EVOLUTIONARY COMPUTATION BASED ON BAYESIAN CLASSIFIERS

TERESA MIQUÉLEZ[*], ENDIKA BENGOETXEA[*], PEDRO LARRAÑAGA[**]

[*] Department of Computer Architecture and Technology
University of the Basque Country
P.O. Box 649, 20080 San Sebastian, Spain
e-mail: `{teresa, endika}@si.ehu.es`

[*] Department of Computer Science and Artificial Intelligence
University of the Basque Country
P.O. Box 649, 20080 San Sebastian, Spain
e-mail: `ccplamup@si.ehu.es`

Evolutionary computation is a discipline that has been emerging for at least 40 or 50 years. All methods within this discipline are characterized by maintaining a set of possible solutions (individuals) to make them successively evolve to fitter solutions generation after generation. Examples of evolutionary computation paradigms are the broadly known Genetic Algorithms (GAs) and Estimation of Distribution Algorithms (EDAs). This paper contributes to the further development of this discipline by introducing a new evolutionary computation method based on the learning and later simulation of a Bayesian classifier in every generation. In the method we propose, at each iteration the selected group of individuals of the population is divided into different classes depending on their respective fitness value. Afterwards, a Bayesian classifier—either naive Bayes, seminaive Bayes, tree augmented naive Bayes or a similar one—is learned to model the corresponding supervised classification problem. The simulation of the latter Bayesian classifier provides individuals that form the next generation. Experimental results are presented to compare the performance of this new method with different types of EDAs and GAs. The problems chosen for this purpose are combinatorial optimization problems which are commonly used in the literature.

Keywords: hybrid soft computing, probabilistic reasoning, evolutionary computing, classification, optimization, Bayesian networks, estimation of distribution algorithms

1. Introduction

Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2001; Mühlenbein and Paaß, 1996; Pelikan *et al.*, 1999) deals with evolutionary computation techniques that store more than a solution every iteration similarly as Genetic Algorithms (GAs) (Goldberg, 1989; Holland, 1975). The main difference between these two paradigms is the fact that GAs evolve using crossover and mutation operators, without explicitly expressing the characteristics of the selected individuals within a population. EDAs take into account these characteristics by considering the interdependencies between the different variables that form an individual and learning a probabilistic graphical model to represent them.

EDAs allow us to take into account the dependencies between variables, and they have therefore shown to be more suitable for complex problems where these types of dependencies apply (Inza *et al.*, 2000). EDAs have a theoretical foundation in probability theory and are based on probabilistic modelling of promising solutions in combination with the simulation of models induced to guide their search.

In most of EDAs all selected individuals chosen for building the probabilistic graphical model, usually the fittest ones, are treated equally for the learning step, and no difference is done between the fitness of one or another. One of EDAs in which the learning takes into account the differences in fitness among the selected individuals is the Bit-Based Simulated Crossover (Syswerda, 1993). Other authors have already applied fitness in many evolutionary computation operators in the past, for instance, in multi-objective GAs (Zitzler *et al.*, 1999; Thierens and Bosman, 2001) and in discretization (Cantu-Paz, 2001).

This paper introduces EBCOAs (Evolutionary Bayesian Classifier-based Optimization Algorithms) as a new approach in evolutionary computation. The motivation for this approach that makes it innovative is twofold: firstly, it evolves a generation of individuals by constructing Bayesian classifier models that take into account deeper differences rather than simply a subset of individuals of the previous population. Secondly, it also takes into account the differences between individuals in the population that make them more or less fit regarding their fitness values, and it applies this knowledge to create a new population by enhancing the characteristics of the fitter individuals and tries to avoid the less fitted ones. In this paper we analyse many of the different possibilities that can be exploited in this new framework. Briefly speaking, the main contribution of this new approach is to propose the use of classification techniques in the form of Bayesian networks applied to optimization problems in order to improve the generation of individuals in every iteration.

This paper is organised as follows: the next section describes the estimation of distribution algorithms, paying special attention to the step of learning the probabilistic graphical model that allows the population to improve step after step. Section 3 describes the new paradigm that we propose in this paper as an innovative way of constructing probabilistic graphical models in the discrete domain by taking into account not only the dependencies between the different variables, but also the different fitness values of each of the individuals. Section 4 describes the experiments carried out, as well as the results obtained compared to other evolutionary computation techniques. Finally, Section 5 explains the conclusions and the future work to be done in this domain.

2. Estimation of Distribution Algorithms (EDAs)

2.1. Introduction

The main idea of Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2001; Mühlenbein and Paaß, 1996; Pelikan *et al.*, 1999) is to keep a population of individuals (or a set of solutions to a particular problem) and to make them evolve in order to obtain in each iteration a population of fitter individuals. Each individual is a vector of values considered to be instantiations of statistical variables. In EDAs the new population of individuals is generated by sampling from a probabilistic graphical model. This probabilistic graphical model is learned from a database containing only selected individuals from the previous generation, and the interrelations between the different variables that form each individual are expressed explicitly through the joint probability distribution associated to the individuals selected in each iteration.

Generally speaking, the EDA approach, illustrated in Fig. 1, contains the following steps:

1. The first population D_0 of R individuals is generated. The generation of these R individuals is usually produced by assuming a uniform distribution on each variable, and then each individual is evaluated.
2. A number N ($N < R$) of individuals are selected from D_l following some criteria (usually the ones with the best fitness values are selected, although in the literature there are many different selection procedures[1] that allow any individual to be selected). These individuals form the selected population D_l^N.
3. The n-dimensional probabilistic model $p_l(\boldsymbol{x}) = p_l(\boldsymbol{x}|D_l^N)$ that better represents the interdependencies between the n variables is induced. This model is created in the form of a probabilistic graphical model (i.e. a Bayesian network if the domain is discrete) containing the variables $X_1, X_2, \ldots, X_n$, where n is the size of each individual.
4. Finally, the new population D_{l+1} formed from the R new individuals is obtained by carrying out the simulation of the probability distribution learned in the previous step.

Steps 2, 3 and 4 are repeated until a stopping criterion is satisfied. Examples of stopping criteria are: achieving a fixed number of populations or a fixed number of different individuals, uniformity in the generated population, or the fact of having arrived at the optimum solution (at least, if the latter is known).

The step of estimating the joint probability distribution associated with the database of the selected individuals constitutes the hardest work to perform, and this task has to be performed for each generation. That is why methods proposed for learning probabilistic graphical models from data have been applied to EDAs. Furthermore, all the different EDA approaches proposed in the literature can be categorized in order of interdependencies between variables that they can take into account as follows: the ones that consider all the variables to be independent of each other (Baluja, 1994; Harik *et al.*, 1998; Mühlenbein, 1998; Syswerda, 1993), the ones that consider pairwise dependencies (Baluja and Davies, 1997; Chow and Liu, 1968; Pelikan and Mühlenbein, 1999), and the ones that can take into account multiple dependencies between the variables (Etxeberria and Larrañaga, 1999; Harik, 1999; Mühlenbein and Mahning, 1999; Mühlenbein *et al.*, 1999; Pelikan *et al.*, 1999). The reader can find a more complete review on this topic in (Larrañaga and Lozano, 2001).

2.2. Towards a More Efficient Learning Phase

The step of learning the probabilistic graphical model is performed at each iteration, and this results in a new population. In EDAs, the set of individuals selected to learn the probabilistic graphical model are usually the best ones

[1] Other methods in the literature propose to create multiple copies of the fittest solutions and fewer for the inferior ones to form the new population.

