Connection Science

ISSN: 0954-0091 (Print) 1360-0494 (Online) Journal homepage: www.tandfonline.com/journals/ccos20

Combining Bayesian classifiers and estimation of distribution algorithms for optimization in continuous domains

Teresa Miquélez, Endika Bengoetxea, Alexander Mendiburu & Pedro Larrañaga

To cite this article: Teresa Miquélez, Endika Bengoetxea, Alexander Mendiburu & Pedro Larrañaga (2007) Combining Bayesian classifiers and estimation of distribution algorithms for optimization in continuous domains, Connection Science, 19:4, 297-319, DOI: 10.1080/09540090701725524

To link to this article: https://doi.org/10.1080/09540090701725524

Connection Science
Vol. 19, No. 4, December 2007, 297–319

Taylor & Francis
Taylor & Francis Group

Combining Bayesian classifiers and estimation of distribution algorithms for optimization in continuous domains

TERESA MIQUÉLEZ, ENDIKA BENGOETXEA, ALEXANDER MENDIBURU and PEDRO LARRAÑAGA*

Intelligent Systems Group, Computer Engineering Faculty, University of the Basque Country, San Sebastian, Spain

This paper introduces a evolutionary computation method that applies Bayesian classifiers to optimization problems. This approach is based on Estimation of Distribution Algorithms (EDAs) in which Bayesian or Gaussian networks are applied to the evolution of a population of individuals (i.e. potential solutions to the optimization problem) in order to improve the quality of the individuals of the next generation. Our new approach, called Evolutionary Bayesian Classifier-based Optimization Algorithm (EBCOA), employs Bayesian classifiers instead of Bayesian or Gaussian networks in order to evolve individuals to a fitter population. In brief, EBCOAs are characterized by applying Bayesian classification techniques – usually applied to supervised classification problems – to optimization in continuous domains. We propose and review in this paper different Bayesian classifiers for implementing our EBCOA method, focusing particularly on EBCOAs applying naïve Bayes, semi-naïve Bayes, and tree augmented naïve Bayes classifiers. This work presents a deep study on the behavior of these algorithms with classical optimiztion problems in continuous domains. The different parameters used for tuning the performance of the algorithms are discussed, and a comprehensive overview of their influence is provided. We also present experimental results to compare this new method with other state of the art approaches of the evolutionary computation field for continuous domains such as Evolutionary Strategies (ES) and Estimation of Distribution Algorithms (EDAs).

1. Introduction

Evolutionary computation approaches, which are based on storing more than a solution every iteration, have undergone an important development with paradigms such as genetic algorithms (GAs) (Holland 1975, Goldberg 1989), evolutionary strategies (ES) (Rechenberg 1973, Hansen *et al.* 2003, Hansen 2006), and estimation of distribution algorithms (EDAs) (Larrañaga and Lozano 2001, Lozano *et al.* 2006, Paaß 1996, Mühlenbein *et al.* 1999, Pelikan *et al.* 1999, Pelikan 2005, Sebag and Ducoulombier 1998). These are examples of classical approaches that have been the object of a lot of experimentation and literature for many years. Also, many other evolutionary computation paradigms have been proposed in recent years, in which the main ideas of these three paradigms are combined together with other techniques that are able to consider other types of relationships and combinations among individuals. Examples of the latter are the learnable evolution model (LEM) (Michalski 2000),

*Corresponding author. Email: pedro.larranaga@ehu.es

Connection Science
ISSN 0954-0091 print/ISSN 1360-0494 online
http://www.tandf.co.uk/journals
DOI: 10.1080/09540090701725524

and the evolutionary Bayesian classifier-based optimization algorithms (EBCOAs) (Miquélez *et al.* 2004).

The main difference between all these evolutionary computation paradigms is the way of improving the set of solutions, the so-called population of individuals, in order to obtain fitter solutions to a concrete optimization problem. In GAs and ES the evolution is based on using crossover and mutation operators, without explicitly expressing the characteristics of the selected individuals within a population. EDAs take into account these explicit characteristics by considering the interdependencies between the different variables that form an individual, and by learning a probabilistic graphical model to represent them.

The approach of LEM, EBCOAs, and other similar methods in this direction (Llorà and Goldberg 2003, Muñoz 2003) have the distinctive characteristic of applying classification techniques to build models. These models represent the main features that determine individuals in the same population to be among the fittest or less fit ones. Nevertheless, the added value of these paradigms is that they are able to also take into consideration less fit individuals of the population in order to enhance and estimate the differences between the best and worst approaches. This knowledge is later used for instantiating new individuals for the next generation.

EDAs and EBCOAs can also be regarded as new hybrid soft computing paradigms (Bonissone *et al.* 1999) where probabilistic reasoning is combined with evolutionary computing. In this case, Bayesian networks and Bayesian classifiers are used respectively to evolve a generation of solutions to a fittest one.

The main motivation to develop EBCOAs was to try to overcome some performance drawbacks of EDAs. In most optimization problems EDAs obtain quite positive results, although there are some other specific results for which EDAs converge too fast leading to premature convergence to local optima. We believe that this behavior could be avoided by paying more attention to the difference in fitness of the individuals in the population. In that sense, in the graphical probabilistic model of EDAs (a Gaussian network (Shachter and Kenley 1989) in continuous domains) only the values of the predictor variables of selected individuals are taken into account, but once selected, the learning step of EDAs does not distinguish individuals regarding their relative fitness value[†]. This learning step is crucial in EDAs since it determines the convergence success of the whole search, and we deem that the search process could be considerably improved if the relative fitness value of each of the selected individuals is also reflected in the learning of the probabilistic graphical model. We propose to do this by adding a discrete variable to our population. This will enable us to group all the individuals in the population regarding their performance measured by their respective fitness value. In EBCOAs, individuals are classified – in different subgroups – according to their fitness, and this information is used to generate a probabilistic graphical model by applying a Bayesian classifier. Following this approach, EBCOAs are able to evolve a fitter generation by constructing models that take into account more aspects than simply a subset of the fittest individuals.

This paper introduces EBCOAs (Evolutionary Bayesian Classifier-Based Optimization Algorithm) for continuous domains. Continuous EBCOAs generalize the approach presented in (Miquélez *et al.* 2004) that was only applicable to discrete domains by applying classification techniques in the form of conditional Gaussian networks to improve a population of individuals every generation. Furthermore, this paper also provides a comprehensive study of the behavior of EBCOAs on classical continuous optimization problems depending on how they are parameterized, and compares their performance to other optimization algorithms in the evolutionary computation field.

[†]Except from some especial cases such as bit-based simulated crossover (Syswerda 1993).

The paper is organized as follows: the next section describes the different classification techniques in the form of conditional Gaussian networks that will be applied in the learning step of EBCOAs. Section 3 describes EBCOA in continuous domains paying special attention to the step of learning the classification model which is the core of the whole algorithm, as this step is the key for a proper evolution to fitter individuals. In this step, the classifiers reviewed in the previous section are constructed taking into account not only the dependencies between the different variables that form an individual, but also the respective fitness values of each individual in the population. Section 4 describes the experiment that we have carried out in classical optimization problems in the continuous domain, presenting a comprehensive study of the behavior of this new method, and compares the results obtained by other well known evolutionary computation techniques. Finally, section 5 provides the conclusions and future work perspectives on this new method.

2. Bayesian classifiers for continuous domains

Literature reviews suggest several examples of classifiers combined with evolutionary computation techniques. One of the first examples is the LEM algorithm (Michalski 2000) which makes use of rules to build classifiers that record the main differences between the groups of best and worst individuals of each population.

In EBCOAs, the classifiers that are used for the same purpose are based on the learning of probabilistic graphical models. More specifically, in the case of continuous domains, we apply Bayesian classifiers based on conditional Gaussian networks (Lauritzen and Wermuth 1989). This section starts with a formal description of the supervised classification problem, followed by a revision of classifiers of this type that have been proposed in the literature for continuous domains so that they can be applied to continuous EBCOAs.

2.1 *The supervised classification problem*

The *supervised classification* problem with n continuous predictor variables consists of assigning any vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{R}^n$ to one of the $|C|$ classes of a class variable C. The class value is denoted by c and, therefore, we have that $c \in \{1, 2, \ldots, |C|\}$. As a result, a classifier in supervised classification is defined as a function $\gamma : (x_1, \ldots, x_n) \to \{1, 2, \ldots, |C|\}$ that assigns class labels to observations.

The criterion to compare classifiers depends on the use of a *cost function* that measures the cost of any misclassified vector. In the specific case of supervised classification, the $0/1$ loss function is defined as the function that considers 1 to be the cost of misclassifying an element –and 0 if the classification is correct. When such a loss function $0/1$ applies Duda and Hart (1973) proved that the Bayesian classifier that minimizes the total misclassification error cost is the one that assigns to the example $\boldsymbol{x} = (x_1, \ldots, x_n)$ the class with the highest *a posteriori* probability, that is

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

All the Bayesian classifiers reviewed in this section were originally proposed for discrete supervised classification problems, and in this paper we show how to adapt them to continuous domains. Note that our aim when choosing a Bayesian classifier for EBCOAs is to have a relatively effective learnable algorithm that can be built in a reasonable time, since it is to be applied every iteration. That is why a balance between effectiveness and induction complexity is considered when deciding the type of classifier to be used in EBCOAs.

