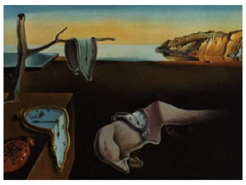

Journal of Statistical Software

July 2010, Volume 35, Issue 7. *http://www.jstatsoft.org/*

Mateda-2.0: Estimation of Distribution Algorithms in MATLAB

Roberto Santana
Universidad Politécnica de Madrid

Concha Bielza
Universidad Politécnica de Madrid

Pedro Larrañaga
Universidad Politécnica de Madrid

Jose A. Lozano
University of the Basque Country

Carlos Echegoyen
University of the Basque Country

Alexander Mendiburu
University of the Basque Country

Rubén Armañanzas
Universidad Politécnica de Madrid

Siddartha Shakya
BT Innovate

Abstract

This paper describes **Mateda-2.0**, a MATLAB package for estimation of distribution algorithms (EDAs). This package can be used to solve single and multi-objective discrete and continuous optimization problems using EDAs based on undirected and directed probabilistic graphical models. The implementation contains several methods commonly employed by EDAs. It is also conceived as an open package to allow users to incorporate different combinations of selection, learning, sampling, and local search procedures. Additionally, it includes methods to extract, process and visualize the structures learned by the probabilistic models. This way, it can unveil previously unknown information about the optimization problem domain. **Mateda-2.0** also incorporates a module for creating and validating function models based on the probabilistic models learned by EDAs.

Keywords: estimation of distribution algorithms, probabilistic models, statistical learning, optimization, MATLAB.

1. Introduction

Statistical techniques have been extensively applied to optimization problems from different domains. Applications of Markov chain Monte Carlo methods (Tierney 1994), Metropolis sampling (Metropolis *et al.* 1953) and simulated annealing (Kirkpatrick *et al.* 1983) are some of the most successful alternatives for many different optimization problems. Recently, a new type of algorithms employing statistical methods and machine learning techniques is steadily receiving more attention from the optimization community.

Estimation of distribution algorithms (EDAs, Larrañaga and Lozano 2002; Mühlenbein and Paaß 1996; Pelikan 2005) are population based optimization algorithms similar to genetic algorithms (GAs, Holland 1975) but which use estimation and sampling from probabilistic models instead of crossover and mutation operators. While the genetic operators used by GAs may have a disruptive effect on problems with complex interactions, EDAs overcome this drawback by capturing and using the interactions to generate new solutions (Mühlenbein *et al.* 1999). This allows EDAs to achieve better results than GAs for this type of problems.

The main idea behind EDAs is to construct a probabilistic model representation of a set of promising solutions already visited by the algorithm and conveniently sample solutions from this representation to guide the search. This leads the algorithm towards the optimal solution. The type of probabilistic models used by EDAs and the methods employed to learn them may vary according to the characteristics of the optimization problem. Therefore, several different EDAs have been proposed for discrete, continuous and mixed problems. EDAs have been applied to a variety of problems in domains such as engineering (Grosset *et al.* 2006; Simionescu *et al.* 2007), biomedical informatics (Armañanzas *et al.* 2008; Larrañaga *et al.* 2006), robotics (Hu *et al.* 2006; Yuan *et al.* 2007), etc.

In addition to their use for optimization, another attribute of EDAs is that the analysis of the probabilistic models learned during the evolution can reveal previously unknown information about the structure of black-box optimization problems. This could provide the user with important knowledge about the characteristics of the problem. The application of statistical techniques can also be useful for extracting this information.

There are a number of implementations of EDAs (de la Ossa *et al.* 2006; Mateo and de la Ossa 2007; Pelikan 2000; Pelikan *et al.* 2006; Zhang and Li 2007; Zhang *et al.* 2008). However, these programs generally implement a particular class of EDAs (e.g., EDAs based on Bayesian networks) and do not allow users to combine their own EDA components with the provided code. Furthermore, these programs do not include methods for processing, extracting and visualizing information from the models learned during the evolution.

Practical EDA implementations should allow an easy evaluation and integration of its components. Namely, different learning and sampling algorithms but also local optimization methods, selection procedures, etc., should be easy to integrate and evaluate. Users should be able to combine available probabilistic learning and sampling methods with their own methods.

In this paper we introduce the **Mateda-2.0**, a MATLAB package (The MathWorks, Inc. 2007). This package is, in many respects, a more complete version of the initial release of the **Mateda** software. **Mateda-2.0** intends to solve general optimization problems using EDAs. It also includes methods for extracting, processing and visualizing characteristic features of the structures in different types of probabilistic graphical models. **Mateda-2.0** was conceived as an open platform, and users can add and use new methods together with the available library of EDA components in **Mateda-2.0**. The idea is for the users to be able to easily evaluate a number of EDA variants before making a decision about the final implementation.

The software takes advantage of the statistical methods provided by the MATLAB statistical toolbox and of a set of learning, sampling and inference techniques included in other two MATLAB libraries: the MATLAB Bayes Net toolbox (**BNT**, Murphy 2001) and **BNT** structure learning package (Leray and Francois 2004). Thanks to the modularity and flexibility of the software, the statistical techniques can be combined with other methods (e.g., clustering algorithms) to effectively use the information gathered during the optimization process.

The paper is organized as follows. Section 2 introduces EDAs, describing their main components and focusing on the algorithms' learning and sampling steps. We also cover EDA research related to the main modules of **Mateda-2.0**. Section 3 presents a general description of the **Mateda-2.0** implementation. It explains the use of the input and output parameters, and the implementation of the probabilistic models. Section 4 outlines the steps for solving an optimization problem using **Mateda-2.0**. Also this section presents detailed examples illustrating how to use some of the methods included in the package. The conclusions of our paper are presented in Section 5.

2. Estimation of distribution algorithms

Algorithm in Table 1 describes the pseudocode of the standard EDA. Each of the main methods the user can implement using **Mateda-2.0** are highlighted in italics.