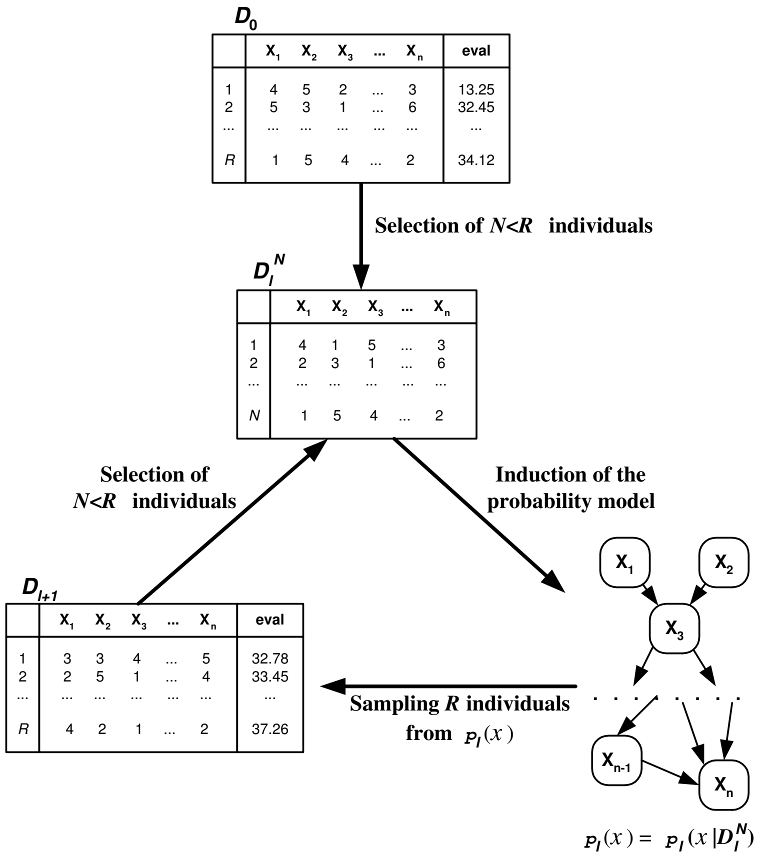


Fig. 1. Illustration of the EDA approach in the optimization process.

(i.e. the fittest ones). This particular selection of individuals ensures that the model will represent the interdependencies of the variables on the selected individuals. In addition, in practically all EDAs proposed in the literature the fitness value of each of the selected individuals is not taken into account, and therefore the best and worst individuals within D_l^N are treated equally in the learning step (i.e. the fitness value of each individual is ignored after the selection-of-individuals step).

Considering that the fitness of each of the individuals should be also taken into account in the learning step, three of the possible ways of considering these are the following:

- *Weighting the influence of the individuals depending on their fitness value.* This possibility consists in adding the fitness value given by the objective function directly in the learning step. This fitness value is used to give a different weight to the different selected individuals in the construction of the probabilistic graphical model. An example of this idea is present in BSC (Syswerda, 1993). Another way of taking into account the differences in the fitness of individuals in a population is to use also a proportional selection method, as well as a Boltzman distribution based selection (Mühlenbein and Mahning, 1999).

- *Adding the fitness as a new variable.* This second category takes into account the fitness value obtained by the different individuals as a new variable. This variable is included in the probabilistic graphical model together with the variables $X_1, \dots, X_n$. The fact of including the fitness value as another variable requires that the learning algorithms that are to be applied deal with a variable that is typically continuous, while the rest of the variables are usually discrete. When this is the case, the learning procedures that can be applied for the construction of the probabilistic graphical model are more complex and require considerable CPU time.

- *Turning into a supervised classification problem.* The main idea here is to classify all individuals of a population into different classes, and to use algorithms to build Bayesian classifiers in order to create new individuals taking into account the characteristics of the fittest classes and trying to avoid those of the worst classes. The aim is to guide the search taking into account both the fittest and the less fitted individuals. This is the approach that we propose in this paper, and it is described in the next section.

3. Evolutionary Bayesian Classifier-Based Optimization Algorithm

This section describes the new method called Evolutionary Bayesian Classifier-based Optimization Algorithms (EBCOAs) that we propose for optimization problems. In much the same way as EDAs, this approach combines both probabilistic reasoning and evolutionary computing. In particular, EBCOAs are based on using Bayesian classifiers in evolutionary computation. A description of the state of the art approaches to applying supervised classification techniques to optimization is introduced firstly. Next, some notation is introduced prior to the formal description of the new method.

3.1. State of the Art

One of the first proposals in the literature for applying classification techniques in optimization is the Learnable Execution Model (LEM) (Michalski, 2000). In contrast to other evolutionary computation techniques such as GAs and EDAs, LEM algorithms apply classifiers to develop a population of solutions. In this approach, individuals of a population are divided into the fittest and the less fitted ones, and characteristics of the good ones are strengthened while bad ones are avoided. Michalski proposed in his work an original machine learning method called AQ18 (Kaufman and Michalski, 1999). This supervised classification method uses general inductive rule learning methods that are configurable for faster convergence. LEM can be regarded as a hybrid approach that applies non-statistical model learning approaches while relying on traditional evolutionary computation mechanisms (Ventura *et al.*, 2002).

There are also other statistical approaches that combine statistical classifier-construction methods and evolutionary computation. Examples of these are, for instance, the use of decision trees (Llorà and Goldberg, 2003; Muñoz, 2003).

3.2. Description of the New Method

Our innovative approach, called Evolutionary Bayesian Classifier-based Optimization Algorithms (EBCOAs), proposes the use of classifiers in the form of Bayesian networks for optimization problems by applying them in a manner analogous to that used in the EDAs. The main idea is that in each generation the population will evolve by constructing a new Bayesian classifier, but in contrast to EDAs, individuals that are used for constructing the probabilistic graphical model in EBCOAs are not simply the selected ones (i.e. most usually the fittest ones), and in each generation the bad (less fitted) individuals will also be considered for the learning procedure so that the algorithm also takes into account the characteristics that the less fitted individuals have when evolving to the next generation. This idea aims at providing faster convergence in optimization problems by modelling the different characteristics that make individuals in the current population fitter or worse using Bayesian classifiers.

EBCOAs follow an evolutionary computation approach similar to EDAs, although the main differences between EDAs and EBCOAs concern the method for building the Bayesian network: in the former the learning algorithms are taken from the general purpose Bayesian network induction algorithms while the latter are algorithms to build Bayesian classifiers using the information provided by the fitness function in a more appropriate manner. Figure 2 illustrates the EBCOA approach, and Fig. 3 shows its pseudocode. If we compare these figures with Fig. 1, it can be seen that the main difference between EBCOAs and EDAs is precisely the step of learning the model.