Similarly, as with continuous EDAs, continuous EBCOAs can be categorized depending on the order of the Bayesian classifier applied, *i.e.* depending on the maximum number of dependencies between variables that the classifier is able to take into account. The resulting classifier has the form of a conditional Gaussian network.

2.2 *Naïve Bayes*

The Bayesian classifier that considers all the variables $X_1, \ldots, X_n$ to be conditionally independent given the class value C is known as naïve Bayes (Minsky 1961). In this case, the probabilistic graphical model can be considered to be a fixed structure as illustrated in figure 1(a). In continuous domains it is usual to assume that the joint density function follows a n-dimensional normal distribution, and because independence between the variables – given the class variable C – is assumed, this is factorized by a product of one-dimensional and conditionally independent normal densities. Therefore, when classifying a new individual using the naïve Bayes classifier we have:

$$p(C = c|X_1 = x_1, \ldots X_n = x_n) \propto p(c) \cdot f(x_1|c) \cdot f(x_2|c) \cdot \ldots \cdot f(x_n|c)$$

where

$$f(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{ic}} e^{-1/2((x_{ic}-\mu_{ic})/\sigma_{ic})^2}$$

for all $i = 1, \ldots, n$ and $c = 1, \ldots, |C|$ with μ_{ic} and σ_{ic} representing the mean and the standard deviation of $X_i|C = c$, respectively.

The main advantage of naïve Bayes is that they have a fixed structure, which simplifies the learning process to uniquely estimate the probabilities that are to be considered following this conditional Gaussian network. In order to apply a naïve Bayes classifier in EBCOAs, the estimation of the *a priori* probability of the class, $p(c)$, as well as the parameters μ_{ic} and σ_{ic} of the conditional density functions, $f(x_i|c)$, are carried out from the database of selected individuals at each generation.

Note that this Bayesian classifier is more suited for optimization problems in which the variables are indeed independent given the class.

2.3 *Semi-naïve Bayes*

The semi-naïve Bayes classifier (Kononenko 1991) provides more complexity than the former since it takes into account dependencies between groups of variables. This method represents the variables found to be related as a *merged* node in the conditional Gaussian network,

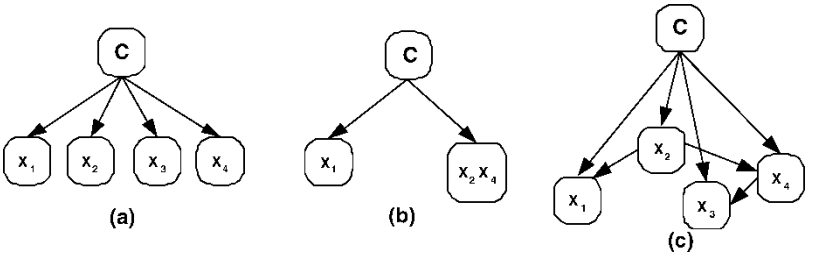

Figure 1. Example of structures of Bayesian classifiers that can be obtained as a result of the different classification model building algorithms, in a problem with four variables $X_1, \ldots, X_4$ and the class-variable C: (a) naïve Bayes, (b) semi-naïve Bayes, and (c) tree augmented naïve Bayes.

that is the semi-naïve Bayesian classifier proposed to group some variables in a single node of the structure. Figure 1(b) illustrates the structure of a semi-naïve Bayesian classifier for a problem with four variables, treating each of these grouped variables as a single super-variable regarding the factorization of the probability distribution. When grouping variables, all the inter-dependencies between them are taken into account implicitly in the Bayesian classifier. Note that as illustrated in figure 1(b), in a semi-naïve Bayesian classifier it is also possible to ignore some variables and therefore not to include them in the final probabilistic graphical model, which has the effect of considering these variables not to be relevant for labeling vectors.

Compared to the naïve Bayes classifier, the semi-naïve Bayes mode requires an additional step in its construction in order to learn the most suited probabilistic graphical model structure following this approach. Pazzani (1997) introduced a greedy algorithm to detect irrelevant as well as dependent variables (susceptible to being grouped) and to propose variables that are likely to be ignored in a Bayesian classifier, although this is described only for discrete domains. We propose to adapt it to the case of continuous domains by grouping dependent continuous variables as a single multidimensional variable in the form of one node in the conditional Gaussian network. The resultant pseudocode of the algorithm that is applied to build a semi-naïve Bayes classifier model for continuous domains is illustrated in figure 2.

Considering that the example in figure 1(b) is a semi-naïve Bayes model structure learned for a particular supervised classification problem, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the following class:

$$c^* = \arg\max_c p(c) f(x_1|c) f(x_2, x_4|c) \tag{2}$$

Following this approach, in cases in which a variable X_i is estimated to be conditionally independent from the rest given the class variable (such as variable X_1 in the latter example), $f(x_i|c)$ will be computed as

$$f(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{ic}} e^{-1/2((x_{ic}-\mu_{ic})/\sigma_{ic})^2}$$

Analogously, for the case of dependencies with a number p of grouped variables over a single node in the structure, similar to variables X_2 and X_4 in our example, the corresponding factor will be assumed to follow a p-dimensional normal distribution for each value of variable C. Considering that $\mathbf{z}_j$ represents the jth group of p variables, we have

$$f(\mathbf{z}_j|c) = \frac{1}{\sqrt{(2\pi)^p |\Sigma_{jc}|}} e^{-1/2(\mathbf{z}_j-\boldsymbol{\mu}_{jc})\Sigma_{jc}^{-1}(\mathbf{z}_j-\boldsymbol{\mu}_{jc})^T}.$$

where Σ_{jc} is a $p \times p$ matrix representing the variance–covariance matrix of the jth group of p variables when $C = c$ and $\boldsymbol{\mu}_{jc}$ denotes its corresponding expectation.

Initialize the set of variables to be used in the empty set
Classify all the examples as being of the 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, assuming a multinormal density
  Evaluate each possible option by means of the estimation of the percentage of well classified cases
Until no improvement can be obtained

Figure 2. Pseudocode of the algorithm to build semi-naïve Bayes models.

2.4 *Tree augmented naïve Bayes*

Another Bayesian classifier that takes into account dependencies between variables is the tree augmented naïve Bayes classifier (Friedman *et al.* 1997). Its name comes from the fact that structures obtained as a result of its learning approach have the form of a tree. The corresponding pseudocode is illustrated in figure 3. This two-step algorithm constitutes an adaptation of the Chow-Liu algorithm (Chow and Liu 1968) for predictor continuous variables that apply the following expression to estimate the mutual information between two univariate normal distributions conditioned to the class (Cover and Thomas 1991):

$$I\left(X_i, X_j \mid C\right) = -\frac{1}{2}\sum_{c=1}^{|C|} p(c)\log\left(1-\rho_c^2(X_i, X_j)\right) \tag{3}$$

where $i < j$ and $j = 2, \ldots, n$.

Figure 1(c) shows the type of structures that could be obtained when applying the tree augmented naïve Bayes algorithm to a similar problem as for the two previous Bayesian classifiers. Following this particular example, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the class

$$c^* = \arg\max_c p(c) f(x_1|c, x_2) f(x_2|c) f(x_3|c, x_4) f(x_4|c, x_2) \tag{4}$$

Note that in this case, the calculation of $f(x_i|c, x_{k(i)})$, where $X_{k(i)}$ represents the predictor parent variable of variable X_i in the case that this parent exists, can be computed as

$$f(x_i|c, x_{k(i)}) = \frac{p(c)\cdot f(x_i, x_{k(i)}|c)}{p(c)\cdot f(x_{k(i)}|c)} = \frac{f(x_i, x_{k(i)}|c)}{f(x_{k(i)}|c)}$$

3. Description of evolutionary Bayesian classifier-based optimization algorithm (EBCOAs)

This section introduces the continuous evolutionary optimization algorithms EBCOAs (Evolutionary Bayesian Classifier-based Optimization Algorithms) which are based on combining Bayesian classifiers such as the ones described in the previous section, together with evolutionary computation. The justification for this approach is to provide a new mechanism to solve an optimization problem. In each generation this population evolves towards a fitter one in a completely different manner to Genetic algorithms (GAs) or Estimation of Distribution Algorithms (EDAs). In GAs the evolution towards a fitter population of individuals

Calculate $I(X_i, X_j \mid C) = -\frac{1}{2}\sum_{c=1}^{|C|} p(c)\log\left(1-\rho_c^2(X_i, X_j)\right)$, with $i < j$, $j = 2, \ldots, n$
where $\rho_c^2(X_i, X_j) = \frac{Cov_c(X_i, X_j)}{\sqrt{Var_c X_i Var_c X_j}}$ is the coefficient of correlation between X_i and X_j when $C = c$
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 naïve Bayes structure adding a node labelled as C, and later add one arc from C to each of the predictor variables X_i $(i = 1, \ldots, n)$

Figure 3. Pseudocode of the *tree augmented naïve Bayes* algorithm for continuous predictors.

is performed by applying crossover and mutation operations, while in EDAs the evolution is based on learning a probabilistic graphical model built uniquely considering a subset of selected fit individuals. On the other hand, in EBCOAs this evolution is performed by building a Bayesian classifier model which considers the characteristics between the best and worst individuals of each population. That is, in EDAs the result of the learning process is a Gaussian network – in continuous domains – or a Bayesian network – in discrete domains – while in EBCOAs the learning follows a very different approach by building a Bayesian classifier. As a result, in EBCOAs the less fit individuals are also taken into consideration when building the probabilistic graphical model.

