outperforms the GA.

The reminder of this paper is organized as follows. Section 2 reviews related work in the areas of BNC and automatic evolution of algorithms. 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

This section is divided in two parts. The first part describes the basic concepts of BNs and BNCs, and discusses different types of BNCs present in the literature. The second part refers the reader to other works that show how EAs were previously used in BNs models, and then reviews a set of methods proposed to evolve classification algorithms different from BNCs.

2.1 Bayesian Networks

A BN is a robust tool to describe/represent knowledge and draw conclusions about their properties, while taking into account conditions of uncertainty [8]. It is represented by a directed acyclic graph (DAG) with a conditional probability table for each node, where the tables are considered as BN parameters. The BN structure contains nodes representing the domain of the variables and arcs between these nodes constituting their probabilistic dependencies. In the construction of BNs from datasets, nodes represent dataset attributes.

There are many advantages in using BNs to model data [16]. First, a BN is a model that encodes the dependencies among all variables and is ready to deal with possible lack of data. Second, it can be used to learn causal relationships and, therefore, gain information about the problem domain and predict the consequences of events. Third, because it has causal semantics, BN considers the encoding of causal prior knowledge in a straightforward manner. As it also defines the strength of causal relationships as probabilities, a BN is an ideal representation for combining prior knowledge and data. Finally, Bayesian statistical methods in conjunction with BNs provide an efficient approach to avoid data overfitting.

This paper focuses on the classification task, where BNs can be adapted to build BNCs [24]. In this case, the connections among attributes are different from those in simple BNs, depending largely on the class attribute. A BNC learning algorithm will find relations among the n attributes of the dataset $(A_1, ..., A_n)$, and causal relationships between this set of attributes and the class attribute C.

Both BNs and BNCs are created in two main phases: the definition of the network structure and the definition of the parameters (probability tables). The main difference between BNs and BNCs is that algorithms for learning the structure of general BNs do not consider as part of the process a special node, namely the class node. However, general BN learning algorithms can be modified to generate classifiers, but this is not always an easy task. For more information on general learning of BNs, see [12].

BNs were not considered classifiers until the appearance of Naïve Bayes (NB), a simple kind of BNC that assumes the dataset attributes are independent given the class node [7]. A full description of the NB is present in Langley *et al.* [18], which shows that NB obtains surprisingly effective results (even considering independence among the attributes) when compared to more sophisticated classification methods. In contrast, what makes BNCs important for learning when compared to other methods, such as Artificial Neural Networks, K-Nearest Neighbours and Decision Trees, is exactly the fact that they take into account the dependencies (influences) between pairs of attributes.

The process of learning the structure of BNs is based on two main approaches: scoring-based and constraint-based. In the first case, the idea is to guide the search algorithm that builds the structure of the network using a scoring measure, such as the minimum description length [25]. On the other hand, constraint-based algorithms have as their major component a conditional independence test. The test is used to analyse/explore the conditional dependence relationships between variables, and uses these relationships as constraints to build the BN. Examples of these tests are the χ^2 and mutual information test [7].

Regardless of the approach followed, depending on the type of relationship defined among the data attributes, BNCs can be represented by graphs, trees or independent variables. Friedman *et al.* [15] introduces the Tree-Augmented Naïve Bayes (TAN) classifier. TAN assumes that the class variable has no causal predecessors (parents), and that each attribute must have the class node and at most another attribute as its predecessors. Friedman *et al.* used modifications of the work of Chow and Liu [9] in order to map the causal relationships between attributes of a dataset, and the resulting BN is a tree.

In contrast, Cooper and Herskovits [10] proposed K2, one of the most popular BNCs. K2 is actually an extension of the Kulató system, proposed by the same authors in [17].

Kulató takes as input a dataset and a causal ordering of the variables, and generates as output the BN structure. It is a scoring-based method guided by entropy, which initially assumes that all variables are marginally independent. Then, the system performs a greedy search, and incrementally adds directed edges between the nodes. The search goes on while the BN entropy decreases. The main difference between Kulató and K2 is the scoring metric, which was replaced by the Bayesian score, which works by maximizing the probability of the resulting structure. K2 was developed for learning general BNs, but with a simple change in its features, it can be converted into a classifier.