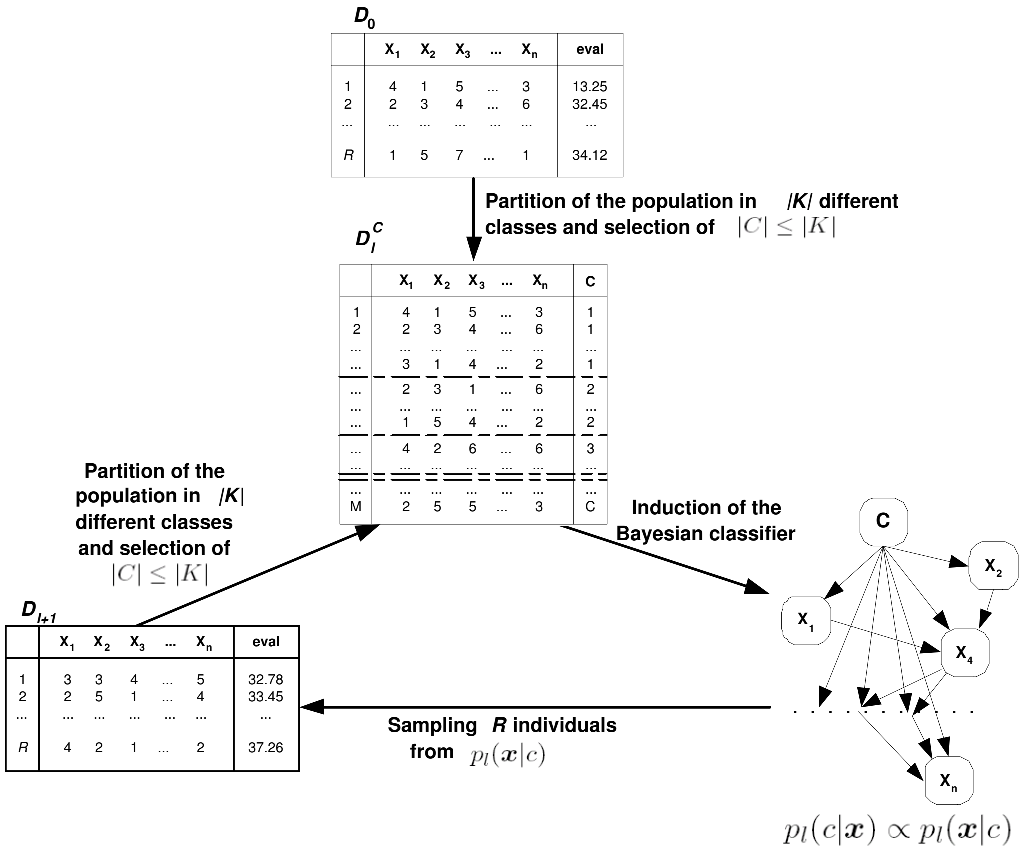
3.3. Notation

Let $\boldsymbol{X} = (X_1, \dots, X_n)$ be an n-dimensional random variable. Then $\boldsymbol{x} = (x_1, \dots, x_n)$ represents one of its possible instantiations and therefore one of the possible individuals. The probability of $\boldsymbol{X}$ will be denoted by $p(\boldsymbol{X} = \boldsymbol{x})$, or simply $p(\boldsymbol{x})$. The conditional probability of the variable X_i given the value x_j of the variable X_j will be written as $p(X_i = x_i | X_j = x_j)$, or simply as $p(x_i|x_j)$.

Let D_l be the l-th population (database) of the R individuals that has to evolve into the $(l+1)$-th one. In EBCOAs, before proceeding to the learning, the population D_l is divided into $|K|$ different classes following a supervised classification approach, and we define a variable K that can take the values $\{1, 2, \dots, |K|\}$. We denote by D_l^K the database D_l after it has been divided into he $|K|$ classes, in which for each individual in the population we have assigned a value k to the variable K with $1 \leq k \leq |K|$ in order to represent the class to which each individual has been assigned. Since all the classes are not usually used for the learning, prior to training the Bayesian classifier we choose $|C| \leq |K|$ classes and the rest are simply ignored for learning purposes. We denote by D_l^C the subset of D_l^K that will be used for the learning. We also denote by C the variable that assigns a class c (with $1 \leq c \leq |C|$) to each of the individuals in D_l^C.

The result of the learning step is the construction of a probabilistic graphical model, that is, a Bayesian network in the discrete domain. In EBCOAs, this Bayesian network is a Bayesian classifier that takes into account the variables $X_1, X_2, \dots, X_n$, as well as the variable C.

The main task in an EBCOA is to estimate $p_l(\boldsymbol{x} \mid c)$, that is, the probability of an individual $\boldsymbol{x}$ to be part of


D_0
Partition of the population in $|K|$ different classes and selection of $|C| \leq |K|$
D_l^C
Partition of the population in $|K|$ different classes and selection of $|C| \leq |K|$
Induction of the Bayesian classifier
C
X_1
X_2
X_4
X_n
D_{l+1}
Sampling R individuals from $p_l(\boldsymbol{x}|c)$
$p_l(c|\boldsymbol{x}) \propto p_l(\boldsymbol{x}|c)$

D_0

	X_1	X_2	X_3	...	X_n	eval
1	4	1	5	...	3	13.25
2	2	3	4	...	6	32.45
...	...	...	...	...	...	...
R	1	5	7	...	1	34.12

D_l^C

	X_1	X_2	X_3	...	X_n	C
1	4	1	5	...	3	1
2	2	3	4	...	6	1
...	...	...	...	...	...	...
...	3	1	4	...	2	1
...	2	3	1	...	6	2
...	...	...	...	...	...	...
...	1	5	4	...	2	2
...	4	2	6	...	6	3
...	...	...	...	...	...	...
...	...	...	...	...	...	...
M	2	5	5	...	3	C

D_{l+1}

	X_1	X_2	X_3	...	X_n	eval
1	3	3	4	...	5	32.78
2	2	5	1	...	4	33.45
...	...	...	...	...	...	...
R	4	2	1	...	2	37.26

Fig. 2. Illustration of the EBCOA approach in the optimization process.

```
$D_0$ ← Generate $R$ individuals (the initial population) randomly

**Repeat** for $l = 0, 1, 2 \ldots$ until a stopping criterion is met

    $D_l^K$ ← Divide the $R$ individuals in $|K| < R$ different classes from $D_l$
            according to a criterion

    $D_l^C$ ← Select the $|C| \leq |K|$ classes of $D_l^K$ that will be used for building the
            Bayesian classifier, usually taking into account at least the best and worst classes.
            The individuals of the classes not included in $D_l^C \subset D_l^K$ are ignored

    $p_l(c|\boldsymbol{x}) \propto p_l(\boldsymbol{x}|c)$ ← Estimate the probability distribution of an individual in $D_l^C$
            of being part of any of the different possible $|C|$ classes

    $D_{l+1}$ ← Sample $R$ individuals (the new population) from $p_l(\boldsymbol{x}|c)$
```

Fig. 3. Pseudocode for the EBCOA approach.

each of the classes $1, 2, \ldots, |C|$ in D_l^C. This probability must be estimated in every generation since the population and hence the nature of the classes are different for each of them. In EBCOAs, the Bayesian network structure S that is induced as a result of the learning step will contain the variables $X_1, \ldots, X_n$ as in EDAs, but also the newly defined variable C. This variable C will be present in all the structures that are obtained using Bayesian classifier-building algorithms by EBCOAs, and C will always be a parent of all the other variables in S.

In EBCOAs we apply methods from the Bayesian classifier-building algorithms that are described in the next section.

3.4. Bayesian Classifiers

This section revises some of the classifiers in the form of Bayesian networks that have been proposed as classifiers in the literature. Their main characteristic is the number of dependencies between variables that the Bayesian network can take into account. We revise here these classifiers from the simplest to the most complex ones.

The *supervised classification* problem consists in assigning a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ to one of the $|C|$ classes of variable C. The true class is denoted by c and it takes values from the set $\{1, 2, \ldots, |C|\}$. We can regard the classifier as a function $\gamma : (x_1, \ldots, x_n) \rightarrow \{1, 2, \ldots, |C|\}$ that assigns labels to observations.

According to (Duda and Hart, 1973), and for the particular case of having a loss function $0/1$[2], the optimum Bayesian classifier (in the sense that it minimizes the total misclassification error cost) is obtained by assigning to the example $\boldsymbol{x} = (x_1, \ldots, x_n)$ the class with the highest posterior probability, i.e.

$$\gamma(\boldsymbol{x}) = \arg\max_c p(c|x_1, \ldots, x_n). \qquad (1)$$

This section revises Bayesian classifiers that are meant specifically for classification problems. Therefore, some of these classifiers can be considered as too simplistic or not very efficient from the point of view of the classification task, and some of them can be of interest for optimization with EBCOAs since our purpose is to have a relatively effective learnable algorithm that can be executed in a reasonable period of time at every iteration.