Figure 4 illustrates the EBCOA approach.

(1) First, the initial population D_0 formed by R individuals is generated. The generation of these R individuals is usually done by assuming an uniform distribution on each variable, and next each individual is evaluated.
(2) Second, the population D_l is subdivided in a number $|K|$ of classes depending on the relative fitness value of the individuals on that population. The result of this step is denoted by D_l^K.
(3) Third, as differences between the individuals of the very close k classes of $k \in K$ are usually not big enough to allow a clear classifier to be built, a subset $C \subseteq K$ of those is selected and the rest are simply ignored for the next steps. Therefore, this is a *class-selection* step. The resultant population is denoted by D_l^C.
(4) Fourth, the n-dimensional Bayesian classifier that better reflects the differences between the different classes (usually between the fittest and the less fit ones) is built. The structure

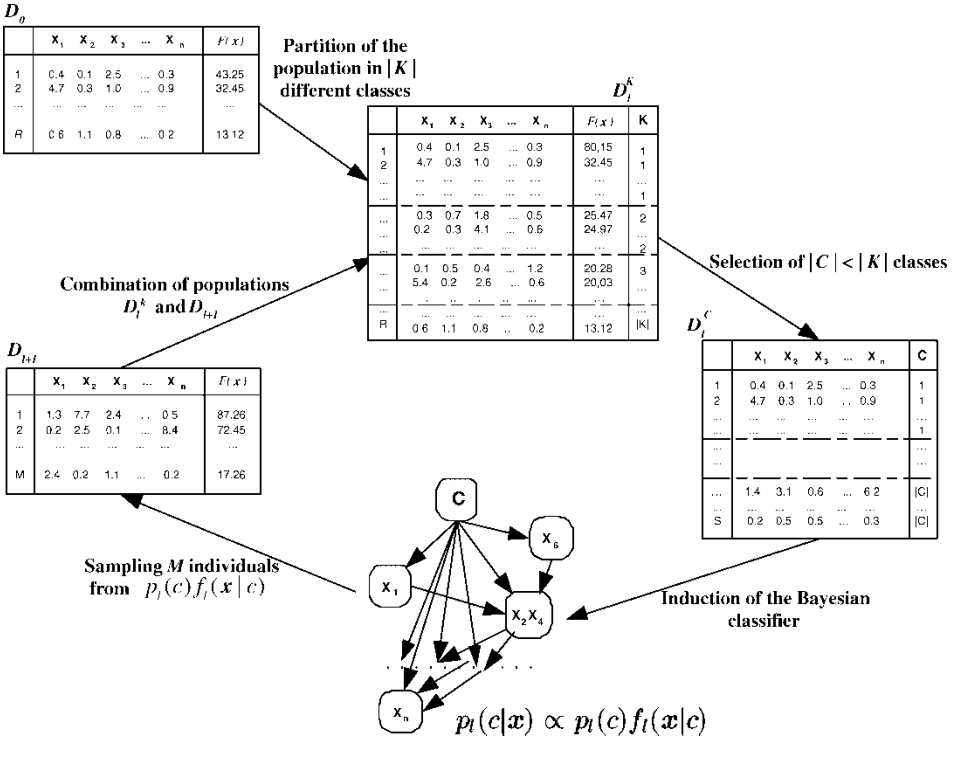

Figure 4. Description of the optimization process in EBCOAs in continuous domains.

of this Bayesian classifier contains the C variable as the parent of the rest, apart from all or some of the predictor variables $X_1, X_2, \ldots, X_n$.

(5) Finally, the new population D_{l+1} constituted by M new individuals is obtained by carrying out the simulation of the probability distribution learnt in the previous step. This new population D_{l+1} is combined with some representative individuals from the different classes of D_l^K in order to obtain a total of R individuals for the next generation. The latter is a *population-combination* step.

Steps 2, 3, 4, and 5 are repeated until a stopping condition is verified. The pseudocode of the EBCOA approach for continuous domains as described here is shown in figure 5. We next review some notation as well as each of these steps in detail.

3.1 *Notation*

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a n-dimensional random variable and $\boldsymbol{x} = (x_1, \ldots, x_n)$ one of its possible instantiations – that is, one of the possible individuals. The probability of a single variable $\boldsymbol{X}$ to take one of its values $\boldsymbol{x}$ will be denoted $p(\boldsymbol{X} = \boldsymbol{x})$ – 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 population of the lth generation that has to evolve to the next generation $(l + 1)$. In EBCOA, as a preliminary step to the learning of the Bayesian classifier, individuals are ordered by fitness value and they are divided in a fixed number of $|K|$ different classes, assigning to each individual a k label where $1 \leq k \leq |K|$ and K being the class variable. We denote by D_l^K the population after it has been subdivided in $|K|$ classes.

Since all the classes are not usually relevant for the learning due to slight differences among some of them, previously to learning the Bayesian classifier we choose $|C| \leq |K|$ classes and the rest are simply ignored for learning purposes. Let D_l^C be 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 aim of this learning step is to build a probabilistic graphical model – that is, a conditional Gaussian network in the continuous domain. In EBCOA, this conditional Gaussian network is a Bayesian classifier that takes into account the variables $X_1, X_2, \ldots, X_n$ as well as the variable C.

3.2 *The supervised classification step: labeling the individuals and the class-selection process*

In contrast to EDAs in which a subset of individuals (usually the fittest) are selected previously to the learning step to ignore the rest, in EBCOAs all the population is initially classified in

$D_0 \leftarrow$ Generate R individuals (the initial population) randomly

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

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

$D_l^C \leftarrow$ Select the $|C| \leq |K|$ classes of D_l^K that will be used for building the Bayesian classifier in continuous domains, usually taking into account at least the best and worst classes. The individuals on $D_l^K \setminus D_l^C$ are ignored

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

$D_{l+1} \leftarrow$ Sample M new individuals using $f_l(\boldsymbol{x}|c)$ according to $p_l(c)$

Figure 5. Pseudocode for the EBCOA approach in continuous domains.

a fixed number $|K|$ of different classes. These classes are formed by ordering the whole population in groups of individuals from the fittest ones to the less fit ones. This procedure assigns to each individual in D_l a label $k \in \{1, 2, \ldots, |K|\}$ such that each class k contains the same number of individuals. As a result, we obtain the population D_l^K.

As the main motivation for EBCOAs is to take into account in the evolution of the population the main characteristics that make individuals potentially fit or not, some of the classes in D_l^K could be ignored to facilitate the learning by enhancing these differences. An example is to ignore the middle classes in D_l^K for the learning of the Bayesian classifier, so that the classifier is induced taking into account the most distant classes. This procedure is called class-selection. D_l^C is the result of removing from D_l^K the individuals labeled to classes that are ignored for the learning, and C is the class variable that is used for the learning as the root of the Bayesian classifiers, with $|C| \leq |K|$.

3.3 *The learning step: building the Bayesian classifier*

The most difficult step for EBCOAs is actually to estimate satisfactorily the probability distribution $p_l(c|\boldsymbol{x})$, as the computation of all the parameters needed for obtaining optimum classifiers in the form of a conditional Gaussian network is very time consuming (Pérez *et al.*, 2006). Bearing in mind that our aim is to be able to distinguish potentially fitter individuals over the ones that are clearly worse, 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 that will be ignored in the next generations. That is why several approximations propose to factorize the probability distribution according to a Bayesian classifier that is able to perform relatively well in a reasonable computation time.

The learning step in EBCOAs is performed by applying an algorithm to induce a Bayesian classifier in the form of a conditional Gaussian 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 variables X_1 to X_n can also be present. This conditional Gaussian network will be formed following different classifier induction algorithms such as the ones described in the previous section, and it will contain a maximum of $n + 1$ nodes (variables X_1 to X_n and C), with the variable C always being the root and the parent of all the others present in this probabilistic graphical model. As a result of this learning procedure, the probability distribution can be represented by a factorization of the form $p_l(c|\boldsymbol{x}) \propto p_l(c)\, f_l(\boldsymbol{x}|c)$.