An EDA begins with the generation of an initial set of solutions (usually called a population). Although the first population is usually randomly generated, it can be generated using a particular heuristic or *seeding method* in some situations, e.g., when previous information about the approximate location of the optimal solutions is available.

Repairing methods should be applied for constrained problems where sampled solutions may be unfeasible and some strategy to repair these solutions is available.

```
Set $t \Leftarrow 0$
Do
    If $t = 0$
      Generate an initial population $D_0$ using a seeding method.
      If required, apply a repairing method to $D_0$.
      Evaluate (all the objectives of) population $D_0$ using an evaluation method.
      If required, apply a local optimization method to $D_0$.
    Else
      Sample a $D_{Sampled}$ population using a sampling method.
      Evaluate (all the objectives of) population $D_{Sampled}$ using an evaluation method.
      If required, apply a repairing method to $D_{Sampled}$.
      If required, apply a local optimization method to $D_{Sampled}$.
      Create $D_t$ from populations $D_{t-1}$ and $D_{Sampled}$ using a replacement method.
      Select a set $D_t^S$ of points from $D_t$ according to a selection method.
      Compute a probabilistic model of $D_t^S$ using a learning method.
$t \Leftarrow t + 1$.
Until the evaluation of the termination criteria method is true.
```

Table 1: Estimation of distribution algorithm.

The *evaluation method* comprises the evaluation of the solutions. For multi-objective problems this may imply the evaluation of several different functions. An advantage of EDAs and other evolutionary algorithms is that the function (usually called fitness function) does not have to be differentiable or even continuous.

EDAs are global optimization algorithms and their results can be improved when used together with *local optimization methods* that do some local search around the current solution.

Sampling methods are used to generate new solutions from the learned probabilistic models. They depend on the type of probabilistic model and the characteristics of the problem. They can be conceived to deal with certain types of constraints in the solutions.

Replacement methods combine the solutions visited in previous generations with the current set of sampled solutions. They help to retain the best solutions found so far and, in some cases, are useful for maintaining the diversity of the solutions.

Selection methods serve to identify the set of solutions that will be modeled, which are usually the solutions with the best fitness evaluation.

The *learning method* is a characteristic and critical EDA component. Depending on the class of models used, this step involves parametric or structural learning, also known as model fitting and model selection, respectively.

Finally, the *termination criteria method* determines the stopping conditions for EDA evolution. These criteria can be as simple as a fixed number of generations or imply a statistical analysis of the current population to evaluate the fulfilment of the stopping condition criteria.

Although some programs included in **Mateda-2.0** can be used for more than one purpose, the programs in the package can be roughly classified into three separate modules. We will organize the explanation of the different functionalities of EDAs provided by **Mateda-2.0** by the modules listed below.

- Optimization module: Implements different EDAs for single and multi-objective problems.
- Data analysis and visualization module: Extracts and visualizes the structures of the models learned during the evolution.
- Function approximation module: Creates and validates models of the functions based on the probabilistic models learned by EDAs.

2.1. Optimization module

The programs included in the optimization module are designed to solve single and multi-objective problems. In single-objective optimization the goal is to obtain an optimal solution to a given problem. The solution of a multi-objective problem is usually defined as an approximation of the problem's Pareto set of solutions, i.e., a set of non-dominated solutions. There are important differences between the components of EDAs for single- and multi-objective problems.

Mateda-2.0 implements a number of optimization functions and some methods used to generate instances of these functions. The implemented functions are from the class of additively decomposable functions.

An additively decomposable function (ADF) is defined as the sum of a set of functions (called subfunctions). Each subfunction is defined on a subset of variables (called definition sets) of the original function. In **Mateda-2.0**, an ADF can be implemented as the sum of its subfunctions for each definition set. Single objective functions can also be implemented as a particular case of multi-objective functions. The package includes a number of methods for defining multi-objective decomposable functions.

Users can also define general multi-objective decomposable functions using global variables. The main difference between this type of multi-objective decomposable functions and ordinary ADFs is that each subfunction will map to a different objective. As a result, we can have different objectives defined on the same subset of variables.

EDAs are usually evaluated using a testbed of instances of a given problem. In **Mateda-2.0**, users can define function generator methods to generate multiple instances of a given decomposable function. The package includes a number of auxiliary methods for this purpose. These methods can also be used to save and read the structure and values of the functions from files.

2.2. Probabilistic modeling in EDAs

One distinctive characteristic of EDAs is probabilistic modeling of the selected set of solutions. In this section we present the main types of probability models implemented in **Mateda-2.0**.

Notation

Let X be a random variable. A value of X is denoted x. $\mathbf{X} = (X_1, \ldots, X_n)$ will denote a vector of random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $N = \{1, \ldots, n\}$, and $\mathbf{X}_S$ (respectively, $\mathbf{x}_S$) a subset of the variables of $\mathbf{X}$ (respectively, a subset of values of $\mathbf{x}$) determined by the indices in S. The joint generalized probability distribution of $\mathbf{x}$ is represented as $\rho(\mathbf{X} = \mathbf{x})$ or $\rho(\mathbf{x})$. $\rho(\mathbf{x}_S)$ will denote the marginal generalized probability distribution for $\mathbf{X}_S$. We use $\rho(X_i = x_i \mid X_j = x_j)$ or, in a simplified form, $\rho(x_i \mid x_j)$, to denote the conditional generalized probability distribution of X_i given $X_j = x_j$.

If $\mathbf{X}$ is discrete, $\rho(\mathbf{X} = \mathbf{x}) = p(\mathbf{X} = \mathbf{x})$ or $p(\mathbf{x})$ denotes the joint probability mass function for $\mathbf{X}$. Similarly, $p(X_i = x_i)$ is the marginal mass probability function of X_i and $p(x_i \mid x_j)$ is the conditional mass probability of X_i given $X_j = x_j$.