3.4.1. Naive Bayes

The paradigm that combines the Bayes theorem and the conditionally independent hypothesis given the class is known as *idiot Bayes* (Ohmann *et al.*, 1988), *naive Bayes* (Kononenko, 1990), *simple Bayes* (Gammerman and Thatcher, 1991), or *independent Bayes* (Todd and Stamper, 1994). Although it has a long tradition in the *pattern recognition* community (Duda and Hart, 1973), the naive Bayes classifier was commented for the first time in the *machine learning* field by (Cestnik *et al.*, 1987). Gradually, the machine learning community realized its potential and robustness in supervised classification problems. In that sense, although in this classifier the estimation of $p(c|x_1, \ldots, x_n)$ is not well calibrated, naive Bayes has proved to be quite effective for many classification problems (Domingos and Pazzani, 1997), being able to obtain results comparable to other more complex classifiers.

The naive Bayes approach (Minsky, 1961) is the simplest among the classifiers that are presented in this paper. This Bayesian network has always the same structure: all variables $X_1 \ldots X_n$ are considered to be conditionally independent given the value of the class value C. Figure 4 shows the structure that would be obtained in a problem with four variables.

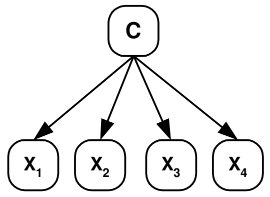


Fig. 4. Graphical structure of the naive Bayes model.

The main advantage of this approach is the fact that the structure is always fixed, that is, the process of learning the classifier is very fast since the order of dependencies to be found is fixed and reduced to two variables. In naive Bayes, the only task to accomplish so far is the estimation of the probabilities that are to be considered following this Bayesian network.

Following the naive Bayes model, we have that when classifying an example $\boldsymbol{x}$, it will be assigned to the class c for which it has a higher posterior probability. In order to calculate this posterior probability, we have

$$p(c \mid \boldsymbol{x}) \propto p(c, \boldsymbol{x}) = p(c) \prod_{i=1}^{n} p(x_i|c). \qquad (2)$$

The estimation of the prior probability of the class, $p(c)$, as well as the conditional probabilities $p(x_i|c)$, is performed based on the database of selected individuals in each generation.

3.4.2. Selective Naive Bayes

The main difference between the selective naive Bayes approach (Kohavi and John, 1997; Langley and Sage, 1994)

[2] In a $0/1$ loss function the cost of misclassifying an element is always 1.

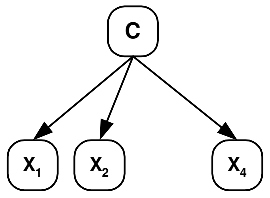


Fig. 5. Example of a graphical structure of the selective naive Bayes model for a problem of four variables.

and naive Bayes is that in the former not all variables have to be present in the final model. Figure 5 shows the structure that could be obtained in a problem with four variables, where one of them is missing in the final structure. In naive Bayes the condition of having to take into account all variables appears to be very strict for some type of classification problems, since some variables could be irrelevant (i.e. variables that always have the same values in all classes) or redundant (i.e. those in which all values appear similarly in the different classes and therefore do not reflect any difference between the characteristics of the classes) for classification purposes.

It is known (Liu and Motoda, 1998; Inza *et al.*, 2000) that the behaviour of the naive Bayes paradigm degrades with redundant variables, and therefore the motivation for this approach is to remove those variables in order to obtain more efficient classifiers.

Following the selective naive Bayes model, and using the selective naive Bayes classifier shown in Fig. 5, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the class

$$c^* = \arg\max_c p(c)p(x_1|c)p(x_2|c)p(x_4|c). \qquad (3)$$

3.4.3. Seminaive Bayes

The seminaive Bayes approach (Kononenko, 1991) can be considered as a more sophisticated type of the Bayesian classifier regarding the type of dependencies that it can take into account, as it allows groups of variables to be considered as a single node in the Bayesian network. The aim of this seminaive Bayesian classifier is to avoid the strict premises of the naive Bayes paradigm by allowing to group some variables in a single node of the structure. Figure 7(3) illustrates an example of a seminaive Bayesian classifier in a problem with four variables, showing that the Bayesian network structure treats these grouped variables as a single one regarding the factorization of the probability distribution. When grouping variables, whether two, three or more, all dependencies between them are taken into account implicitly for classification purposes. On the other hand, Fig. 7(3) also shows that it is possible that some variables are not included in the final classifier.

Pazzani (1997) presents a greedy approach in which redundant and dependent variables are detected. When dependent variables are found, a new variable is created as the Cartesian product of these. Two greedy algorithms are presented, the first of them in a forward direction called *FSSJ (Forward Sequential Selection and Joining)*, and the second in the backward direction named *BSEJ (Backward Sequential Elimination and Joining)*. The pseudocode of *FSSJ* is shown in Fig. 6. The *BSEJ* algorithm follows an analogous approach, and can be interesting in optimization problems in which the objective function depends on all or nearly all variables. Note that this modelling process follows a wrapper approach (Kohavi and John, 1997).

Figure 7 shows an example of the application of the *FSSJ* algorithm. The procedure that is followed in this figure is explained next. In (1), after comparing all naive Bayes models with a single predictor variable, the variable X_4 was selected. In (2), the rest of the variables were compared, and adding the variable X_2 is the model that provides most gain after comparing the following subsets of variables: $\{X_4, X_1\}$, $\{X_4, X_2\}$, $\{X_4, X_3\}$, $\{(X_4, X_1)\}$, $\{(X_4, X_2)\}$, $\{(X_4, X_3)\}$. In (3), adding the variable X_1 grouped to X_2 is the winner of the remaining possibilities: $\{X_4, X_2, X_1\}$, $\{X_4, X_2, X_3\}$, $\{(X_4, X_1), X_2\}$, $\{X_4, (X_1, X_2)\}$, $\{(X_4, X_3), X_2\}$, $\{X_4, (X_3, X_2)\}$. As the algorithm ends without adding the variable X_3 to the final structure, this means that the models $\{X_4, X_3, (X_1, X_2)\}$, $\{(X_4, X_3), (X_1, X_2)\}$, $\{X_4, (X_3, X_1, X_2)\}$ do not exhibit any improvement over the model obtained in (3). As a result, following the seminaive Bayes model and using the final classifier obtained in this figure, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the following class:

$$c^* = \arg\max_c p(c)p(x_1, x_2|c)p(x_4|c). \qquad (4)$$

3.4.4. Tree Augmented Naive Bayes

Tree augmented naive Bayes (Friedman *et al.*, 1997) is another Bayesian network classifier in which the dependencies between variables other than C are also taken into account. These models represent the relationships between the variables $X_1, \ldots, X_n$ conditional on the class variable C by using a tree structure.

The tree augmented naive Bayes structure is built in a two-phase procedure for which the pseudocode is given in Fig. 8. Firstly, the dependencies between the different variables $X_1, \ldots, X_n$ are learned. This algorithm uses a score based on information theory, and the weight of a branch (X_i, X_j) on a given Bayesian network S is defined by the mutual information measure conditional on

```
Initialize the set of variables to be used in the null set.
Classify all the examples as being of a class with higher $p(c)$
**Repeat** in every iteration: choose the best option between
    (a) Consider each variable that is not in the model as a new one to be
        included in it. Each variable should be added as conditionally
        independent of the variables in the model given the class
    (b) Consider grouping each variable not present in the model with a variable
        that is already in it
    Evaluate each possible option by means of the estimation of the percentage
        of cases well classified
**Until** no improvement can be obtained
```

Fig. 6. Pseudocode of the *FSSJ* algorithm for seminaive Bayes models.