We will use the following notation to describe the different types of EBCOAs depending on the Bayesian classifier that they apply in their learning phase: EBCOA_{NB} (for the one based on naïve Bayes), $\text{EBCOA}_{\text{TAN}}$ (for the one based on the tree-augmented network), and so on.

The complexity of the Bayesian classifier that is chosen for the EBCOA, that is, the degree of interdependencies between the different variables $X_1, X_2, \ldots, X_n$, and C determines the capability of the Bayesian classifier to better reflect the main differences between the fitter and less fit individuals of the population.

3.4 *The simulation step: instantiating new individuals for a new population*

The instantiation of the probabilistic graphical model to obtain the next population D_{l+1} is performed in a similar way as in EDAs, although there is an important difference due to the inclusion of the C variable in the conditional Gaussian network. Every individual will be generated using a specific criterion, for instance by means of the probability distribution $f_l(\boldsymbol{x}|c)$ according to $p_l(c)$. In this way, the simulation of the individual is performed following the probability distribution learnt in the previous step.

However, the main difference between EDAs and EBCOAs is that EBCOAs reflect the different characteristics that make individuals among the fittest or the worst classes in the forthcoming generations. That is why it is important that in the next population D_{l+1} individuals from all the most relevant classes in C are present. Our proposal is to generate M new individuals by assigning a different number of representatives per each class $c \in C$ from D_l^C by instantiating the probability distribution of all the classes proportionally to $p(c)$, knowing that in case of a maximization problem we have

$$p(c) \propto \sum_{\boldsymbol{x}|\ \gamma(\boldsymbol{x})=c} F(\boldsymbol{x}) \tag{5}$$

where $F(\boldsymbol{x})$ is the fitness value of the individual $\boldsymbol{x}$, and $\gamma(\boldsymbol{x})$ is the class assigned to the individual $\boldsymbol{x}$ in D_l^C. We denote by D_{l+1} the population of the M newly generated individuals.

After generating D_{l+1}, we combine these M individuals with some individuals of the previous generation's population D_l^K, and discard individuals from this combination in order to obtain a total of R individuals that will form the next generation's population D_{l+1}^K.

The reason for such a complex simulation process is to ensure that individuals from all the classes will be present in the next generation, while allowing at the same time individuals from the fitter classes to have a bigger chance to be present in D_{l+1} according to the need to provide fitter individuals in the next population. Following this procedure, individuals from the less fit classes will also be instantiated every generation, ensuring that the differences between the fittest individuals and the less fit ones are still preserved as the algorithms converge to the optimum solution. The fact of keeping these differences (and not simply concentrating on the fittest individuals) is important since the convergence of EBCOAs is based on the ability of the Bayesian classifier to model the main characteristics that make an individual be among the fittest found in the whole search process.

3.5 *Evolving the populations from one generation to the next*

Once the new population D_{l+1} has been created after instantiation of the Bayesian classifier built in the previous learning step, there are many different options to generate a new population D_{l+1}.

The initial step of creating D_0^K starting from the initial population D_0 is performed by directly assigning the class c to each individual depending on its corresponding individual fitness value in the population, as described in the previous section. This is the only alternative that provides us with the most logical way of building the different classes.

However, in the successive generations we create M new individuals instantiating the Bayesian classifier. At this stage, it is also recommendable to keep track of some of the individuals from the previous generation D_l^K. Note that D_l^C simply serves as an intermediate step for the learning phase of EBCOAs. The inclusion of a minimum of individuals from the previous population D_l^K has the aim of providing some *memory* of the characteristics of individuals from the less promising regions of the search space previously visited in former generations. This fact advises us to combine individuals from D_l^K (where $|D_l^K| = R$) and the newly generated population D_{l+1} (where $|D_{l+1}| = M$) in order to form the population D_{l+1}^K that will constitute the population of the $(l+1)$th generation (also with $|D_{l+1}^K| = R$). Figure 6 shows an example of this process.

On the other hand, the combination of populations should also be done in a way that preserves at least a minimum number of promising individuals both from D_l^K and D_{l+1}. EBCOAs show a better performance when the individuals to be kept in D_{l+1}^K are the ones that best suit the characteristics of each of the classes of K. Note that at every generation the population D_l^K

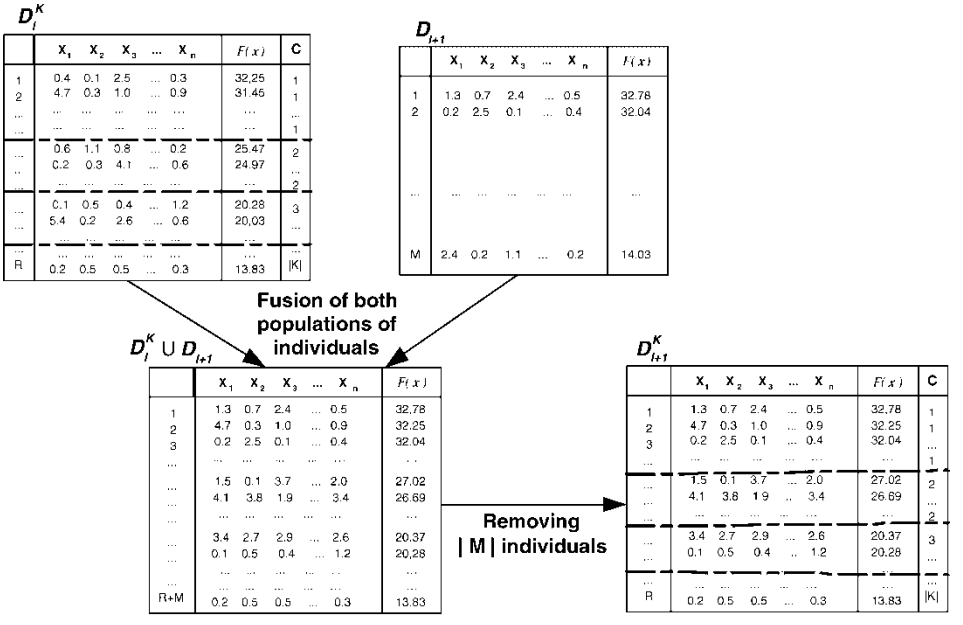

Figure 6. Illustration of the EBCOA step called *population-combination*, which is applied in EBCOAs for providing the transition between D_l^K and D_{l+1} to D_{l+1}^K.

keeps the same size of R individuals, and therefore, each of the $|K|$ classes in which this population is divided at every generation contains the same number of individuals.

It is important to note that the way of combining individuals D_l^K and D_{l+1} in order to form the new population D_{l+1}^K directly influences the performance of the EBCOA, since a total of R individuals of any of these populations will be discarded. Since the convergence of EBCOAs towards a better population is fully dependent on the ability of the learning step to provide a comprehensive structure that represents the differences between the fittest and less fit classes, this convergence will depend on the complexity of the optimization problem to be solved, the number of classes considered ($|K|$ and $|C|$), and the size of the whole population (R). From our experience we concluded that the most advisable practice is to ensure that the best individuals of both D_l^K and D_{l+1} are at least preserved in D_{l+1}^K, in order to ensure that the class of the fittest of the next generations will be at least composed of the best individuals found during the whole search process so far. The latter will ensure at the same time that these individuals will provide the learning algorithm with the information of the best characteristics that exhibit the most promising individuals.

3.6 *Variables that could be omitted in the conditional Gaussian network*

Section 2 describes the semi-naïve Bayes approach as one example that can be applied in EBCOAs and that could induce, as a result of the learning step, a Bayesian classifier in whose structure some of the predictor variables $X_1, X_2, \ldots, X_n$ are not present. 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 the $|C|$ classes are not relevant to distinguish them. This has an important reading, since it does not mean that the value assigned to such variables is not important when classifying the individual. Note that the individual constitutes a point in the search space for a specific problem, and that all the values assigned to the variables are usually relevant for obtaining a fit individual and therefore converge to the optimum solution.

As the search process goes on, it is likely that some variables have the same values in the best and worst classes, and therefore in the learning step of some EBCOAs these will be removed from the Bayesian classifiers.

The way of instantiating variables that are not present in the conditional Gaussian network is another important issue to take into account. Note that the instantiation of new individuals using such a model is a completely different situation than having these variables present but having no arcs in the probabilistic graphical model. When a variable is disconnected it has a density function that has been learned and therefore it has a corresponding estimated $f_l(\boldsymbol{x})$ that allows simulating new individuals.

Since in EBCOAs it is possible to employ Bayesian classifiers that can omit some predictor variables in the conditional Gaussian network (*e.g.*, the semi-naïve Bayes), the question of how to instantiate these in order to generate the values for the corresponding variables in the new M individuals must be addressed. A method is needed to generate new individuals with values in all the variables (including the ones not appearing in the Bayesian classifier). We propose to solve this issue by distinguishing the two different reasons that make predictor variables not present in the structure: *irrelevant variables* (*i.e.*, variables that have in all the classes the same probability distribution) and *redundant variables* (*i.e.*, those which appear to be very similar to other variables that are already present in the probabilistic graphical model and therefore their inclusion in the structure is not relevant to reflect differences in the characteristics of the classes). For the former, we consider that the estimated probability for an irrelevant variable X_i to take its kth value is computed as $f(x_i) = f(x_i^k|c)$. For the latter type of variables, we consider that the probability distribution is uniform.