If $\mathbf{X}$ is continuous, $\rho(\mathbf{X} = \mathbf{x}) = f_{\mathbf{X}}(\mathbf{x})$ or $f(\mathbf{x})$ is known as the joint density function of $\mathbf{X}$. Similarly, $f_{X_i}(x_i)$ is the marginal density function of X_i, and $f(x_i \mid x_j)$ is the conditional density function of X_i given $X_j = x_j$.

Factorized distributions

Factorizations are important because they provide a condensed representation of probability distributions which are otherwise very difficult to store. Some probability distributions can be expressed as a product of marginal probability distributions, each of which is called a factor. The factorization has two components: The structure and the parameters of the factorization. The structure contains information about which variables belong to each of the factors and the relationships with other factors. The parameters of the factorization are the parameters of each of the factors. Probabilistic graphical models (Lauritzen 1996) serve to represent interactions in complex systems in terms of probabilistic dependencies and to construct factorizations. They usually comprise two components: a graphical structure and a quantitative component.

The two most recurrent criteria used to define factorizations in EDAs have been: (1) Use the problem structure (when available) to construct the factorization; (2) Search the factorization in a constrained space of factorizations consistent with a particular class of probabilistic graphical model.

EDAs like the factorized distribution algorithm (FDA, Mühlenbein *et al.* 1999) use the same factorization structure, usually extracted from a priori knowledge about the problem, in all the generations. Other EDAs, such as those based on Bayesian networks (Etxeberria and Larrañaga 1999; Pelikan *et al.* 1999) and Markov networks (Santana 2005; Shakya and McCall 2007), learn the structure and parameters of the model in each generation.

In **Mateda-2.0**, users can easily specify the structure of a factorization. The package also includes programs that generate structures corresponding to a certain type of models. For instance, it is possible to define a model where each variable forms a factor with its previous k variables in a given order. This model serves to represent Markov-like dependencies between the variables.

Finally, factorizations can be learned by searching in the space of factorizations consistent with a particular class of probabilistic graphical model. In this case, users specify a particular type of learning algorithm that outputs a desired type of probabilistic graphical model. The sampling algorithm should be consistent with the model of choice.

A brief review of the main classes of probabilistic models that can be used with **Mateda-2.0** follows.

Bayesian networks

Bayesian networks (Pearl 1988) are probabilistic graphical models based on directed acyclic graphs. In a Bayesian network, whose directed acyclic graph is represented by S, where the discrete variable X_i has r_i possible values, $\{x_i^1, \ldots, x_i^{r_i}\}$, the local distribution $p(x_i \mid \mathbf{pa}_i^{j,S}, \theta_i)$ is an unrestricted discrete distribution where $\mathbf{pa}_i^{1,S}, \ldots, \mathbf{pa}_i^{q_i,S}$ denote the values of $\mathbf{Pa}_i^S$, the set of parents of X_i in the structure S. q_i is the number of possible different instances of the parent variables of X_i, hence $q_i = \prod_{X_g \in \mathbf{Pa}_i^S} r_g$. The local parameters are given by these conditional probability distributions.

Gaussian networks

In a Gaussian network (Shachter and Kenley 1989), each variable $X_i \in \boldsymbol{X}$ is continuous and each local density function is the linear regression model:

$$f(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \equiv \mathcal{N}(x_i; m_i + \sum_{x_j \in \boldsymbol{pa}_i} b_{ji}(x_j - m_j), \frac{1}{v_i}) \qquad (1)$$

where $\mathcal{N}(x; \mu, \sigma^2)$ is a univariate normal distribution with mean μ and variance σ^2. Given this form, a missing arc from X_j to X_i implies that $b_{ji} = 0$ in the above linear regression model. The local parameters are given by $\boldsymbol{\theta}_i = (m_i, \boldsymbol{b}_i, v_i)$, where $\boldsymbol{b}_i = (b_{1i}, \ldots, b_{i-1i})^t$ is a column vector.

The meaning of the components of the local parameters is as follows: m_i is the unconditional mean of X_i, v_i is the conditional variance of X_i given $\boldsymbol{pa}_i$, and b_{ji} is a linear coefficient reflecting the strength of the relationship between X_j and X_i.

Markov networks

A Markov network or Markov random field (MRF, Kindermann and Snell 1980) is a pair (G, Φ), where G is the structure and Φ is the parameter set of the network. $G = (V, E)$ is an undirected graph where each node corresponds to a random variable and each edge represents conditional dependencies between variables. Unlike Bayesian networks, Markov random fields have undirected edges. Here, the relationship between two nodes should be seen as a neighborhood, rather than a parenthood, relationship.

The neighborhood $N(X_i)$ of a vertex $X_i \in \mathbf{X}$ is defined as $N(X_i) = \{X_j : (X_j, X_i) \in E\}$. The set of edges uniquely determines a neighborhood system on the associated graph G. The boundary of a set of vertices, $\mathbf{X}_S \subseteq \mathbf{X}$, denoted as $bd(\mathbf{X}_S)$, is the set of vertices in $\mathbf{X} \setminus \mathbf{X}_S$ that neighbors at least one vertex in $\mathbf{X}_S$.

A MRF is characterized by its local Markov property known as Markovianity (Besag 2000). Markovianity states that the probability distribution of a node X_i can be completely defined by knowing only its neighboring nodes, i.e., $\forall \mathbf{x} \in \mathbf{X}$, $\forall i \in \{1, \ldots, n\}$, $p(x_i|\mathbf{x} \setminus x_i) = p(x_i|bd(x_i))$.