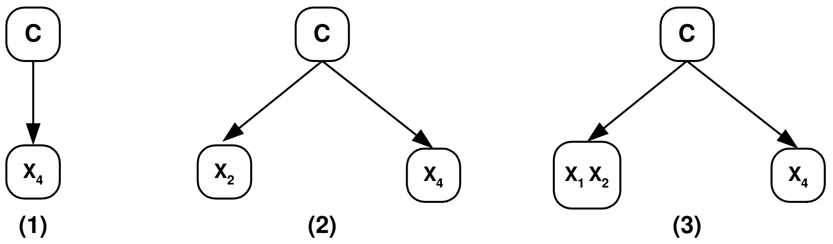


Fig. 7. Steps of the construction of a Bayesian classifier following the *FSSJ* algorithm in a problem with four variables. X_1, X_2, X_3, X_4 are the predictor variables and C is the variable to be classified.

```
Calculate $I(X_i, X_j \mid C) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{r=1}^{w} p(x_i, y_j, c_r) \log \frac{p(x_i, y_j | c_r)}{p(x_i | c_r) p(y_j | c_r)}$
        with $i < j, j = 2, \ldots, n$
Build an undirected complete graph, where the nodes correspond to the predictor
    variables: $X_1, \ldots, X_n$. Assign the weight $I(X_i, X_j \mid C)$ to the edge connecting
    variables $X_i$ and $X_j$
Assign the largest two branches to the tree to be constructed

**Repeat** in every iteration:
        Examine the next largest branch and add it to the tree unless it forms a loop.
        In the latter case discard it and examine the next largest branch
**Until** $n - 1$ branches have been added to the structure

Transform the undirected graph in a directed one, by choosing a random
    variable as the root
Build the tree augmented naive Bayes structure adding a node labelled as $C$, and later
    add one arc from $C$ to each of the predictor variables $X_i$ $(i = 1, \ldots, n)$
```

Fig. 8. Pseudocode of the *tree augmented naive Bayes* algorithm.

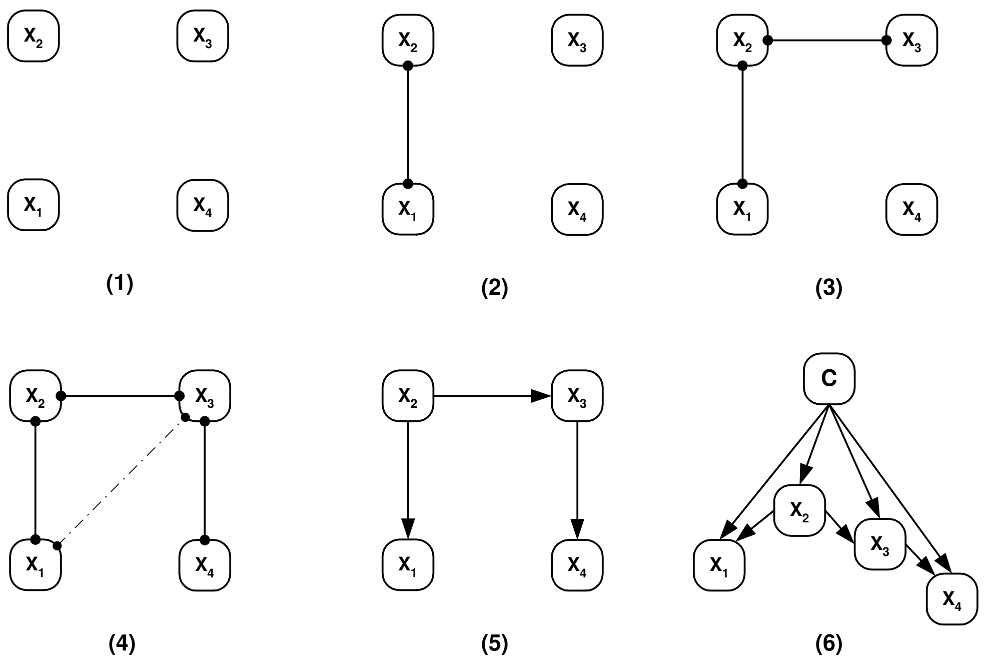


Fig. 9. Illustration of the steps for building a tree augmented naive Bayes classifier in a problem with four variables. X_1, X_2, X_3, X_4 are the predictor variables and C is the variable to be classified.

the class variable as

$$I(X_i, X_j|C) = \sum_c p(c) I(X_i, X_j|C=c) = \sum_c \sum_{x_i} \sum_{x_j} p(x_i, x_j, c) \log \frac{p(x_i, x_j|c)}{p(x_i|c)p(x_j|c)}. \quad (5)$$

With these conditional mutual information values the algorithm builds a tree structure. In the second phase, the structure is augmented into the naive Bayes paradigm.

Figure 9 shows an example of the application of the tree augmented naive Bayes algorithm. This figure assumes that $I(X_1, X_2|C) > I(X_2, X_3|C) > I(X_1, X_3|C) > I(X_3, X_4|C) > I(X_2, X_4|C), I(X_1, X_4|C)$. In (4) the branch (X_1, X_3) is rejected since it would form a loop. Here (6) is the result of the second phase of augmenting the tree structure. Following the tree augmented naive Bayes model, and using the classifier shown in this figure, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the class

$$c^* = \arg\max_c p(c) p(x_1|c, x_2) p(x_2|c) \times p(x_3|c, x_2) p(x_4|c, x_3). \quad (6)$$

In contrast to the wrapper approach as a score to measure the goodness of the structures applied in the semi-naive Bayes model, the tree augmented naive Bayes algorithm follows a method that is analogous to filter approaches, where only pairwise dependencies are considered.

3.4.5. Other Methods

There are several other methods to build Bayesian classifiers taking into account more or fewer dependencies between variables. These methods have been extensively proposed in the last years and their number is growing quite fast as they constitute a hot research topic. Examples of Bayesian classifiers that can be found in the literature are the K-dependence Bayesian classifier (Sahami, 1996), Bayesian augmented networks (Cheng and Greiner, 1999), general Bayesian networks (Neapolitan, 2003), and Bayesian multinets (Kontkanen *et al.*, 2000).

3.5. Description of the Main Steps of EBCOAs

Having described the different Bayesian classifiers that we can apply to EBCOAs, this section describes the main steps of the method as well as the implications of the different choices to be done on them.

3.5.1. Supervised Classification Step: Labelling Individuals and Selecting Classes

In EBCOAs, instead of selecting a subset of individuals as EDAs do, the whole population is firstly classified into a fixed number $|K|$ of different classes. These classes are formed by dividing the whole population into groups of individuals from the fittest to the least fitted ones. The result of this procedure is to assign to each individual in D_l a label k (with $k \in \{1, 2, \ldots, K\}$). Each of the R individuals is assigned a label k, and they form the class variable K in the database D_l^K.

As in EBCOAs the aim is also to take into account the main characteristics that distinguish both the fittest and the less fitted classes, some of the classes in D_l^K could be discarded to facilitate the learning. An example of this idea is to ignore the middle classes in D_l^K for the learning of the Bayesian classifier, so that the differences between the most distant classes are enhanced. D_l^C is the result of removing from D_l^K the classes that are not used for learning, and C is the class variable that is used for learning as the root of the Bayesian classifiers, with $|C| \leq |K|$.

3.5.2. Learning Step: Building the Bayesian Classifier

Learning is performed by applying an algorithm to induce a Bayesian classifier such that it forms a Bayesian network in which the root is the variable C representing the labels of the individual (C is treated as another variable), and the rest of the variables X_1 to X_n can also be present. This Bayesian network will be formed following different classifier construction algorithms such as the ones described in the previous section. Therefore, the probabilistic graphical model obtained as a result of this method will contain a maximum of $n+1$ nodes (the variables X_1 to X_n and C), with the variable C always being the root and the parent of all the rest. As a result of this learning procedure, probability distribution can be represented by a factorization of the form $p_l(c|\boldsymbol{x}) \propto p_l(\boldsymbol{x}|c)$.