3.7 *The stopping criterion*

All the previous steps are repeated in EBCOAs until a stopping condition is verified. 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. Experiments on standard optimization problems in continuous domains

A set of different experiments were carried out in order to measure both the behavior and performance of EBCOAs when applied to classical optimization problems in continuous domains. The optimization problems selected for this purpose are the ones proposed in Bengoetxea *et al.*, (2001) to compare evolutionary computation algorithms in continuous domain, namely Ackley (Ackley 1987 and Bäck 1996), Griewangk (Törn and Zilinskas 1989), and Sphere model. These optimization problems are briefly described next.

(1) Ackley: the fitness function of this minimization problem is defined as:

$$F(\boldsymbol{x}) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + \exp(1). \tag{6}$$

hence the optimum value is 0, which is obtained when the individual has all its variables at 0. The problem is defined with $-20 \leq x_i \leq 30,\ i = 1, \ldots, n$.

(2) Griewangk: this is a minimization problem defined as:

$$F(\boldsymbol{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right). \tag{7}$$

where the range of all the components of the individual is $-600 \leq x_i \leq 600,\ i = 1, \ldots, n$, and the fittest individual corresponds to a value of 0 which only can be obtained when all the components of the individual are 0.

(3) Sphere model: this is a simple minimization problem. It is defined following $-600 \leq x_i \leq 600,\ i = 1, \ldots, n$, and the fitness function is:

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2. \tag{8}$$

The fittest individual is the one whose all components are 0, which corresponds to the fitness value 0.

These functions have been chosen for the experimentation phase due to the very different nature that they exhibit. For instance, the Sphere model does not contain any local optimum, while Ackley has many local optima that are bigger in the case of Griewangk. These optimization problems are widely known in the literature and their diverse nature will serve to study the behavior and performance of our new method as well as to compare it with such of EDAs and CMA-ES.

Regarding the parameters of EBCOAs, we chose for all the experiments the following: $|K| = 3$ and $|C| = 2$. The size of each of the three classes of D_l^K was fixed at $R/3$, where R is the size of the population D_l^K. The influence of this choice of parameters in the final result will be left for a future work, and in this paper we concentrate on studying the general behavior of EBCOAs having fixed these.

We next describe how the simulation process is performed in our experiments. As described in section 3.4, it is important to ensure that individuals from each of the C classes of D_l^C are included in D_{l+1}, having to follow the proportionality principle described in equation (5). Since in our particular case we have two classes in D_l^C, the number of individuals to be instantiated by each of the two classes to have M new individuals in D_{l+1} is obtained according to the following proportion: A total of $\bar{F}_H^l/(\bar{F}_H^l + \bar{F}_L^l) \cdot M$ individuals are instantiated from class $c = 1$, and the other $\bar{F}_L^l/(\bar{F}_H^l + \bar{F}_L^l) \cdot M$ will be instantiated from class $c = 1$ for the case of a maximization optimization problem, where $\bar{F}_H^l$ is the mean of the fitness values of the class of the best individuals and $\bar{F}_L^l$ is the less fit one. Both $\bar{F}_H^l$ and $\bar{F}_L^l$ are computed assuming that all the individuals have a positive fitness value, otherwise they would have to be normalized accordingly. Using this procedure we generate in total M new individuals, so that the bigger the difference between $\bar{F}_H^l$ and $\bar{F}_L^l$ the more individuals from the best class ($c = 1$) will be generated. The purpose of this mechanism is that the number generated from the class of best individuals is adapted in order to avoid too direct a convergence and minimize the risk of falling into local optima.

Finally, regarding the type of EBCOAs that we apply in our experiments, we decided to concentrate mainly on the ones based on naïve Bayes and TAN (EBCOA_{NB} and $\text{EBCOA}_{\text{TAN}}$). The former is characterized by the lack of a learning phase, while $\text{EBCOA}_{\text{TAN}}$ contains a learning phase that includes the induction of a new structure every generation. These two EBCOAs will serve as representatives from different learning step types of EBCOAs in our experiments.

4.1 *The choice of class-selection and population-combination procedures*

The aim of setting the number of initial classes to $|K| = 3$ is to enhance the general differences between the fittest and less fit classes, since having set $|C| = 2$ we will consider only two of these three classes for the learning process. Therefore, there are two reasonable ways of proceeding with the class-selection step for the transition from D_l^K to D_l^C:

(1) Ignoring the middle class and keeping for D_l^C only the top and bottom classes. We will represent this choice as CS_{1+3}.
(2) Ignoring the bottom class and keeping for D_l^C the two better ones, which will be represented by CS_{1+2}.

The other possible option (*i.e.*, ignoring the class of the fittest individuals, or CS_{2+3}) will not ensure that the new generation would contain individuals at least as promising as before, and that is why it has not been considered.

The class-selection step is an important parameter of EBCOAs that determines the success of the search process due to its direct link with the learning phase. However, another important parameter that also influences directly both the learning and the simulation phases is the population-combination phase. This is represented in figure 4 as the transition from D_{l+1} to D_{l+1}^K. As explained previously, it is recommended to ensure that a number of individuals from the lth generation will remain in the generation $l + 1$. This practice is beyond the pure elitist approach of GAs (as well as some EDAs) of keeping the best individual of the previous generation, since in EBCOAs it is also necessary in some cases to record some of the less promising individuals. Therefore, we also have the choice of at least three comprehensive options to combine the individuals from D_l^K and D_{l+1} in order to create the population D_{l+1}^C.

4.1.1 Considering only the best and worst classes. This option is illustrated in figure 7a. This provides the most different individuals found during the whole search process in all the previous generations to the learning step in EBCOAs. This option is more likely to be applied with the class-selection CS_{1+3} of D_l^K (the fittest and worst individuals). In doing so, the best and worst individuals found during all the generations would be kept in D_{l+1}^K.

This option has also the characteristic of keeping in all the generations both the best and worst individuals ever found in any of the previous generations. The former individuals are appropriated in all the cases, but the latter would create the drawback, for some EBCOAs,

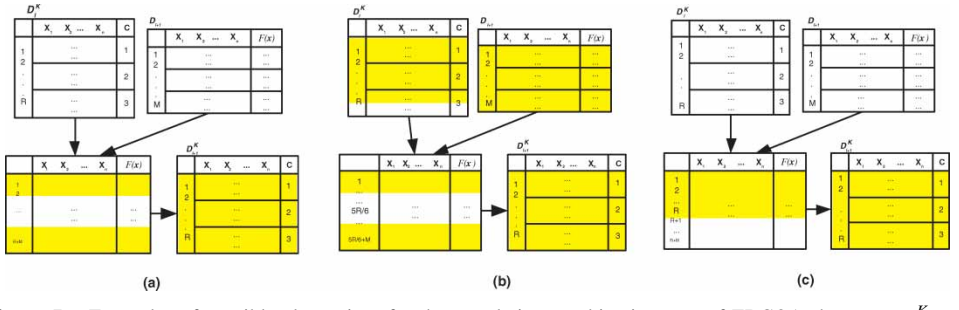

Figure 7. Examples of possible alternatives for the population-combination step of EBCOAs between D_l^K and D_{l+1}. The dark regions represent the individuals that would be present in D_{l+1}^K and the rest would be discarded. In the (a) and (c) options all the individuals of D_l^K and D_{l+1} are considered, whereas in the (b) case the worst $R/6$ individuals from D_l^K are removed from the very beginning before recombination with D_{l+1}.

of confusing the learning step since in most of the complex optimization problems the worst individuals of the search process are visited usually on the first generations. Therefore the characteristics of those EBCOAs would not be representing concrete regions on which search algorithms concentrate as the search process advances.

4.1.2 Keeping worst individuals of recent generations only. The combination of populations in this option is illustrated in figure 7b. This option is similar to the previous one, although in this case the $R/6$ worst individuals of the population from the previous generation D_l^K are not considered for D_{l+1}^K. This removal ensures that the class of the worst individuals in D_{l+1}^K will contain a set of less promising individuals newly generated in D_{l+1}, as well as the best half of the worst class in the previous generation from D_l^K. This adds the notion of *aging* to the worst individuals, while preserving the most promising in the successive generations, in the hope that EBCOAs will be able to concentrate more on specific regions of the search space as the search progresses.

4.1.3 Elitist combination. The two previous options are more appropriated for combination with the CS_{1+3} class-selection type on the transition from D_l^k to D_l^C. However, in some concrete optimization problems it is likely that convergence will be improved if the two top classes are preserved for D_l^C. This would be adequately combined with a combination of populations as illustrated in figure 7c.