The BNCs described in the preceding paragraphs are of great importance for this paper, since they have been chosen as baselines to the proposed approach. However, the works of Cheng and Greiner [7, 8], Sacha *et al.* [23] and Santos *et al.* [24] also discuss other important BNCs that serve as basis for this work.

2.2 Evolutionary Algorithms and BNCs

Before reviewing the main works proposed for the evolution of classifiers, it is important to recall that the method proposed here differs significantly from others already proposed to build a BNC instead of a BNC *algorithm*. More information on the former can be found in [19], which reviews how previous work with EAs have solved problems such as BN structure search, feature selection and searching the causal ordering of the variables.

This section focuses on other methods already proposed to automatically generate algorithms. It is important to point out that the automatic generation of algorithms can be viewed as a third type of meta-learning. The area of meta-learning has emerged to recommend the best classification method to a given (or a set of) dataset(s) [26]. There are basically two kinds of meta-learning techniques: algorithms selection and combination of models. In the first case, the algorithm selects a learning algorithm to a dataset according to meta-data characteristics, such as number of instances, classes, among many others. The combination of models, in turn, explores models coming from different algorithms with different biases or models built from different subsets of data [6, 14, 29].

The third type of meta-learning, generation of algorithms, can be performed following two main approaches: generation by selection and generation by construction. While generation by selection chooses existing components of potentially different methods and combines them, generating an algorithm, generation by construction combines these components with algorithm primitives (loops, conditional, etc.) and components that have not been tested yet for constructing new learning methods. The method proposed here and the work of Floreano *et al.* (Artificial Neural Networks) [13] are examples of generation by selection, while the works of Pappa and Freitas (Rule Induction Algorithms) [21], Lourenço *et al.* (Evolutionary Algorithms) [20] and Barros *et al.* (Decision Tree Algorithms) [2, 3] are examples of generation by construction.

3. AN APPROACH FOR EVOLVING BAYESIAN NETWORK CLASSIFIERS

This section presents the proposed approach to automatically evolve BNCs. As others works already proposed in the literature, the idea here is to generate BNCs tailored to specific datasets, instead of performing experiments for selecting one among various methods and testing different parameter configurations. The motivation for the study of the evolution of BNC algorithms is that, in addition to the advantages described in Section 2, they are represented by a model visually understandable and interpretable, in the same way of decision trees or rule induction algorithms.

The proposed method has two main components: (i) a set of BNC related components and (ii) a search method to explore different combinations of these components. Figure 1 shows the process followed by the method. It receives as input a dataset and, considering all the components identified from BNCs and the search method, returns a BNC tailored to the input dataset. The returned BNC will be suitable for the domain of the input data, but that does not mean it cannot generalize to other datasets.

Here we use a integer-coded genetic algorithm (GA) to search the space of BNCs, but also experiment with a greedy search method. In the GA, each individual represents a BNC algorithm (see Section 3.1), randomly generated from a combination of the available components. During the evaluation of the individuals, a mapping between the individual and a BNC algorithm, implemented in the frameworks Weka [27] or jBNC [22, 23] is performed (see Section 3.2).

Following, the individuals undergo uniform crossover and one-point mutation operations to generate a new population. After a predefined number of generations, the best BNC generated is returned and run in a test set coming from the same application domain.

3.1 Individual Representation

As already mentioned, one of the most important steps in this research is to identify a list of relevant components the search algorithm (in this case, a GA) should explore to build a solution (BNC algorithm) tailored to a specific dataset. After studying the algorithms (classifiers) discussed in the previous section, seven major components were identified and incorporated into the GA individual representation. Figure 2 shows the individual representation and the range of possible values for each position. The genes and their allowed values are detailed bellow:

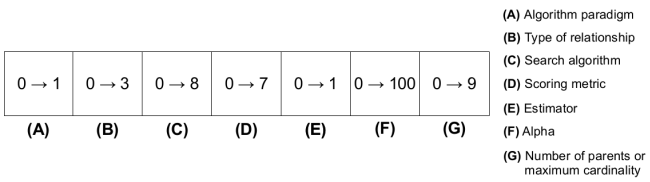


Figure 2: Individual representation for the GA.