It is important to realize that in our case we are not interested in obtaining the best possible Bayesian classifier to represent a strictly correct classifier. These algorithms for obtaining optimum classifiers in the form of a Bayesian network are very time consuming, and the execution time requirement is crucial in EBCOAs. Taking into account the fact that this learning step (i.e. the classifier building step) is going to be applied in every generation, it is more important to use a Bayesian classifier builder that will return a satisfactory classifier in a reasonable time rather than a perfect classifier that will be ignored in the next generation.

3.5.3. Simulation Step: Instantiating the New Population

The step of instantiating the probabilistic graphical model to obtain the new R individuals is also performed in a similar way as in EDAs, although there is an important difference due to the fact of the existence of the C variable in the Bayesian network: every individual will be generated using a specific criterion, such as, for instance, the probability distribution $p_l(\boldsymbol{x}|c)$. Therefore, the simulation of the individual is performed following the probability distribution learned in the previous step.

But the main difference comes from the need to reflect the different characteristics of individuals from the fittest and less fitted classes. In that sense, to perform the simulation and thus the generation of new individuals that will form the next population D_{l+1}, the individuals should be generated using all classes in C. Our proposal is to generate R new individuals by assigning a different number of individuals by instantiating the probability distribution of all classes proportional to $p(c)$, knowing that

$$p(c) \propto \sum_{\boldsymbol{x}|\ C(\boldsymbol{x})=c} f(\boldsymbol{x}), \tag{7}$$

where $f(\boldsymbol{x})$ is the fitness value of the individual $\boldsymbol{x}$, and $C(\boldsymbol{x})$ is the class assigned to the individual $\boldsymbol{x}$ in D_l^C. After generating these new R individuals, we fuse these with the previous R individuals of the population D_l, and we select the R/C individuals that better adjust to the characteristics of each of the casses of C, thus obtaining the R individuals that will form the next population D_{l+1}.

The reason for doing the simulation in this way is to ensure that individuals from all classes will be present in the next generation, while giving more chance to include individuals from the fittest ones according to the fitness value of the individuals. Following this procedure, even individuals from the less fitted classes will be included in the new generations, and this fact ensures that the differences between the fittest individuals and the less fitted ones are still present in the last generations of the search process as the algorithms converge to the optimum solution. The fact of keeping these differences is important since the convergence of the whole approach is based on the ability of the Bayesian classifier to model the main characteristics that place an individual within the fittest class found in the whole search process.

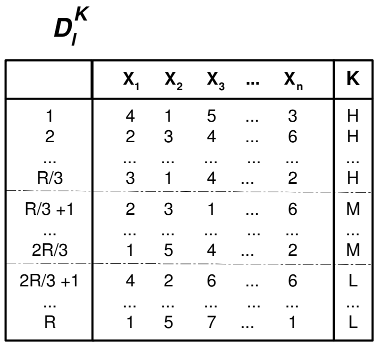
Another important point worth commenting regarding the generation of new individuals of the next population D_{l+1} is the decision of how to instantiate some of the variables that are not present in the Bayesian network classifier. This can happen, for instance, if in the learning step we apply algorithms such as selective Bayes or seminaive Bayes. These two algorithms can induce a Bayesian classifier in which some of the variables $X_1, X_2, \ldots, X_n$ are not present at all. Note that for the purpose of instantiating new individuals using such a model, this is a different situation as to have these variables present but disconnected, as even when the variables appear to be disconnected they have a probability distribution that has been learned and therefore they have probability distribution $p_l(\boldsymbol{x})$ estimated for allowing simulating new individuals. The meaning of not having a variable X_i present in the final Bayesian classifier structure implies that the values assigned to such a variable in the individuals of all $|C|$ classes are not relevant for distinguishing between them. This has an important consequence, since it does not mean that the value assigned to such variables is not important and that any value can be set. Note that the individual is a point in the search space for a specific problem, and that all values assigned to all the variables are usually relevant for obtaining a fitted individual and therefore converge to the optimum solution. However, as the search goes on, some variables might have the same values on the best and worst classes, and therefore in the learning step of EBCOAs these will be removed from the Bayesian classifiers. As a result, we propose to simulate the variables not present in the Bayesian classifier as follows: we consider that it is important to distinguish between irrelevant variables (i.e. variables that always have the same values in all classes) and redundant variables (i.e. those in which all values appear similarly in the different classes and therefore do not reflect any difference between the characteristics of the classes). For the former, the estimated probability for a redundant variable X_i to take its k-th value is computed as $\hat{p}(x_i) = p(x_i^k|c)$. For the latter type of variables, we assume that the probability distribution is uniform.

3.5.4. Stopping Criterion

All the previous steps are repeated in EBCOAs until a stopping condition is satisfied. Examples of stopping conditions are: achieving a fixed number of populations or a fixed number of different evaluated individuals, uniformity in the generated population, and the fact of not obtaining an individual with a better fitness value after a certain number of generations.

4. Experimental Results

An experiment was carried out in order to test the performance of EBCOAs compared with some EDAs and GAs. This section describes the experiments and the results obtained. We chose EDAs that take into account different numbers of dependencies between variables, in particular, UMDA (Mühlenbein, 1998), MIMIC (de Bonet *et al.*, 1997), and EBNA$_{BIC}$ (Etxeberria and Larrañaga, 1999).



D_l^K

	X_1	X_2	X_3	...	X_n	K
1	4	1	5	...	3	H
2	2	3	4	...	6	H
...	...	...	...	...	...	...
R/3	3	1	4	...	2	H
R/3 +1	2	3	1	...	6	M
...	...	...	...	...	...	...
2R/3	1	5	4	...	2	M
2R/3 +1	4	2	6	...	6	L
...	...	...	...	...	...	...
R	1	5	7	...	1	L

Fig. 10. Three classes in D_l^K chosen for our experiments, from which only the clases H and L will be used in our case. The class M is simply ignored in the learning step, since those individuals will not be present in D_l^C.

The GAs that we chose are the broadly known basic (cGA) (Holland, 1975), elitist (eGA) (Whitley and Kauth, 1988) and steady state (ssGA) (Michalewicz, 1992) ones. We tried three standard optimization problems in the discrete domain such as HIFF, IsoPeak, and IsoTorus, which are known to be complex and full of local optima. Table 1 describes briefly these three functions. The reader can find more information on these problems in (Santana, 2004).

In our particular experiments, in order to show the validity of EBCOAs, we divided each population into three different classes ($|K| = 3$), from which only those of the best and worst individuals are used for the learning step ($|C| = 2$). This is illustrated in Fig. 10. The stopping criterion in all experiments is when obtaining the optimum solution in that generation or reaching the generation number 500.

Table 2 shows the mean fitness of the best individual found in the last generation, as well as the number of generations to reach the final solution for each of the experiments. In IsoPeak there is a local optimum with fitness 3906 which corresponds to the individuals with only zeros, very close to the global optimum. This fact confuses most algorithms, and even if some of them are sometimes able to find it (EBNA 3 times, and ssGA once), the results show that only the EBCOA$_{TANB}$ algorithm was able to find the optimum in all the ten runs. In IsoTorus there are also other local optima, and EDAs and GAs fall in these in some of the executions. From the ten runs of each algorithm, most of EDAs and GAs were able to find sometimes the global optimum (MIMIC once, EBNA and cGA 4 times, and ssGA and eGA 5 times), but EBCOA$_{nBayes}$ and EBCOA$_{TANB}$ found the global optimum in all of the 10 runs, while EBCOA$_{seminnB-BSSJ}$ and EBCOA$_{selectivenBayes}$ also found it 8 times and twice, respectively. In the HIFF fitness function the results

Table 1. Description of the HIFF, IsoPeak, and IsoTorus fitness functions. The first column describes the objective funtion, the second the size of the individual, and the third and the fourth contain are the optimum solutions and their respective fitness values.