In order to illustrate the behavior of EBCOAs, we performed an experiment combining the class-selection and population-combination choices already presented. Table 1 shows the performance of $EBCOA_{NB}$ and $EBCOA_{TAN}$ for every alternative of class-selection and population-combination types. Since the population size plays an important role in the general performance of EBCOAs (section 4.2 provides the explanations with regard to this), this table shows different population sizes for each algorithm in order to provide a fair comparison, showing for each algorithm the population size that obtained the best mean performance for 30 runs in the three optimization problems. As the stopping criterion, a maximum of

Table 1. Mean results after 30 executions with each EBCOA combining class-selection and population-combination methods for D_l^K and D_{l+1} to obtain D_{l+1}^K.

	Ackley				Griewangk				Sphere			
	CS_{1+2}		CS_{1+3}		CS_{1+2}		CS_{1+3}		CS_{1+2}		CS_{1+3}	
	Cv.	E	Cv.	E	Cv.	E	Cv.	E	Cv.	E	Cv.	E
$EBCOA_{NB}$												
Population combination												
Best/worst classes	93	45,615	83	35,749	90	34,030	80	28,201	97	40,000	77	32,931
Best/recent worst	83	44,099	77	39,221	90	33,057	83	30,090	97	38,991	63	35,276
Elitist combination	93	40,746	80	37,886	93	30,690	70	28,818	90	35,932	90	33,588
$EBCOA_{TAN}$												
Population combination												
Best/worst classes	100	56,319	100	30,375	87	55,907	93	28,248	100	47,891	100	26,567
Best/recent worst	100	55,000	100	32,651	80	56,159	97	24,419	100	47,599	100	30,295
Elitist combination	100	40,564	100	31,688	100	39,174	93	29,003	100	34,805	100	27,260

Note: A population size of 200 and 25 was set for $EBCOA_{NB}$ and $EBCOA_{TAN}$, respectively. The Cv. column represents the percentage of convergence, and E the evaluations number for only those runs in which the algorithm converged. A maximum of 60,000 evaluations was allowed.

60,000 evaluations were allowed unless convergence was reached. This serves for providing an appropriated comparison regardless of the different population sizes.

The results obtained are very different for both types of EBCOAs, and these have to be understood in the differences that the learning phases of these two algorithms present. If we consider the case of EBCOAs for which there is no learning step to be done (*i.e.*, EBCOA_{NB} has a fixed structure) or this step is not very complex, the behavior is similar to that of continuous UMDA (UMDA_c) if the proportion $\bar{F}_H^l / \bar{F}_H^l + \bar{F}_H^l$ is very close to 1. This proportion is maximized providing a quicker convergence when choosing CS_{1+2} and the first of the population-combination types presented. However, this convergence speed is very appropriated in problems such as Sphere model and Ackley in which the local optima are non existent or small, but when local optima are deeper such as in the Griewangk problem, the algorithm performs better when the class-selection and population-combination choices are CS_{1+2} and the elitist combination. The reason for the latter is probably that the problem complexity recommends concentrating on the best individuals with the support of the medium-fitted individuals in order to provide a means to escape from these local optima by searching on regions around the fittest one.

For the case of other EBCOAs in which the learning phase allows considering more dependencies or more complex procedures, other selection and population-combination options apply. This is the case of the $\text{EBCOA}_{\text{TAN}}$, for which the learning phase needs to construct a new structure each generation. Therefore, different options of class-selection and population-combination would lead to very different convergence rates. The nature and complexity of the optimization problem also determines the best type of parameters that EBCOAs should apply as table 1 shows, although for the type of problems chosen for our experiments the best choice was a class-selection of CS_{1+2} with an elitist combination for EBCOA_{NB}. On the other hand, in $\text{EBCOA}_{\text{TAN}}$ the best option is to take a population-combination of Best/Worst classes type and a class-selection type of CS_{1+3}.

4.2 *The influence of the population size*

As many other paradigms in evolutionary computation, EBCOAs also have a set of parameters that directly affect their performance. The previous section provides some ideas on some of these parameters, although a deeper study is necessary.

Since the performance of EBCOAs is very dependent on the way that different individuals of each class represent distinct regions of the search space, the algorithm will converge more appropriately when the size of the population is chosen according to the complexity of the problem. The population size R has to be set in such a way that the division in classes represents also individuals with similar fitness regions, since this parameter compromises the convergence rate at least as much as the class-selection and population-combination ones. Furthermore, this parameter is very problem-dependent and its choice has to be adapted to each problem's particularities to allow an appropriate performance in EBCOAs. The plots in figure 8 illustrate this aspect for the case of the Ackley, Griewangk, and Sphere optimization problems. This shows that, to get a comparable behavior, the size of the population for $\text{EBCOA}_{\text{TAN}}$ needs to be much smaller than for EBCOA_{NB}. The different nature of these three optimization problems show the behavior of EBCOAs when there is no local optimum (Sphere model) or when the local optima are less deep or deeper (Ackley and Griewangk, respectively). This figure suggests that the size of the population influences the overall performance of EBCOAs, and our experiments showed that its effect is even bigger than the choice of the class-selection or the population-combination parameters.

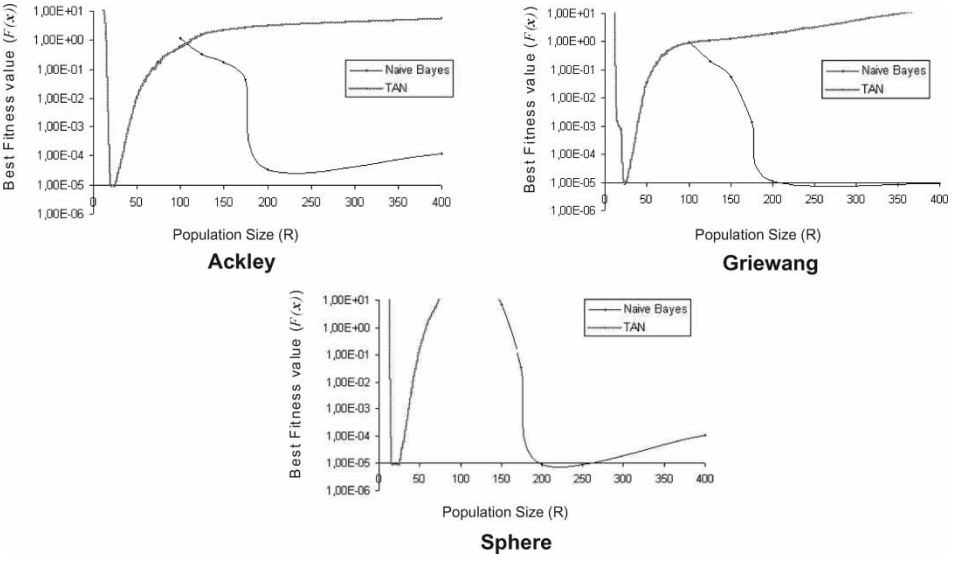

Figure 8. Three examples of how the population size affects the overall performance of EBCOAs. In the three problems (Ackley, Griewangk, and Sphere with $n = 100$) the range of population sizes in $\text{EBCOA}_{\text{TAN}}$ for a full convergence is smaller than that for EBCOA_{NB}. Experiments with EBCOA_{NB} and $\text{EBCOA}_{\text{TAN}}$ were performed with the best class-selection and population-combination values obtained for each of the problems as shown in table 1. The x-axis represents the population size (R) while the y-axis is the best fitness obtained (0 being the global optimum in all the optimization problems).

4.3 *Study of the tuning parameters of EBCOAs and their most adequate values*

In order to have an illustrative overview of the relative dependencies and importance when setting the parameters in EBCOAs, we proceeded to a general study in which the main parameters were tuned in different ways. Analogous type of studies of dependencies between the different configurations of parameters for GAs to provide the best tuning has been an important research trend for a long time (Barr *et al.* 1995, Ballester *et al.* 2005).

In this section, we present such a study for EBCOAs in the format of a Bayesian network that represents the interdependencies between EBCOA parameters for the three optimization problems defined previously. As mentioned before, each different combination of parameter values was executed a total of 30 times. The different execution parameters that were combined in the tests are the following:

(1) Optimization problem: Ackley, Griewangk, Sphere model.
(2) Size of individuals of the problem: 50, 100, 150.
(3) EBCOA type: EBCOA_{NB}, $\text{EBCOA}_{\text{TAN}}$.
(4) Population size: 10, 15, 20, 25, 50, 100, 150, 200, 400.
(5) Class-selection type: CS_{1+2}, CS_{1+3}.
(6) Population-combination type: best/worst classes, best/recent worst, elitist combination.

For each run, we recorded the outcomes of the execution in the form of three variables: whether convergence was reached before the maximum of 60,000 evaluations (Cv.), number of evaluations performed (E) and best fitness value obtained ($F(\boldsymbol{x})$).