A probability $p(\mathbf{x})$ on the graph G is called a Gibbs field with respect to the neighborhood system on the associated graph G when it can be represented as

$$p(\mathbf{x}) = \frac{1}{Z} e^{-H(\mathbf{x})}, \tag{2}$$

where $H(\mathbf{x}) = \sum_{C \in \mathcal{C}} \Psi_C(\mathbf{x})$ is called the energy function, $\Psi = \{\Psi_C \in C\}$ being the set of clique potentials, one for each of the maximal cliques in G. The value of $\Phi_C(\mathbf{x})$ depends on the local configuration on the clique C. The normalizing constant Z is the corresponding partition function, $Z = \sum_{\mathbf{x}} e^{-H(\mathbf{x})}$.

Mixtures of distributions

A mixture of distributions (Everitt and Hand 1981) is defined as a distribution of the form:

$$Q(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j \cdot f_j(\mathbf{x}) \tag{3}$$

with $\lambda_j > 0$, $j = 1, \ldots, m$, $\sum_{j=1}^{m} \lambda_j = 1$.

f_j are called component densities or mixture components, and λ_j are called mixture proportions or mixture coefficients. m is the number of components of the mixture. A mixture of probability distributions can be viewed as containing an unobserved choice variable Z, which takes values in $\{1, \ldots, m\}$ with probability λ_j. In some cases the choice variable Z is known. Examples of mixture distributions include mixtures of Gaussian distributions (Everitt and Hand 1981), Bayesian multi-nets (Geiger and Heckerman 1996) and mixtures of trees (Meila and Jordan 2000).

2.3. Learning probabilistic models in EDAs

Probabilistic model learning and sampling are the main characteristics and critical steps of EDAs. Statistical techniques are intensively used in these steps. In this section we briefly describe the characteristics of the learning methods that can be used with **Mateda-2.0**, either because they have been implemented as part of our package or are implemented in one of the packages employed by **Mateda-2.0**. Sampling algorithms are briefly reviewed in the next section.

We focus on structural learning since parametric learning is almost always accomplished through maximum likelihood parameter estimation, generally with Laplace correction. We address the structural learning of some of the probabilistic models mentioned in the previous section: factorized distributions, Bayesian networks, Markov networks and mixtures of distributions.

Mateda-2.0 implements a method for structural learning of a class of factorizations from the data. Marginal product factorizations can be learned by clustering the matrix of mutual information between the variables (Santana *et al.* 2009b). This method is an example of the integration of techniques provided by the package. In a first stage, the mutual information between variables is learned using the solutions from the selected set. In a second stage this matrix is clustered using the affinity propagation algorithm (Frey and Dueck 2007) and the clusters found will match the factors.

There are two main approaches for learning Bayesian networks from data: (1) learning based on detecting conditional independencies by means of independence tests, and (2) score-and-search algorithms.

Dependency relationships among subsets of variables are the inputs of the algorithms in approach (1), whereas the output is the structure of the Bayesian network. The dependency relationships can be obtained from the data by means of independence tests. This type of algorithms differ as to the cost of the statistical test, and the reliability of the results (Larrañaga *et al.* 1999). The PC algorithm (Spirtes *et al.* 1993) is one of the methods of this class that **BNT** and **BNT** structure learning package include.

In approach (2), the problem of finding a good Bayesian network is seen as the optimization of a score function over the set of all possible graph structures. Scores can be viewed as measures of the accuracy of the structure representing the independence relationships in the data. The score usually includes a term for penalizing the network complexity. Likelihood scores are usually employed in conjunction with complexity penalization functions like the Bayesian information criterion (BIC, Schwarz 1978) or the Akaike information criterion (AIC, Akaike 1974). Also Bayesian scores such as the Bayesian-Dirichlet metric (BDe) can be used. BDe metric combines the prior knowledge about the problem and the statistical measure computed from a given data set. **BNT** and **BNT** structure learning package include implementations of the BIC and BDe variants of score-and-search techniques.

Mateda-2.0 includes an algorithm based on a thresholding of the mutual information (Shakya and Santana 2008) to learn the structure of Markov networks in EDAs. This algorithm constructs an approximation of the neighborhood of each variable by computing the variables whose mutual information is above a given threshold. For each variable there will be a factor comprising its closure.

Clustering techniques (Pelikan and Goldberg 2000) and other statistical methods, such as some variants of the expectation-maximization (EM) algorithm (Santana *et al.* 2006), have been used to learn mixtures of distributions in EDAs. Current **Mateda-2.0** methods are based on data clustering using the k-means and affinity propagation algorithms.

2.4. Sampling probabilistic models in EDAs

One common method for generating solutions in EDAs is probabilistic logic sampling (PLS, Henrion 1988). General factorization models, Bayesian and Gaussian networks and the models based on mixtures of probability distributions included in **Mateda-2.0** all use implementations of PLS.

Markov models incorporate the use of Gibbs sampling which is implemented in **Mateda-2.0**. In addition, a temperature parameter is included as part of the sampling algorithms. Thanks to this parameter, linear and Boltzman schedules can be applied to sample the model using simulated annealing (Kirkpatrick *et al.* 1983) as done in (Shakya and Santana 2008).

In some EDAs, it is useful to compute the M most probable configurations (MPCs) or most probable explanations of the graphical model. Nilsson (1998) developed an efficient method for finding the M MPCs. This method is implemented in **Mateda-2.0** for binary Bayesian networks.

2.5. Adaptive modeling and ensemble of probabilistic models

There are two noteworthy issues related to the use of probabilistic modeling in EDAs and the possibilities provided by **Mateda-2.0**:

- Different probabilistic models can be used in different generations: in EDAs, instances of the same class of probabilistic models (e.g., Bayesian network, Markov network, etc.) are usually learned in each generation. However, cases where a different class of probabilistic model could be learned in each generation are conceivable. Switching the class of models according to the characteristics of the data to be modeled is a natural way to introduce adaptation in EDAs (Santana *et al.* 2008a). However, it has not been incorporated into other available EDA packages, probably due to implementation difficulties. **Mateda-2.0** allows users to learn different classes of models in each generation. The only requirement is that the model learned in generation t must be compatible with the sampling algorithm used at the same generation.