1. **Algorithm paradigm:** Defines if the algorithm uses a search method and a scoring metric to create the structure of the BN (scoring-based approach) or examines the conditional dependencies to use them as constrains for the search method (constraint-based approach). In the GA, this component can have values 0 or 1.

2. **Type of relationship among the attributes**: Specifies the type of relationship among the attributes of

the dataset. It may be none (Naïve Bayes), a tree (TAN and others), a forest (Forest-Augmented Naïve Bayes and others) or a graph (K2 and others). The four options just mentioned can be assumed by the second gene of the chromosome.

3. **Search algorithm to build the BNC structure**: Determines the procedure that builds the network structure, and can represent any traditional search method, such as a greedy search, simulated annealing, among others. However, the required functionalities of the search method depend heavily on the type of relationship among the attributes (item 2). The main differences among the search methods for difference structures are in the way the arcs (between two nodes) are added, removed and reversed, and whether the methods are local or global.

   - **No relationship among the attributes:** There is no need for a search method in this case, as a fixed structure with no arcs among attributes is defined and used by Naïve Bayes. In this case, different versions of Naïve Bayes can be used, differing from one another on the way they model numeric attributes. Three different options can be used for numerical attributes: normal distribution, kernel density estimator, and supervised discretization [4, 28].
   - **Relationship represented by a tree:** Can use local and global search methods. The most popular search method in this class is based on TAN, and is a modification of the Chow and Liu algorithm [9] that uses the concept of maximum weight spanning tree together with Kruskal's algorithm [11]. Variations include STAN (Selective Tree-Augmented Naïve Bayes) and STAND (Selective Tree-Augmented Naïve Bayes with Discarding), which vary according to the operators that manipulate the graph nodes.
   - **Relationship represented by a forest:** The scope of the forest algorithms is more restricted. They consider three search algorithms also based on Kruskal's algorithm. These algorithms were used by FAN (Forest-Augmented Naïve Bayes), SFAN (Selective Forest-Augmented Naïve Bayes) and SFAND (Selective Forest-Augmented Naïve Bayes with Discarding), and differ on the types of operators used to cut tree edges in order to generate forests [22, 23].
   - **Relationships in a graph form:** This kind of relationship generates nine algorithms (maximum range of the chromosome). The scoring-based approaches were inserted in the GA space search. The search methods are: Hill Climbing (local and global), K2 (local and global), Tabu Search (local and global), Repeated Hill Climbing (local and global) and LAGD Hill Climbing (just local). Two Constraint-based approaches were also considered: Conditional Independence (CI) Search Algorithm and ICS Search Algorithm. For more information on these methods, see [4, 28].

4. **Scoring metric:** Criteria for evaluating the structure of the BNC. It may not exist in constraint-based algorithms (NB), and can assume at most eight different values. The metrics are usually local or global, and hence combined with local or global search methods. In contrast with local metrics, global metrics cannot decompose the scores of the individual nodes from the BNC structure, and the whole structure must be considered to get a score.

   For global search methods, a cross-validation strategy is used to obtain a score based on metrics of accuracy of the model. Three variations of cross-validation can be used: k-Fold-CV (k-Fold Cross-Validation), LOO-CV (Leave One Out Cross-Validation) or Cumulative-CV (Cumulative Cross-Validation) [4]. Additionally, two other metrics can be useful in this context: HGC (Heckerman-Geiger-Chickering) and SB (Standard Bayesian) [22].

   The local search methods, in turn, allow eight different scoring metrics: Bayes, Entropy, BDeu, MDL (Minimum Description Length), AIC (Akaike Information Criterion), LC (Local Criterion), which can also be combined with two different cross-validation procedures, namely LOO (Leave-one-out validation) and $CV_{i,j}$ (i-Folds j-Times Cross-Validation) – see [4, 22, 28] for details.