A Bayesian network was created for each EBCOA type to best illustrate the dependencies and relative importance that the different parameters have in the final performance of EBCOAs,

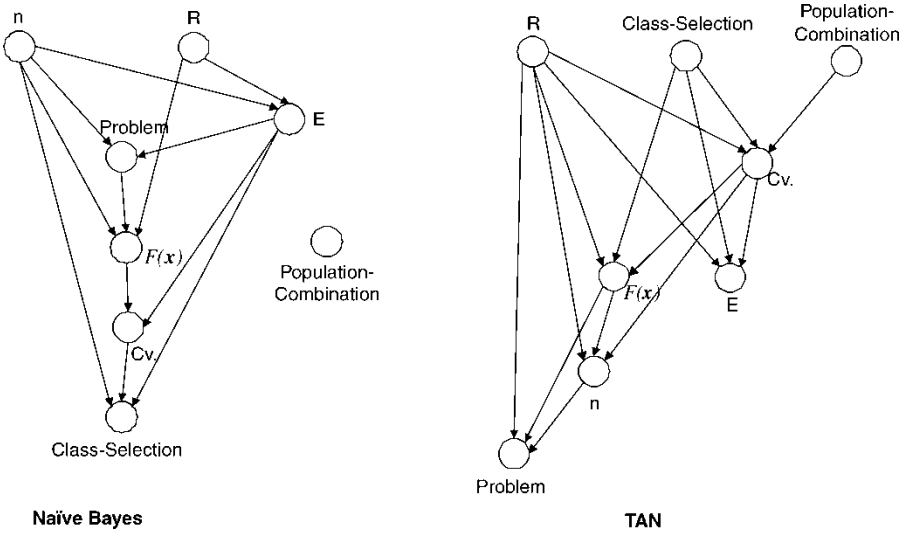

Figure 9. Bayesian networks that illustrate the results of the experiments combining all the different parameters for EBCOAs. The first Bayesian network is for EBCOA_{NB} and the second for $\text{EBCOA}_{\text{TAN}}$. The nodes of the Bayesian networks represent the population size (R), individual size (n), optimization problem (*problem*), class-selection type (*class-selection*), population-combination type (*population-combination*), convergence (Cv.), evaluation number (E), and best fitness value obtained ($F(\boldsymbol{x})$).

adding to it the three outcomes (Cv., E, and $F(\boldsymbol{x})$). Figure 9 shows the two Bayesian networks obtained. As expected, these Bayesian networks show that one of the most important parameters for a good performance on EBCOAs is the population size. This conclusion is shown in figure 8, although the Bayesian networks evidence that this parameter is among the most important ones. In addition, the main differences that exhibit EBCOA_{NB} and $\text{EBCOA}_{\text{TAN}}$ for their learning phase are also confirmed to be very important to bear in mind when choosing the execution parameters, since the class-selection and population-combination parameters appear to not be very significative for EBCOA_{NB} whereas for $\text{EBCOA}_{\text{TAN}}$ these are critical ones. This result was also expected since in the case of $\text{EBCOA}_{\text{TAN}}$ the convergence of the algorithm is improved or worsened depending on the accuracy of the learned Bayesian classifier to represent the characteristics of the more and less promising individuals, and these two parameters are essential to determine the nature of individuals that D_l^C will contain in each of the classes. In naïve Bayes the fact of having a fixed structure makes the algorithm less dependent on these parameters.

As regards the relationships between the three outcome variables (Cv., E, and $F(\boldsymbol{x})$) and their relative dependence to the parameter *optimization problem* shown in figure 9, it must be noted that all these three optimization problems have their optimums in the same fitness value 0, which explains why the arcs between these nodes have different directions and characteristics in both Bayesian networks.

4.4 *Comparative performance of EBCOAs and other evolutionary computation techniques in continuous domains*

An experiment was carried out in order to test the performance of continuous EBCOAs compared to other continuous EDAs and ES. This section describes the experiments and the results obtained.

For the comparison, we have chosen continuous EDAs that take into account a different number of dependencies between variables: UMDA_c, MIMIC_c, and EGNA_{ee}.[†] Other EDAs for continuous domains that do not have restrictions in the type of dependencies such as EGNA_{BGe} were tested in Miquélez *et al.* (2006) for simpler problems, and we have not included them here due to poor comparative performance that they showed regarding the mentioned EDAs. In addition, the overall performance was compared to ES (Schwefel 1995). We chose Evolutionary Strategy with Covariance Matrix Adaptation (CMA-ES) (Hansen *et al.* 2003, Hansen and Kern 2004) which is considered as one with better performance.[‡]

The experiments have been carried out with individuals of 100 variables ($n = 100$) for the three optimization problems described previously. In order to carry out a fair comparison between the different EBCOAs, EDAs and CMA-ES, different population sizes were tested and table 2 uniquely presents the results obtained with population sizes R for which each algorithm obtained the best three results.

In the case of EBCOAs, we set the same parameters as in the previous experiments: $|K| = 3$ and $|C| = 2$, the size of each of the classes was chosen to be $R/3$ (R being the size of D_l^K). In the case of continuous EDAs, the learning of the probabilistic graphical model is done after selecting the $R/2$ best individuals of the population, and applying an elitist approach. Finally, for CMA-ES we set $\mu = \lambda/2$ (maximum number of parents), being λ the population size (using the original authors notation), while the rest of parameters were left with the default values suggested by the original authors.

The stopping criterion for all the algorithms and fitness functions was satisfied when the optimum solution was found (assuming this case when the result obtained was closer than 10^{-6} from the global optimum fitness), or when a maximum of 60,000 evaluations was reached.

Each algorithm and combination of parameters was run 30 times for each of the optimization problems, and the results shown in table 2 are the mean values of these for each problem and algorithm. In order to illustrate more clearly the performance of the different algorithms, these mean values have been computed in this table as follows:

(1) Cv. indicates (in percentage) the convergence rate.
(2) E or number of evaluations, was calculated taking into account only those runs in which the algorithm converged.
(3) $F(x)$ or fitness value, includes only the executions that reached the maximum number of evaluations without converging.

Even if a competitive comparison was not the main purpose of this article, the results of table 2 show that, in general, EBCOAs perform quite well in the different optimization problems, particularly $EBCOA_{\text{TAN}}$.

From the results, we can observe that EDAs' behavior is quite poor, not being able to reach convergence in most of the problems. Only the simplest one, UMDA_c, obtains acceptable results when using a population of 200 individuals. Looking at the characteristics of EDAs, these results are not surprising since they often need to use a large number of generations to converge, and in this case the number of evaluations has been limited to 60,000.

Regarding CMA-ES, this approach is the one that shows the best performance of all the algorithms, reaching a convergence rate close to 100 in almost all the problems. Competitive results are obtained by $EBCOA_{\text{TAN}}$ as it obtained convergence for all the problems and most of the population sizes, even if the number of evaluations required for convergence is slightly larger than in CMA-ES.

[†] See (Larranñaga and Lozano 2001) for a deep review on these algorithms.

[‡] For the simulation step we have applied the MATLAB program cmaes.m version 2.34 which is available on the web at `http://www.icos.ethz.ch/software/evolutionary_computation/cmaes.m`.

Table 2. Mean results after 30 executions with each algorithm and objective function. The Cv., $F(\boldsymbol{x})$ and E columns represent, respectively, the execution converge rate (expressed in percentage), the mean fitness value, and the mean evaluations number.

	150			200			400		
$EBCOA_{NB}$	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E
Ackley	30	6.0E−2	28,050	90	1.6E−3	40,746	0	1.3E−4	–
Griewangk	40	3.0E−2	26,747	90	2.4E−3	30,690	100	–	56,034
Sphere	30	2.3E−1	30,512	90	9.5E−5	38,991	0	1.2E−4	–
	15			20			25		
$EBCOA_{TAN}$	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E
Ackley	20	2.1E+0	21,706	100	–	23,049	100	–	30,375
Griewangk	80	8.8E−3	13,286	90	9.9E−3	30,265	100	–	39,174
Sphere	100	–	11,603	100	–	15,620	100	–	26,567
	200			400			600		
$UMDA_c$	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E
Ackley	20	1.2E−1	49,950	0	3.7E−3	–	0	1.0E−1	–
Griewangk	100	–	29,692	100	–	59,532	0	4.1E−3	–
Sphere	100	–	35,980	0	4.3E−4	–	0	2.4E−1	–
	200			400			600		
$MIMIC_c$	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E
Ackley	0	5.4E−5	–	0	8.3E−4	–	0	3.3E−2	–
Griewangk	0	2.1E−5	–	30	1.1E−5	51,339	0	5.4E−4	–
Sphere	0	2.2E−5	–	0	3.0E−5	–	0	3.7E−2	–
	400			600			800		
$EGNA_{ee}$	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E
Ackley	0	5.2E+00	–	0	5.1E+00	–	0	6.0E+00	–
Griewangk	0	1.6E−01	–	0	6.4E−03	–	0	2.6E−02	–
Sphere	0	1.3E+01	–	0	2.7E−01	–	0	1.7E+00	–
	25			50			100		
CMA-ES	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E	Cv.	$F(\boldsymbol{x})$	E
Ackley	100	–	18,312	100	–	26,587	100	–	43,212
Griewangk	80	7.4E−3	17,268	100	–	25,112	100	–	38,962
Sphere	100	–	11,530	100	–	17,077	100	–	28,182

A last comment has to be made on the computation cost of the different algorithms. Experiments with EDAs and EBCOAs have been performed in a UNIX machine with a 2.7 GHz processor, 16 GB RAM and 512 KB cache. These algorithms were implemented on C++. CMA-ES is implemented in Matlab, and for running it we have used a PC under Windows XP with a 3 GHz CPU, 1 GB RAM and 512 KB cache. The best computation times were obtained with $EBCOA_{NB}$, where each execution lasted approximately 5 s in all the cases. EDA_{UMDA} and EDA_{MIMIC} were next with approximately 15 s, and close to the 20 s required by CMA-ES. In the case of $EBCOA_{TAN}$, execution times per run are longer, around 40 s each. In all the cases EDA_{EGNAee} is the slowest algorithm, since each run could last more than an hour – in our experiments between 27 min and up to 1 h and 58 min is required.