HIFF	$HIFF(x) = f(x_1,\dots,x_n)$ $f(x_1,\dots,x_n) = \begin{cases} 1, & \text{if } (\lvert s\rvert = 1) \\ \lvert s\rvert + f(x_1,\dots,x_{\frac{s}{2}}) + f(x_{\frac{s}{2}+1},\dots,x_s) & \text{if } (\lvert s\rvert > 1) \text{ and } \left(\sum_{i=1}^{\lvert s\rvert} x_i = 0\right), \text{ or } \left(\sum_{i=1}^{\lvert s\rvert} x_i = \lvert s\rvert\right) \\ f(x_1,\dots,x_{\frac{s}{2}}) + f(x_{\frac{s}{2}+1},\dots,x_s) & \text{otherwise} \end{cases}$	$n = 64$	$\begin{cases} (1,1,\dots,1) \\ (0,0,\dots,0) \end{cases}$	$Opt = 448$
IsoPeak	$m = n+1$ $IsoC_1 = \begin{cases} m & \text{if } x = 00 \\ m-1 & \text{if } x = 11 \\ 0 & \text{otherwise} \end{cases}$ $IsoC_2 = \begin{cases} m & \text{if } x = 11 \\ 0 & \text{otherwise} \end{cases}$ $F_{IsoPeak}(\boldsymbol{x}) = IsoC_2(x_1,x_2) + \sum_{i=2}^{m} IsoC_1(x_i,x_{i+1})$	$n = 64$	(1,1,…,1)	$Opt = 3907$
IsoTorus	$n = m^2$ $IsoT_1 = \begin{cases} m & \text{if } u = 0 \\ m-1 & \text{if } u = 5 \\ 0 & \text{otherwise} \end{cases}$ $IsoT_2 = \begin{cases} m^2 & \text{if } u = 5 \\ 0 & \text{otherwise} \end{cases}$ $F_{IsoTorus} = IsoT_1(x_{1-m+n} + x_{1-m+n} + x_1 + x_2 + x_{1+m}) + \sum_{i=2}^{n} IsoT_2(x_{up} + x_{left} + x_i + x_{right} + x_{down})$ where $x_{up}, x_{left}, x_i, x_{right}, x_{down}$ are defined as the appropriate neighbors	$n = 64$	(1,1,…,1)	$Opt = 505$

Table 2. Mean results after 10 executions with each algorithm and objective function. The *Ev* and *Val* columns represent respectively the best fitness value obtained in the last generation, and the evaluations number in which it ended.

	HIFF		IsoPeak		IsoTorus	
	Ev.	Val.	Ev.	Val.	Ev.	Val.
EBCOA$_{nBayes}$	105036.8	290	51995.4	3906	25175.9	505
EBCOA$_{selectivenBayes}$	94640.7	355.2	43910.0	3906	207914.1	472
EBCOA$_{seminnB-FSSJ}$	249838.2	290.2	249893.5	3859.8	227610.3	471.6
EBCOA$_{seminnB-BSSJ}$	189178.9	184.5	58694.3	3803.8	66701.9	474.3
EBCOA$_{TANB}$	4589.9	448	4391.8	3907	3989.6	505
UMDA	107120.4	295.6	67303.3	3905.5	47244.7	400.3
MIMIC	97572.0	283.2	69385.9	3906	46941	422.3
EBNA	23336.0	448	19708.6	3906.3	28703.0	485.2
cGA	202000	395.2	202000	3628.1	202000	477.2
eGA	202000	388.8	202000	3793.7	202000	488.5
ssGA	202000	448	202000	3906.1	202000	488.5

are more similar between EBCOAs, EDAs and GAs, since EBCOA$_{TANB}$, EBNA and ssGA obtained the best result in all the 10 runs. Also note that most of EBCOAs require fewer evaluations (e.g. fewer different solutions to be evaluated during the search) to reach these final results.

These results show that the tree augmented naive Bayes approach performs very well in all these fitness functions, even improving the results obtained in many EDAs and GAs. Also, if we compare the behaviour of EBCOA$_{nBayes}$ with that of UMDA, its EDA equivalent in taking into account the dependencies between variables, we see that the results are at least comparable. Finally, regarding the seminaive and selective Bayes approaches, after monitoring the evolution of the search we realized that the choice of how to instantiate the variables that are not present in the Bayesian classifier is the main reason for these results, and further research is already in progress.

In the light of the results we can conclude that the new paradigm EBCOA produced promising results in this experiment, sometimes giving better and comparable results to GAs and EDAs. However, their potential is still to be analysed, as there are still many different aspects that need to be tested and could result in a considerable improvement in the performance of these algorithms.

5. Conclusions and Further Work

This paper introduces for the first time a new paradigm, Evolutionary Bayesian Classifier-based Optimization Algorithms (EBCOAs), which combines both evolutionary computation techniques and Bayesian classifiers in order to solve optimization problems. The theoretical foundations and the generic pseudocode have been introduced for this new evolutionary computation paradigm. This paper also illustrates the behaviour of these algorithms in standard optimization problems in discrete domains such as HIFF, IsoPeak and IsoTorus.

The first results obtained in these experiments show that some of the choices (such as the instantiation of variables not present in the Bayesian classifier) have to be revised and more techniques have to be tried. However, the fact that some EBCOAs perform in these problems in a similar way and even outperform in some cases EDAs and GAs is a promising result to encourage further testing. This experiment was performed with general objective functions, and further testing should be done with more complex problems and using EBCOAs that can take into account higher-order dependencies between variables. We reckon that the application of more complex EBCOAs to these problems should turn out to improve the performance of even EDAs and GAs.

Future research trends also include the study and experimentation of new Bayesian network classifiers that are capable of taking into account more interdependencies than the ones introduced here. An example of possible structures to apply are the generalization of structures from the EBNA approach in problems where the dependencies between variables are high. Another future research topic for EBCOAs also includes applying classification techniques for building statistical probabilistic graphical models in continuous domains so that we can compare their performance with continuous EDAs and other evolutionary approaches in continuous domains.

Acknowledgments

This work was partly supported by the University of the Basque Country within the project 9/UPV-EHU 00140.226-15334/2003. The authors would also like to thank J.A. Lozano, Iñaki Inza, and the other members of the Intelligent Systems Group of the University of the Basque Country for their useful advise and contribution to this work.

References

Baluja S. (1994): *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning*. — Techn. Rep., Carnegie Mellon, CMU-CS-94-163.

Baluja S. and Davies S. (1997): *Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space*. — Techn. Rep., Carnegie Mellon, CMU-CS-97-107.

Cantu-Paz E. (2001): *Supervised and unsupervised discretization methods for evolutionary algorithms*. — Proc. *Genetic and Evolutionary Computation Conference (GECCO'2001)*, Workshop *Optimization by Building and Using Probabilisitic Models*, San Francisco, California, pp. 213–216.

Cestnik B., Kononenko I. and Bratko I. (1987): *ASSISTANT-86: A knowledge elicitation tool for sophisticated users*, In: Progress in Machine Learning (I. Bratko and N. Lavrac, Eds.). — Wilmslow, U.K.: Sigma Press, pp. 31–45.

Cheng J. and Greiner R. (1999): *Comparing Bayesian network classifiers*. — Proc. 15th Conf. *Uncertainty in Artificial Intelligence*, San Francisco, CA: Morgan Kaufmann Publishers, pp. 101–107.

Chow C. and Liu C. (1968): *Approximating discrete probability distributions with dependence trees*. — IEEE Trans. Inf. Theory, Vol. 14, No. 3, pp. 462–467.

de Bonet J.S., Isbell C.L. and Viola P. (1997): *MIMIC: Finding optima by estimating probability densities*, In: Advances in Neural Information Processing Systems (M. Mozer, M. Jordan and Th. Petsche, Eds.). — Cambridge, MA: The MIT Press, Vol. 9, pp. 424–431.