- Different types of models can be combined in the same generation to represent different features of the optimization problem: some complex real life problems can be better modeled if particular features are represented independently. In other cases, a problem can be approached using different, possibly redundant, variable representations (e.g., addressing a problem simultaneously using discrete and continuous representations). Sampling algorithms should be able to integrate the information comprised in the different classes of models contained in the ensemble.

2.6. Common EDAs that can be implemented with Mateda-2.0

Mateda-2.0 allows the user to work with the following models:

- General factorizations.

- Bayesian networks.
- Gaussian networks.
- Markov networks.
- Mixtures of distributions.

By combining the EDA components included in **Mateda-2.0**, variants of the following EDAs can be implemented:

- Univariate marginal distribution algorithm (UMDA, Mühlenbein and Paaß 1996).
- Factorized distribution algorithm (FDA, Mühlenbein *et al.* 1999).
- EDAs based on trees and forests (Tree-EDA, Baluja and Davies 1997; Pelikan and Mühlenbein 1999; Santana *et al.* 2001).
- EDAs based on Bayesian and Gaussian networks, similar to those presented in Etxeberria and Larrañaga (1999); Larrañaga (2002); Mühlenbein and Mahnig (2001); Pelikan *et al.* (1999).
- Markov chain estimation of distribution algorithm (Mk-EDA, Santana *et al.* 2008b).
- EDAs based on univariate and multivariate Gaussian distributions (Bosman and Thierens 2000a,b; Larrañaga *et al.* 1999).
- EDAs based on mixtures of continuous distributions (Bosman and Thierens 2002).
- Markov optimization algorithm (MOA, Shakya and Santana 2008).
- Affinity propagation EDA (Aff-EDA, Santana *et al.* 2009b).

The learning and sampling algorithms used by these EDAs, with the exception of those based on Bayesian and Gaussian networks, are functionalities implemented as part of **Mateda-2.0**. To learn and sample Bayesian and Gaussian networks, **Mateda-2.0** uses the functionalities provided by **BNT** (Murphy 2001) and **BNT** structure learning package (Leray and Francois 2004). However, **Mateda-2.0** also implements programs with new Bayesian networks functionalities that are not included in these toolboxes. For example, the computation of the most probable configurations of a Bayesian network has been implemented as part of our package. Additionally, the use of these two packages does not prevent the user for adding other implementations of Bayesian and Gaussian networks. In general, EDAs can be conceived and evaluated within **Mateda-2.0** by including new probabilistic models (e.g., factor graphs, probabilistic neural networks, chain graphs, etc.) with their respective learning and sampling methods.

2.7. Data analysis and visualization module

There are two main different classes of data generated during the EDA search.

- Data related to the generated points and the evaluation of the (possible multiple) objective function (e.g., number and distribution of the generated points, shape of the Pareto front approximation, correlations between objectives, etc.).
- Probabilistic models (e.g., structure and parameters of the probabilistic models).

Consequently, the algorithms implemented in **Mateda-2.0** can be divided into two general groups of methods: (1) computation and visualization of fitness-related measures, and (2) analysis and visualization of dependencies between variables and correlations between the objectives.

Computation and visualization of fitness-related measures

The fitness-related measures that **Mateda-2.0** can compute and visualize are obtained from the fitness values of the solutions visited by the algorithm in each generation.

The average fitness ($\bar{f}$) of the population in each generation can be used as a source of information about the behavior of the EDA. If we are looking for a maximum, an increase in the average fitness means that the algorithm is able to generate better solutions. The fitness variance ($\sigma(f)$) can support additional information about whether the fitness values of the population are really diverse. Similarly, the distribution of the fitness function represented using histograms gives a clearer perspective of the population diversity.

The response to selection ($R(t)$) is a general measure of the improvements in the average fitness of the population achieved by applying variation operators. The amount of selection ($S(t)$) is a measure of the effect of the selection operator in terms of fitness. The realized heritability ($b(t)$) is useful for evaluating the effect of the sampling and replacement methods. The mathematical framework involving the use of $R(t)$, $S(t)$ and $b(t)$ was originally proposed in population genetics and has been applied to the analysis of EDAs (Mühlenbein 1997).

Analysis and visualization of dependencies

In EDAs, the approach followed to analyze the dependencies arising during EDA evolution is to process the probabilistic models produced by the EDAs and extract relevant characteristics from the analysis of these models. This is often done by visually inspecting the models learned.

A number of researchers have studied the most frequent dependencies learned by the probabilistic models in EDAs and analyzed how they map the function structure (Bengoetxea 2003; Lima *et al.* 2007; Mühlenbein and Höns 2005). More recently, some work analyzed how different EDA components influence dependencies (Hauschild *et al.* 2007; Lima *et al.* 2007) and the use of the probabilistic models output by EDAs to speed up the solution of similar problems (Hauschild *et al.* 2008).

Mateda-2.0 includes a number of methods for extracting information from the graphs learned during the evolution. These methods can detect particular substructures learned in the models and identify complex patterns of interactions in the problem. For instance, users might be interested in testing a particular hypothesis related to the *problem structure - model structure* mapping. **Mateda-2.0** can be applied to mine the models, compute the frequency of appearance of the particular substructures in the models, and, from these frequencies, test the original hypothesis. These methods can provide the run and generation in which a user-defined substructure is found in the learned models.

Visualization of structural dependencies

Frequency matrices are the most common information extracted from the analysis of Bayesian networks representing the frequency at which each arc appears in the Bayesian networks. The frequencies of two arcs involving the same pair of variables are counted together. Frequency matrices can be computed for a particular generation or taking the information from all the generations.