5. **Estimator:** The estimator is used to define the tables of conditional probabilities of a BN once the structure has been learned. It estimates the probabilities directly from data, and also depends on the type of algorithm being used, being absent in some BNCs. Currently, the GA only uses one type of estimator (Simple Estimator from Weka) as they are not easily adapted from BNs to BNCs. Future versions of the system will focus on this component.

6. **Alpha value (Estimator):** Alpha is a parameter of the estimator, and can be interpreted as the initial count for each probability value. In the GA, its value starts at 0.00 and goes up to 25.00 in steps of 0.25, assuming 100 different values. We chose not to leave this attribute continuous to be able to perform a brute force procedure, as detailed in Section 4.

7. **Number of parents or maximum cardinality:** The last position of the genome has two different meanings depending on the type of classifier (item 1) being used. The maximum number of parents of a node is a parameter of scoring-based algorithms, and is restricted to the type of relationship allowed among the attributes (item 2). Recall that the parents of a node causally influence the node. Constraint-based algorithms, in contrast, do not require a number of parents but a maximum cardinality. The idea of this class of algorithm is to test whether each pair of variables (or attributes) of the dataset, like x and y, are conditionally independent given a set of variables Z. Z is defined by the subset of nodes that are neighbors of both x and y in the causal graph. If the search method used to build the network identifies an independency $(x, y|Z)$, the edge between x and y is removed. The maximum cardinality determines the largest subset of Z to be considered in conditional independence tests

$(x, y|Z)$. For both types of algorithms, this component can assume values varying from 0 (not present) to the maximum number of attributes of the dataset. As here the biggest dataset has 9 attributes (plus class), nine is the maximum number of parents of a node.

3.2 Fitness

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 3 shows the process of evaluation of a given individual.

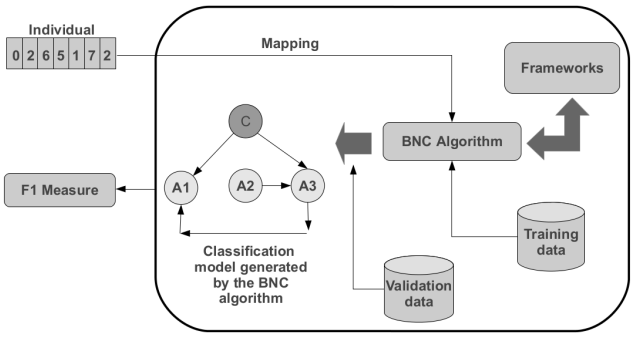


Figure 3: Evaluation process of one individual.

Initially, each individual is mapped to a BNC algorithm using two different frameworks that deal with BNC algorithms: Weka and jBNC. 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 F1 measure [28].

F1 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 in the dataset) 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 set is resampled every n generations.

$$F1 = \frac{2 \cdot (Precision \cdot Recall)}{(Precision + Recall)} \quad (1)$$

4. EXPERIMENTAL RESULTS

This section presents results of evolving BNCs with GAs. As previously mentioned, the first version of the method was simplified to give us a better understanding of the problem. The method was tested in 10 datasets from the UCI (University of California Irvine) repository [1]. Table 1 shows the datasets and their main characteristics, including number of instances, attributes and classes.

The parameters of the genetic algorithms were set in preliminary experiments, and are shown in Table 2. 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, which will be trained and tested in a given dataset. Recall that the training and validation sets were resampled every 10 generations in order to avoid overfitting.

Table 1: Datasets used in the experiments.

Dataset	# inst.	Attributes		#Class
		Type	Number	
Balance-scale	625	Integer	5	3
Car	1728	Nominal	7	4
Diabetes	768	Real	9	2
Ecoli	336	Real	8	8
Glass	214	Real	10	7
Haberman	306	Integer	4	2
Iris	150	Real	5	3
Led7	2880	Binary	8	10
Liver-disorders	345	Integer/Real	7	2
Tic-tac-toe	958	Discrete	10	2

Table 2: Parameters of the Genetic Algorithm.

Parameter	Value
Number of Generations	50
Number of Individuals	60
Tournament size	5
Crossover probability	0.90
Mutation probability	0.10

The method proposed here follows a construction by selection approach, and hence the search space does not bring many possibilities of completely new algorithms, but rather combinations of components already present in other algorithms. The problem was modeled in a way that a brute force search could be performed in the components space, allowing us to compare the results of traditional BNC algorithms with the ones produced by the proposed method and the brute force.