To sum up, the results obtained through this experiment have been useful to demonstrate that EBCOAs are a new promising approach to improve the performance of continuous EDAs,

and that if their parameters are appropriately tuned they can show a better performance than EDAs. On the other hand, since this paper constitutes a step towards applying EBCOAs to optimization in continuous domains, further work is required in order to improve reaching the results obtained with other classical paradigms such as ES.

5. Conclusions and further work

The original contribution of this paper is to generalize EBCOAs to continuous domains by applying Bayesian classifiers adapted for these types of domains. EBCOAs combine evolutionary computation techniques and Bayesian classifiers in order to solve optimization problems. Experimental results to compare the performance of this new approach on typical optimization problems in continuous domains have been provided, and they have been with existing continuous EDAs and ES.

In the light of the results we can conclude that continuous EBCOAs have proved to be a new method able to obtain comparable results to continuous EDAs and ES. On the other hand, EBCOAs appear to be very sensitive to the running parameters, requiring some concrete conditions to ensure a good performance such as an appropriate population size. However, there are still many different aspects that need to be analyzed and tested and that could result in a considerable improvement in the performance of this model.

Future research will include the study and experimentation of other Bayesian classifiers for continuous domains, especially more complex classifiers capable of taking into account more interdependencies between variables that could be useful for more complex problems, such as Bayesian multinets (Kontkanen *et al.* 2000).

An additional research trend is the possibility to substitute in the conditional Gaussian network the discrete class variable C by the actual fitness value provided by the fitness function F without discretization to group all the individuals in the population regarding their performance measured by their respective fitness value. However, this extension requires a careful consideration due to the higher computational cost that it carries out.

Finally, regarding the type of optimization problems for which EBCOAs can be applied, we assume that in other more complex problems than the ones presented here the class-selection CS_{1+3} will be a good choice for avoiding a premature convergence to local optima, although this is again left for future work.

Acknowledgements

This paper has been partially supported by the Spanish Ministry of Science and Technology and the Basque Government with grants TIN2005-03824 and SAIOTEK-S-PE04UN25, respectively. The authors would like to thank the members of the Intelligent Systems Group of the University of the Basque Country for their useful advice and contribution to this work. We would also like to thank Francisco Herrera and the rest of the Soft Computing and Intelligent Information Systems research group of the University of Granada for their support with the use of CMA-ES.

References

D.H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Dordrecht, The Netherlands: Kluwer, 1987.
T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford: Oxford University Press, 1996.

P.J. Ballester, J. Stephenson, J.N. Carter and K. Gallagher, "Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX", in *Proceedings of the IEEE International Congress on Evolutionary Computation*, 2005, pp. 498–505.

R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende and W.R. Stewart, "Designing and reporting on computational experiments with heuristic methods", *J. Heuristics*, 1, pp. 9–32, 1995.

E. Bengoetxea, T. Miquélez, P. Larrañaga and J.A. Lozano, "Experimental results in function optimization with EDAs in continuous domain", in *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, P. Larrañaga and J.A. Lozano, Eds, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001, pp. 181–194.

P.P. Bonissone, Y.-T. Chen, K. Goebel and P.S. Khedkar, "Hybrid soft computing systems: industrial and commercial applications", *Proc. IEEE*, 87, pp. 1641–1667, 1999.

C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees", *IEEE Trans. Inf. Theory*, 14, pp. 462–467, 1968.

T.M. Cover, and J.A. Thomas, *Elements of Information Theory*, New York: John Wiley and Sons, 1991.

R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, New York: John Wiley and Sons, 1973.

N. Friedman, D. Geiger and M. Goldsmidt, "Bayesian network classifiers", *Mach. Learn.*, 29, pp. 131–163, 1997.

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

N. Hansen, "The CMA evolution strategy: a comparing review", in *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*, I. Inza, J.A. Lozano, P. Larrañaga and E. Bengoetxea, Eds, Springer, 2006, pp. 75–102.

N. Hansen, S.D. Müller and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)", *Evol. Comput.*, 11, pp. 1–18, 2003.

N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions", in *Eighth International Conference on Parallel Problem Solving from Nature – PPSN VIII*, 2004, pp. 282–291.

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

I. Kononenko, 1991. "Semi-naïve Bayesian classifiers", in *Proceedings of the 6th European Working Session on Learning*, Porto, Portugal, 1991, pp. 206–219.

P. Kontkanen, P. Myllymäki, H. Tirri and K. Valtonen, "Bayesian multinet classifiers", in *Proceedings of the 10th International Conference on Computing and Information – ICCI 2000*, 2000.

P. Larrañaga and J.A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

S.L. Lauritzen and N. Wermuth, "Graphical models for associations between variables, some of which are qualitative and some quantitative", *Ann. Stat.* 17, pp. 31–57, 1989.

X. Llorà and D.E. Goldberg, "Wise breeding GA via machine learning techniques for function optimization", in *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-03, Part I*, E. Cantú-Paz *et al.*, Lecture Notes in Computer Science 2723 Chicago, Illinois: Springer, 2003, pp. 1172–1183.

J.A. Lozano, P. Larrañaga, I. Inza and E. Bengoetxea, *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*, Berlin: Springer, 2006.

R.S. Michalski, "Learnable evolution model: evolutionary processes guided by machine learning", *Mach. Learn.*, 38, pp. 9–40, 2000.

M. Minsky, "Steps toward artificial intelligence", *Trans. Inst. Radio Eng.*, 49, pp. 8–30, 1961.

T. Miquélez, E. Bengoetxea and P. Larrañaga, "Evolutionary computation based on Bayesian classifiers", *Int. J. Appl. Math. Comput. Sci.* 14, pp. 335–349, 2004.

T. Miquélez, E. Bengoetxea and P. Larrañaga, "Evolutionary Bayesian classifier-based optimization in continuous domains", in *Simulated Evolution and Learning – SEAL´06. Lecture Notes in Computer Science 4247*, T. Wang, X. Li, S. Chen, X. Wang, H. Abbass, H. Iba, G. Chen and X. Yao, Eds, Springer Berlin/Heidelberg, 2006, pp. 529–536.

H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions. I: binary parameters", in *Parallel Problem Solving from Nature – PPSN IV. Lecture Notes in Computer Science 1411*, M. Voigt, *et al.*, Eds, 1996, pp. 178–187.

H. Mühlenbein, T. Mahning and A. Ochoa, "Schemata, distributions and graphical models in evolutionary optimization", *J. Heuristics*, 5, pp. 215–247, 1999.

A. Muñoz, "LEM algorithms", MSc thesis, Computer Engineering Faculty, University of the Basque Country (2003).

A. Pérez, P. Larrañaga and I. Inza, "Supervised classification with conditional Gaussian networks: increasing the structure complexity from naive Bayes", *Int. J. Approx. Reason.*, 43, pp. 1–25, 2006.

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

M. Pelikan, *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, Berlin: Springer, 2005.

M. Pelikan, D.E. Goldberg and E. Cantú-Paz, "BOA: the Bayesian optimization algorithm", in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Orlando FL*, Vol. 1, W. Banzhaf *et al.*, Eds, San Francisco, California: Morgan Kaufmann Publishers, 1999, pp. 525–532.

I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Stuttgart: Frommann–Holzboog, 1973.

H.-P. Schwefel, *Evolution and Optimum Seeking*, Wiley InterScience, 1995.

M. Sebag, and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces", in *Proceedings of the 5th Conference on Parallel Problem Solving from Nature – PPSN V*, T. Bäck, G. Eiben, M. Schoenauer and H.-P. Schwefel, Eds, Berlin: Springer-Verlag, 1998, pp. 418–427.

R. Shachter and C. Kenley, "Gaussian influence diagrams", *Manage. Sci.*, 35, pp. 527–550, 1989.

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

A. Törn and A. Zilinskas, *Global Optimization*, Berlin Heidelberg: Springer–Verlag. Lecture Notes in Computer Science 350, 1989.