The frequency matrix representation has one main limitation: it cannot capture interactions between different substructures of the problem. Not only can **Mateda-2.0** compute frequency matrices, but it also combines the extraction and processing of the structural information contained in the models with different types of visualization techniques, including parallel coordinates, dendrograms and glyphs. The probabilistic structures analyzed can be general factorizations, or Bayesian, Gaussian and Markov networks. Some of the visualization techniques incorporate data clustering.

In the parallel coordinates technique (Inselberg 2009), every observation is plotted for each axis/variable, and a connecting line is drawn for each observation across all the axes. Parallel coordinates are used to visualize the most frequent edges learned at each generation. For this purpose, the vertical axis will represent the generation in which the edges of the model (shown on the horizontal axis) have been learned. A line between two points means that both edges appear in the same structure learned in the same generation.

Dendrograms are graphs used to represent hierarchical trees. A dendrogram consists of many U-shaped lines connecting objects in the hierarchical tree (Johnson 1967). The height of each U represents the distance between the two connected objects. Used to analyze the models, we first compute a hierarchical tree of the selected edges based on a user-defined distance (usually the inverse of the correlation of the edges in the learned structures). Then, the tree is displayed using a dendrogram. This representation can be very effective for detecting hierarchical relationships between substructures in the probabilistic models.

A glyph is a visual representation where the attributes of a graphical entity (e.g., shape, size, color, and position) are dictated by one or more attributes of a data record. The placement or layout of glyphs on a display can communicate significant information regarding the actual data values as well as relationships between data points (Ward 2002). **Mateda-2.0** uses the glyph representation of a subset of edges for a user-defined set of runs and generations. This representation is useful for detecting common substructures in the models and observing the complexity of the models learned through the generations.

2.8. Function approximation module

The goal of the function approximation module is to implement methods for using and validating the probabilistic models learned by EDAs for function approximation. This research trend has received increasing attention in the field of EDAs (Brownlee *et al.* 2008; Sastry *et al.* 2006; Shakya *et al.* 2005).

Probabilistic models of the fitness functions can be useful in different situations:

- To create surrogate functions that help diminish the number of evaluations for costly functions (Sastry *et al.* 2006; Shakya *et al.* 2005).
- To obtain models of black-box optimization problems.

- To unveil and extract problem information that is hidden in the original formulation of the function or optimization problem (Brownlee *et al.* 2008).
- To design improved (local) optimization procedures based on the model structure (Pereira *et al.* 2000; Sastry *et al.* 2006).

An important question is, which of two given probabilistic models of a fitness function is better? Notice that, in this case, models are not intended to be evaluated in terms of how accurately they represent the selected set (i.e., maximize the likelihood of the data). Instead, we would like to use them as general enough *predictors* of the fitness function, or at least of the fitness function of good enough solutions. For multi-objective problems, we can conceive multi-models, each model representing a different objective.

Mateda-2.0 includes methods that allow the user to compare models in terms of criteria such as the correlation between the probabilities assigned by the models to the solutions and their fitness values, the sum of the probabilities assigned to the solutions, and the model entropy.

3. General description of the Mateda-2.0 implementation

In this section, we describe the input and output parameters used by **Mateda-2.0** and the way the probabilistic models are implemented. For a detailed explanation of the implementation characteristics and information about each of the methods implemented, consult the program user's manual (Santana *et al.* 2009a) or access the program's documentation at the **Mateda-2.0** website (Santana and Echegoyen 2009).

The general EDA program `RunEDA.m` is called as:

```
[AllStat,Cache] = RunEDA(PopSize, n, F, Card, cache, edaparams);
```

where the meaning of the input and output parameters are explained in the program user's manual.

Mateda-2.0 represents a factorized distribution using two components:

1. `Cliques`, which represent the variables of each factor, specifying whether they are also included in previous factors or have not appeared before.
2. `Tables`, which contain a probability table for each of the factors.

Each row of `Cliques` is a clique. The first column is the number of overlapping variables with respect to previous cliques in *Cliques*. The second column is the number of new variables. Then, overlapping variables, and finally new variables are listed. `Tables{i}` stores the marginal tables for clique i.

Mateda-2.0 uses **BNT** (Murphy 2001) and **BNT** structure learning package (Leray and Francois 2004) to represent Bayesian and Gaussian models. In **BNT**, a directed acyclic graph (dag) is a data type that serves to represent the structure of a Bayesian network (bnet). The packages also include methods for learning Bayesian and Gaussian networks from data.

In **Mateda-2.0**, `Cliques` are also used to represent the neighborhood structure in models based on Markov networks (Santana 2005; Shakya and McCall 2007). In this particular case, the first column of `Cliques(i,:)` represents the number of neighbors for variable X_i. The second, is the number of new variables (for Markov networks, only one new variable X_i in each clique). Then, neighbor variables are listed and, finally, the variable X_i is added.

The parameters of the Markov network are represented using the *Tables* structure. This stores the conditional probabilities of each variable given its neighbors.

A mixture of models is represented in **Mateda-2.0** as an array of components and coefficients. Each element of the array stores all the relevant information about the corresponding model. The components of a mixture can be models of different classes.

4. How to use Mateda-2.0 for a given problem

In this section, we present the steps to solve an optimization problem using **Mateda-2.0**, illustrated with two examples: a protein model and a feature subset selection problem. Several examples of **Mateda-2.0** applications can be found in the program user's manual (Santana *et al.* 2009a) or accessed from the program documentation at the **Mateda-2.0** website (Santana and Echegoyen 2009).

4.1. Steps to run an EDA in Mateda-2.0

The steps for solving a problem using **Mateda-2.0** are as follows:

1. Create or define the file of the function to be optimized.
2. Define the type of representation to be used.
3. Create the vector with the range of values (for the continuous case) or the cardinality of the variables (for the discrete case).
4. Choose each EDA component, identify its corresponding **Mateda-2.0** implementation and determine the parameters to be passed to each method.
5. Execute `RunEDA.m`.