All experiments were executed using a 5-fold cross-validation procedure, and compared to three popular BNCs, namely Naïve Bayes, Tree Augmented Naïve Bayes (TAN) and K2 in terms of F1. The aforementioned classifiers were chosen because they assume different premisses when building the BN. 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 both algorithms, the value of the parameter α was set to 0.5. The results of the GA are always averages over 10 executions.

Table 3 shows the results of F1 followed by standard deviations. Two versions of the GA are presented: GA represents the method described in Section 3, and GA-I (Initialized GA) introduces a simple modification to GA. In this version, the three baselines are included into the initial population of the algorithm, to test whether it can faster improve over the well-known popular algorithms. Results were compared using a paired t-test with confidence level 95%, and performed in two steps. First, the results among the two versions of the GA are compared. Second, the results among the GA and the three other baselines are considered. For both comparisons, ▲ denotes a statistically significant positive variation for the method in that column, and ▼ a statistically significant negative variation according to the t-test.

Let us first consider the two versions of the GA. Note that in 8 out of 10 cases the GA finds results as good as the

Table 3: Comparisons of two versions of the GA with the selected baselines.

Dataset	GA	GA-I	NB	TAN	K2
Balance-Scale	0.71 (0.05) ▼	0.87 (0.03)	0.87 (0.04)	0.70 (0.055) ▼	0.72 (0.05) ▼
Car	0.97 (0.02)	0.97 (0.02)	0.85 (0.03) ▼	0.94 (0.016) ▼	0.85 (0.03) ▼
Diabetes	0.73 (0.02)	0.74 (0.03)	0.75 (0.03)	0.75 (0.02) ▲	0.74 (0.03)
Ecoli	0.79 (0.05)	0.83 (0.04)	0.85 (0.04)	0.82 (0.03)	0.82 (0.04)
Glass	0.64 (0.07)	0.63 (0.07)	0.40 (0.06) ▼	0.70 (0.02)	0.68 (0.04) ▲
Haberman	0.66 (0.10)	0.67 (0.08)	0.71 (0.075)	0.679 (0.08)	0.68 (0.08)
Iris	0.93 (0.04)	0.93 (0.03)	0.95 (0.04)	0.932 (0.03)	0.93 (0.03)
Led7	0.73 (0.02)	0.73 (0.02)	0.73 (0.02)	0.736 (0.03)	0.73 (0.02)
Liver-Disorders	0.45 (0.10) ▼	0.59 (0.11)	0.56 (0.07)	0.45 (0.11) ▼	0.45 (0.11) ▼
Tic-Tac-Toe	0.70 (0.04)	0.69 (0.03)	0.65 (0.03) ▼	0.70 (0.02)	0.70 (0.21)

ones found by GA-I, but the latter converges faster. The two datasets where there is a significant difference among the two versions are *balance scale* and *liver disorder*. *Liver disorder* is a special dataset, with classes represented by only two examples, reflected in the large value of the standard deviation. Note that, in both cases, the results obtained by GA-I are the same as those produced by Naïve Bayes. The dataset that presents the results we are really looking for is *car*, where the results found by both versions of the GA are better than those obtained by the other algorithms.

Considering the results obtained by the three baselines, NB presents statistically significant worse results than GA in three datasets (see symbol ▼ in Table 3, column NB), and TAN and K2 in other two. In total, the baselines were worse than the GA in 9 cases, and better in two. These two cases correspond to TAN in *diabetes* and K2 in *glass*. This means that, for these datasets, GA-I lost the initial solutions representing these algorithms during the search process. In all other cases the results do not present statistical differences.