Domingos P. and Pazzani M. (1997): *On the optimality of the simple Bayesian classifier under zero-one loss*. — Mach. Learn., Vol. 29, No. 2–3, pp. 103–130.

Duda R. and Hart P. (1973): *Pattern Classification and Scene Analysis*. — New York: Wiley.

Etxeberria R. and Larrañaga P. (1999): *Global optimization with Bayesian networks*. — Proc. 2nd Symp. *Artificial Intelligence, CIMAF99*, La Habana, Cuba, pp. 332–339.

Friedman N., Geiger D. and Goldsmidt M. (1997): *Bayesian network classifiers*. — Mach. Learn., Vol. 29, No. 2, pp. 131–163.

Gammerman A. and Thatcher A.R. (1991): *Bayesian diagnostic probabilities without assuming independence of symptoms*. — Meth. Inf. Medic., Vol. 30, No. 1, pp. 15–22.

Goldberg D.E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning*. — Reading: Addison-Wesley.

Harik G. (1999): *Linkage learning via probabilistic modeling in the EcGA*. — Techn. Rep., University of Illinois, Urbana, IlliGAL Report No. 99010.

Harik G., Lobo F.G. and Golberg D.E. (1998): *The compact genetic algorithm*. — Proc. IEEE Conf. *Evolutionary Computation*, Piscataway, NJ, pp. 523–528.

Holland J.H. (1975): *Adaptation in Natural and Artificial Systems*. — Michigan: The University of Michigan Press.

Inza I., Larrañaga P., Etxeberria R. and Sierra B. (2000): *Feature subset selection by Bayesian network-based optimization*. — Artif. Intell., Vol. 123, No. 1–2, pp. 157–184.

Kaufman K. and Michalski R. (1999): *The AQ18 machine learning and data mining system: An implementation and user's guide*. — Techn. Rep., Machine Learning and Inference Laboratory, George Manson University, Fairfax, Virginia.

Kohavi R. and John G. (1997): *Wrappers for feature subset selection*. — Artif. Intell., Vol. 97, No. 1–2, pp. 273–324.

Kononenko I. (1990): *Comparison of inductive and naïve Bayesian learning approaches to automatic knowledge acquisition*, In: Current Trends in Knowledge Acquisition (B. Wielinga, J. Boose, B. Gaines, G. Shereiber and M. van Someren, Eds.). — Amsterdam: IOS Press, pp. 190–197.

Kononenko I. (1991): *Semi-naïve Bayesian classifiers*. — Proc. 6th *Europ. Working Session on Learning*, Porto, Portugal, pp. 206–219.

Kontkanen P., Myllymäki P., Tirri H. and Valtonen K. (2000): *Bayesian multinet classifiers*. — Proc. 10th Int. Conf. *Computing and Information (ICCI'2000)*.

Langley P. and Sage S. (1994): *Induction of selective Bayesian classifiers*. — Proc. 10th Conf. *Uncertainty in Artificial Intelligence*, Seattle, WA, pp. 399–406.

Larrañaga P. and Lozano J.A. (2001): *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. — Kluwer.

Liu H. and Motoda H. (1998): *Feature Selection. From Knowledge Discovery and Data Mining*. — Boston: Kluwer.

Llorà X. and Goldberg D.E. (2003): *Wise Breeding GA via Machine Learning Techniques for Function Optimization*. — Proc. *Genetic and Evolutionary Computation Conference GECCO-03*, Part I, Chicago, Illinois, pp. 1172–1183.

Michalewicz Z. (1992): *Genetic Algorithms $+$ Data Structures $=$ Evolution Programs*. — Berlin: Springer Verlag.

Michalski R.S. (2000): *Learnable execution model: Evolutionary processes guided by machine learning*. — Mach. Learn., Vol. 38, pp. 9–40.

Minsky M. (1961): *Steps toward artificial intelligence*. — Trans. Inst. Radio Eng., Vol. 49, pp. 8–30.

Mühlenbein H. (1998): *The equation for response to selection and its use for prediction*. — Evolut. Comput., Vol. 5, No. 3, pp. 303–346.

Mühlenbein H. and Mahning T. (1999): *FDA—A scalable evolutionary algorithm for the optimization of additively decomposed functions*. — Evolut. Comput., Vol. 7, No. 4, pp. 353–376.

Mühlenbein H., Mahning T. and Ochoa A. (1999): *Schemata, distributions and graphical models in evolutionary optimization*. — J. Heurist., Vol. 5, pp. 215–247.

Mühlenbein H. and Paaß G. (1996): *From recombination of genes to the estimation of distributions, I. Binary parameters*, In: Parallel Problem Solving from Nature—PPSN IV (M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel, Eds.). — Lecture Notes in Computer Science 1411, Berlin: Springer, pp. 178–187.

Muñoz A. (2003): *LEM algorithms*. — Final year project, Computer Engineering Faculty, University of the Basque Country, (in Spanish).

Neapolitan R.E. (2003): *Learning Bayesian Networks*. — Prentice Hall.

Ohmann C., Yang Q., Kunneke M., Stolzing H., Thon K. and Lorenz W. (1988): *Bayes theorem and conditional dependence of symptoms: Different models applied to data of upper gastrointestinal bleeding*. — Meth. Inf. Medic., Vol. 27, No. 2, pp. 73–83.

Pazzani M. (1997): *Searching for dependencies in Bayesian classifiers*, In: Learning from Data: Artificial Intelligence and Statistics V (D. Fisher and H.-J. Lenz, Eds.). — New York: Springer, pp. 239–248.

Pelikan M., Goldberg D.E. and Cantú-Paz E. (1999): *BOA: The Bayesian optimization algorithm*. — Proc. *Genetic and Evolutionary Computation Conference GECCO-99*, Orlando, pp. 525–532.

Pelikan M. and Mühlenbein H. (1999): *The bivariate marginal distribution algorithm*, In: Advances in Soft Computing-Engineering Design and Manufacturing (R. Roy, T. Furuhashi and P.K. Chandhory, Eds.). — London: Springer-Verlag, pp. 521–535.

Sahami M. (1996): *Learning limited dependence Bayesian classifiers*. — Proc. 2nd Int. Conf. *Knowledge Discovery and Data Mining*, Portland, OR, pp. 335–338.

Santana R. (2004): *Estimation of distribution algorithms with Kikuchi approximations*. — (submitted).

Syswerda G. (1993): *Simulated crossover in genetic algorithms*, In: Foundations of Genetic Algorithms (L.D. Whitley, Ed.). — Vol. 2, San Mateo: Morgan Kaufmann, pp. 239–255.

Thierens D. and Bosman P.A.N. (2001): *Multi-objective mixture-based iterated density estimation evolutionary algorithms*. — Proc. *Genetic and Evolutionary Computation Conference, GECCO'2001*, San Francisco, pp. 663–670.

Todd B.S. and Stamper R. (1994): *The relative accuracy of a variety of medical diagnostic programs*. — Meth. Inf. Medic., Vol. 33, No. ??, pp. 402–416.

Ventura S., Herrera F., Berná J.N. and Hervás C. (2002): *Evolución guiada mediante aprendizaje. Comparación en problemas de optimización*. — Proc. Conf. *Algoritmos Evolutivos y Bioinspirados, AEB'02*, pp. 430–436, (in Spanish).

Whitley D. and Kauth J. (1988): *GENITOR: A different genetic algorithm*. — Proc. *Rocky Mountain Conf. Artificial Intelligence*, USA, Vol. 2, pp. 118–130.

Zitzler E., Deb K. and Thiele L. (1999): *Comparison of multiobjective evolutionary algorithms on test functions of different difficulty*. — Proc. 1999 *Genetic and Evolutionary Computation Conf.*, Orlando, FL, pp.121–122.