Mateda-2.0 maximizes the function. For minimization problems, the fitness function $\hat{f}(\mathbf{x})$ has to be modified (e.g., $f(\mathbf{x}) = -\hat{f}(\mathbf{x})$). The choice of EDA components depends on several factors, including:

- Type of variable representation (discrete or continuous).
- Domain of definition for each variable (discrete problems with high cardinality may not be treated using complex models).
- Existence of prior information about the problem (e.g., when a feasible factorization is known it can be employed).
- Computational cost of the fitness function (should be taken into account when setting the population size).

4.2. Optimization and analysis of the hydrophobic-polar protein model

In this section, we illustrate the capabilities of **Mateda-2.0** using an optimization problem defined on a simplified protein model. We have also used **Mateda-2.0** in other optimization problems such as multi-objective satisfiability, the design of artificial networks that resemble the macaque visual cortex network, a spacecraft trajectory optimization problem, etc. (available at `http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/Applications.html`).

HP protein folding problem

The hydrophobic-polar (HP) protein model (Dill 1985) considers two types of residues: hydrophobic (H) residues and hydrophilic or polar (P) residues. A protein is considered to be a sequence of these two types of residues, which are located in regular lattice models forming self-avoided paths. Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice, but not connected in the chain (topological neighbors).

In the linear representation of the sequence, hydrophobic residues are represented by the letter H and polar residues, by P. In the graphical representation, hydrophobic proteins are represented by black beads and polar proteins by white beads.

In the optimization approach, the search for the protein structure is transformed into the search for the optimal configuration given an energy function. For the HP model, an energy function that measures the interaction between topological neighbor residues is defined as $\epsilon_{HH} = -1$ and $\epsilon_{HP} = \epsilon_{PP} = 0$.

The HP problem consists of finding the solution that minimizes the total energy. The problem of finding such a minimum energy configuration is NP-complete for the 2-d lattice (Crescenzi *et al.* 1998). Performance-guaranteed approximation algorithms of bounded complexity have been proposed to solve this problem (Hart and Istrail 1996), but the guaranteed error bound is not small enough for many applications. Greenwood and Shin (2002) surveyed work on evolutionary search applied to protein structure prediction and protein folding for lattice models and real proteins.

Folder `your_installation_path/Mateda/functions/protein` contains an implementation of the HP protein model that can be used with different EDA implementations.

FDA optimization of the HP protein model

In our problem representation, X_i will represent the relative move of residue i in relation to the previous two residues for a given sequence and lattice. Taking the location of the previous two residues in the lattice as a reference, X_i takes values in $\{0, 1, \ldots, z-2\}$, where $z-1$ is the number of permitted movements in the given lattice. These values respectively mean that the new residue will be located in one of the $z-1$ numbers of possible directions with respect to the previous two locations. A solution $\mathbf{x}$ can be seen as a walk in the lattice, representing one possible protein folding. The codification used is called relative encoding. It has been experimentally compared with absolute encoding in (Krasnogor *et al.* 1999) and returned better results.

In Santana *et al.* (2008b), it was shown that EDAs can achieve state-of-the-art results in the optimization of the HP protein model. Here we show how **Mateda-2.0** can be used to

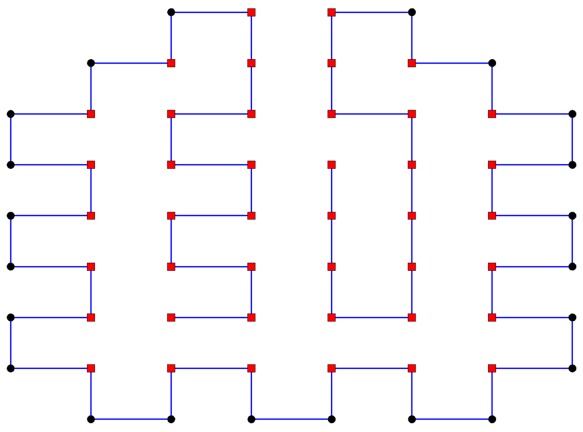

Figure 1: Best solution found by FDA after 144 generations.

implement an EDA approach to this problem. We use an FDA with a chain-like factorization. The following code implements the FDA for a 64-residues HP sequence.

```
global HPInitConf;
HPInitConf = [zeros(1, 12), 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
  1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
  0, 1, 1, 0, 1, 0, 1, zeros(1, 12)];
PopSize = 800;
NumbVar = size(HPInitConf, 2);
cache  = [0, 0, 0, 0, 0];
Card = 3 * ones(1, NumbVar);
maxgen = 300;
Cliques = CreateMarkovModel(NumbVar, 1);
F = 'EvaluateEnergy';
edaparams{1} = {'learning_method', 'LearnFDA', {Cliques}};
edaparams{2} = {'sampling_method', 'SampleFDA', {PopSize}};
edaparams{3} = {'repairing_method', 'HP_repairing', {}};
edaparams{4} = {'stop_cond_method', 'max_gen', {maxgen}};
[AllStat, Cache] = RunEDA(PopSize, NumbVar, F, Card, cache, edaparams)
vector = AllStat{maxgen, 2};
PrintProtein(vector);
```

Other sequence configurations can be tried by modifying the `HPInitConf` variable. The cardinality of variables is 3, and the model considers each variable dependent on the previous one. The fixed structure of the model (`Cliques`) is constructed using the `CreateMarkovModel` method.