In a second experiment, we compare the performance of GA-I with the one produced by a greedy search (GS) algorithm, a simpler method that performs a local search. The GS works as follows: a solution is randomly generated, and for the first position of the individual, all possible values of that component are tested and the best chosen as the most appropriate one. In the next steps, the same procedure is performed for the next positions, keeping the values of components already searched with the best values found by the GS. The greedy search was also executed 10 times. The results are reported in Table 4, where again the symbols ▲ and ▼ represent a statistically significant positive/negative variation. Note that, in this case, the greedy search presented better results than the GA in 7 out of 10 cases, being only worse in the dataset *car*. Comparing the same results with GA-I, it is worse than the GS in 4 cases, and better in 3 cases. However, note that this last comparison is not fair, as the GS was not initialized with the three well-known algorithms.

Finally, we compared the results obtained by the two different search methods (GA and GS) with a brute force approach. Note that current search space allows 44,653 solutions, while the other methods are performing 1,200 evaluations. The results are presented in Table 5. The second column shows the BF values for F1, while in the following column a ▼ represents a statistically significant negative variation and ♦ indicates that no statistical difference was found. It is interesting to note that only 10 out of 50 results presented no statistical difference to the BF, while in all other 40 cases the brute force was better. This results indicate that the algorithms currently used can be improved by finding different combinations of different components, and we need to develop a search method capable of doing that. Note that the GS obtained the best possible result in the dataset *tic-tac-toe*, where the algorithms TAN and K2 also had a good performance. In the other 5 cases, NB obtained the best results as the BF, considering total independence between the variables.

Table 4: Greedy search (GS) versus the two versions of the proposed GA.

Dataset	GS	GA	GA-I
Balance-Scale	0.73 (0.05)	0.71 (0.05) ▼	0.87 (0.03) ▲
Car	0.91 (0.07)	0.97 (0.02) ▲	0.97 (0.02) ▲
Diabetes	0.75 (0.03)	0.73 (0.02) ▼	0.74 (0.03) ▼
Ecoli	0.81 (0.04)	0.79 (0.05) ▼	0.83 (0.04)
Glass	0.70 (0.04)	0.64 (0.07) ▼	0.63 (0.07) ▼
Haberman	0.70 (0.09)	0.66 (0.10) ▼	0.67 (0.08) ▼
Iris	0.95 (0.03)	0.93 (0.04) ▼	0.93 (0.03)
Led7	0.74 (0.02)	0.73 (0.02) ▼	0.73 (0.02) ▼
Liver-Disorders	0.45 (0.10)	0.45 (0.10)	0.59 (0.11) ▲
Tic-Tac-Toe	0.70 (0.02)	0.70 (0.04)	0.692 (0.03)

Table 5: Brute Force (BF) versus GA-I, Greedy (GS) and the three baselines.

Dataset	BF	GA-I	GS	NB	TAN	K2
Balance-Scale	0.88 (0.03)	♦	▼	♦	▼	▼
Car	0.98 (0.01)	▼	▼	▼	▼	▼
Diabetes	0.78 (0.03)	▼	▼	♦	▼	▼
Ecoli	0.89 (0.02)	▼	▼	♦	▼	▼
Glass	0.76 (0.02)	▼	▼	▼	▼	▼
Haberman	0.74 (0.07)	▼	▼	♦	▼	▼
Iris	0.99 (0.02)	▼	▼	▼	▼	▼
Led7	0.75 (0.02)	▼	▼	▼	▼	▼
Liver-Disorders	0.66 (0.05)	♦	▼	♦	▼	▼
Tic-Tac-Toe	0.70 (0.03)	▼	♦	▼	♦	♦

Considering the results obtained by the GA when compared to the greedy-search, we reduced the number of evaluations of the GA from 3,000 to 1,500, 750 and 400, respectively. We report the results for 5 of the 10 initial datasets. The datasets selected were those where the GA was better than the GS (namely *balance-scale* and *liver-disorder*), as good as the GS (*ecoli* and *iris*) and one case where the GA was worst (*diabetes*). The results obtained are showed in Figure 4. For all datasets except *liver disorder* reducing the number of evaluations drastically from 3,000 to 400 did not change the results. While for *liver disorder* it is worth using

3,000 evaluations, for the other datasets 400 evaluations are enough.

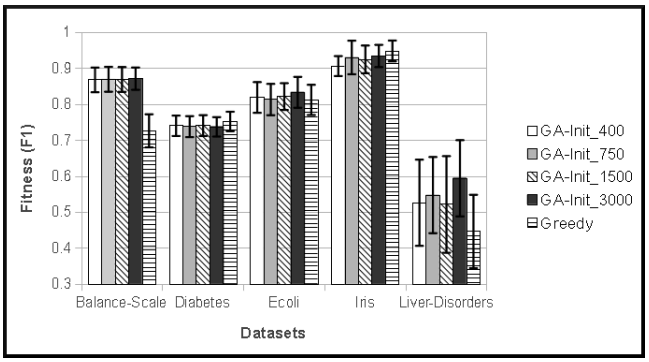


Figure 4: Varying the number of fitness evaluations for the GA.

Figures 5(a) and 5(b) illustrate the fitness convergence of the best individual, worst individual and the population average. Note that the values of fitness can fall abruptly from one generation from another due to training set resampling, but the values are quickly recovered. However, notice also that the range of fitness values the algorithm can obtain here is not huge, as we are not using any components that would not make sense to combine and generate "bad" algorithms. Perhaps a local search method would be useful to exploit the search space after a first general exploration is performed.

In terms of computational time, the smallest dataset (*iris*) took 45 hours to run for all 5 partitions of the 5-fold cross-validation. The largest datasets, namely *led7* and *car*, took in average 19.6 and 39.7 hours to execute for each data partition.

5. CONCLUSIONS AND FUTURE WORK

This work proposed the first version of a method to automatically evolve Bayesian Network Classifiers. The method is based on a GA, where each individual represents a set of seven main components identified in the BNCs. The current search space of the method is not huge, but that was essential to understand the nature of the problem begin dealt with.

The method was evaluated in a set of 10 UCI datasets, and compared with those obtained by the GA and three other common used BNCs, namely Naïve Bayes, Tree Augmented Naïve Bayes (TAN) and K2. A greedy search algorithm was also implemented, and its results compared to the GA and a brute force procedure run to serve as a optimum solution to the problem being handled. Results showed that the GA could generate results comparable to or better than those obtained by the baselines, but that the greedy search could find better solutions with a smaller number of evaluations. From the results obtained we also observed that the three baselines presented worse results than the brute force in the majority of cases, showing there is room for improvement even without adding more components to the search space.

As future work, the number of components and their functionalities will be improved. Our main objective is to change the building approach from selection to construction. In this case, taking into account the characteristics of the datasets at hand to generate the best algorithm, or the state of the search algorithm can be very beneficial for the methods being proposed. In the case of the BNCs, the correlations between the variables, for example, can be a good indicative for choosing the best structure for the network. We also intend to investigate deeper the relations between the results produced by local and global search methods.

Acknowledgments

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

6. REFERENCES

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[2] R. Barros, A. de Carvalho, M. Basgalupp, and A. Freitas. Towards the automatic design of decision tree induction algorithms. In *Proceedings of the GECCO-2011– First Workshop on Evolutionary Algorithms for Evolving Generic Algorithms*, pages 567–574. ACM Press, July 2011.

[3] R. C. Barros, M. P. Basgalupp, A. C. de Carvalho, and A. A. Freitas. A hyper-heuristic evolutionary algorithm for automatically designing decision-tree algorithms. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 1237 – 1244, New York, NY, USA, 2012. ACM.

[4] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse. *WEKA Manual for Version 3-6-9*. University of Waikato, Hamilton, New Zealand, January 2013.

[5] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer, 2008.

[6] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123 – 140, Aug. 1996.

[7] J. Cheng and R. Greiner. Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI'99, pages 101 – 108, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

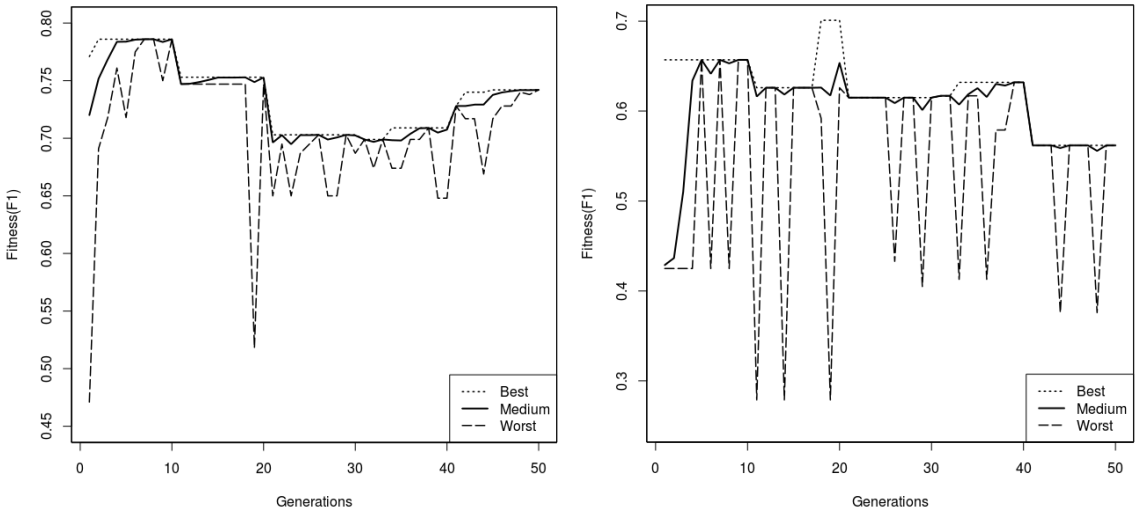
[8] J. Cheng and R. Greiner. Learning Bayesian belief network classifiers: Algorithms and system. In *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '01, pages 141 – 151, London, UK, UK, 2001. Springer-Verlag.

[9] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462 – 467, may 1968.

[10] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309 – 347, Oct. 1992.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[12] R. Daly, Q. Shen, and S. Aitken. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26(02):99 – 157, 2011.



(a) Diabetes dataset. (b) Liver-Disorders dataset.

Figure 5: Fitness convergence for two datasets.

[13] D. Floreano, P. Durr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1:47 – 62, 2008.

[14] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23 – 37, London, UK, UK, 1995. Springer-Verlag.

[15] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Mach. Learn.*, 29(2 - 3):131 – 163, Nov. 1997.

[16] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79 – 119, Jan. 1997.

[17] E. Herskovits and G. F. Cooper. Kulató: An entropy-driven system for construction of probabilistic expert systems from databases. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, pages 117 – 128, New York, NY, USA, 1991. Elsevier Science Inc.

[18] P. Langley, W. Iba, and, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pages 223 – 228. AAAI Press, 1992.

[19] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on evolutionary algorithms in bayesian network learning and inference tasks. *Information Sciences*, page in press, 2013.

[20] N. Lourenço, F. Pereira, and E. Costa. Evolving evolutionary algorithms. In *Proceedings of the GECCO-2012 – Second Workshop on Evolutionary Algorithms for Evolving Generic Algorithms*, GECCO Companion '12, pages 51–58, New York, NY, USA, 2012. ACM.

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

[22] J. P. Sacha. *New synthesis of bayesian network classifiers and cardiac spect image interpretation*. PhD thesis, 1999.

[23] J. P. Sacha, L. S. Goodenday, and K. J. Cios. Bayesian learning for cardiac spect image interpretation. *Artif. Intell. Med.*, 26(1 - 2):109 – 143, Sept. 2002.

[24] E. B. Santos, E. R. Hruschka, Jr., E. R. Hruschka, and N. F. F. Ebecken. Bayesian network classifiers: Beyond classification accuracy. *Intell. Data Anal.*, 15(3):279 – 298, Aug. 2011.

[25] J. Suzuki. Learning bayesian belief networks based on the mdl principle: An efficient algorithm using the branch and bound technique. *IEICE Transactions on Information and Systems*, E82-D:356 – 367, 1999. Institute of Electronics, Information and Communication Engineers.

[26] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artif. Intell. Rev.*, 18(2):77 – 95, Oct. 2002.

[27] Weka Machine Learning Project. Weka. URL: http://www.cs.waikato.ac.nz/~ml/weka.

[28] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[29] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241 – 259, 1992.