The EDA uses a backtracking based repairing method (Cotta 2003) to guarantee that each sequence is folded forming a self-avoided path in the lattice. Notice that, in this case, repairing is used as a way to enforce solution feasibility. At the end of the run, the solution found by the EDA is visualized using the `PrintProtein` method. Figure 1 shows one of the optimal

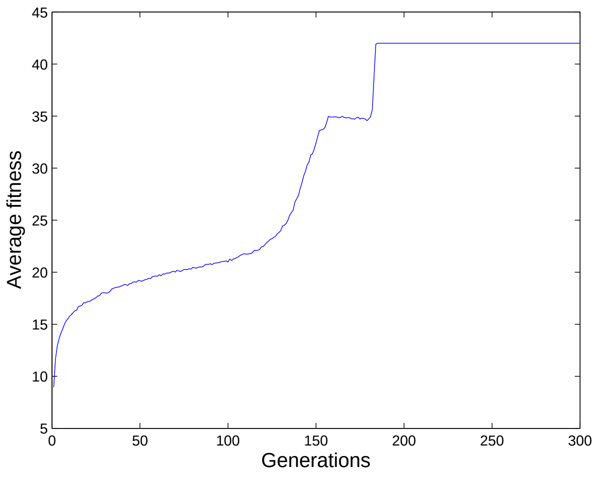

Figure 2: Average fitness of the population in different generations of the FDA.

solutions found by this algorithm in generation 144. This solution is very hard to find for state-of-the-art algorithms (Santana *et al.* 2008b). Figure 2 illustrates the steady increase in the average fitness of the solutions with the generations.

Visualization of the structures

Methods in the **Mateda-2.0**'s data analysis and visualization module can be applied to probabilistic models that have been generated using other EDA implementations. This functionality is achieved by allowing the user to load the model structures in a predefined format Santana *et al.* (2009a). This feature adds to the modular implementation of **Mateda-2.0** guaranteeing a virtual independence between the program's modules.

To illustrate this feature, we present results on the analysis of Bayesian network structures generated by EBNA-Exact (Echegoyen *et al.* 2008), an EDA that employs an exact learning algorithm (Silander and Myllymaki 2006) to learn the Bayesian networks it uses as models. This learning algorithm is very time and memory consuming and feasible only for a small number of variables. Therefore it is not included in **Mateda-2.0**. The C++ implementation of EBNA-Exact was applied to an instance of the HP protein problem. The structures of the Bayesian networks generated by the algorithm[1] were saved and used in the following experiments to illustrate **Mateda-2.0**'s capabilities of extracting and visualizing information about the model structures. Figure 3 shows three Bayesian networks learned in three different generations of an EBNA-Exact run.

We use **Mateda-2.0**'s `ViewPCStruct` method that implements a parallel coordinate visualization. The method allows users to select the most relevant subset of edges for representation. Parameter $const_{edg}$ is the minimal number of times that an edge has to appear in (all) the structures learned to be selected for visualization. Since the clarity of the parallel coordinate visualization depends on the number of variables, this is an important parameter. Another parameter of the algorithm is the minimal number of edges ($min_{edg} > 0$) in the selected

[1]The files containing the structures are located within the **Mateda-2.0** directory.

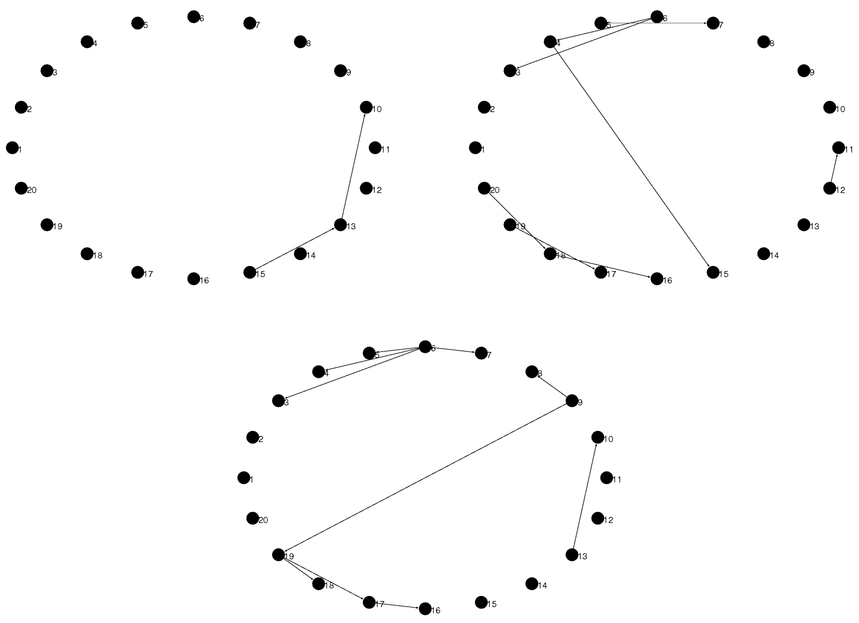

Figure 3: Bayesian networks learned in different generations of an EBNA-Exact run. The top left, top right and bottom graphs, respectively, correspond to the Bayesian networks learned at generations 1, 6 and 11 (last generation).

substructures. Finally, the user can specify the method for ordering the variables before they are displayed. Ordering is useful for grouping features with similar behavior and reducing cluttering in the visualization. `ViewPCStruct` outputs each of the represented edges together with the generation and run in which it was learned.

An example of code in which the `ViewPCStruct` method is applied follows:

```
viewparams{1} = [14];
viewparams{2} = [];
viewparams{3} = 60;
viewparams{4} = 2;
viewparams{5} = 'ClusterUsingDist';
viewparams{6} = 'correlation';
[run_structures, maxgen, nruns] = ReadStructures('ProteinStructsExR.txt', 20);
[results] = ViewStructures(run_structures, 20, maxgen, nruns,
  'viewmatrix_method', 'ViewPCStruct', viewparams);
```

Figure 4 shows the parallel coordinate visualization of the most frequent edges (appearing at least 60 times) in the structures learned in several runs of EBNA-Exact and satisfying the condition of having at least two edges. Notice the different appearance patterns for the edges in Figure 4. For instance, edges numbered as 18 and 20 appear together in all generations. The behavior of the other edges is clearly different.

We now illustrate the use of a method for detecting correlated edges in the structures. This

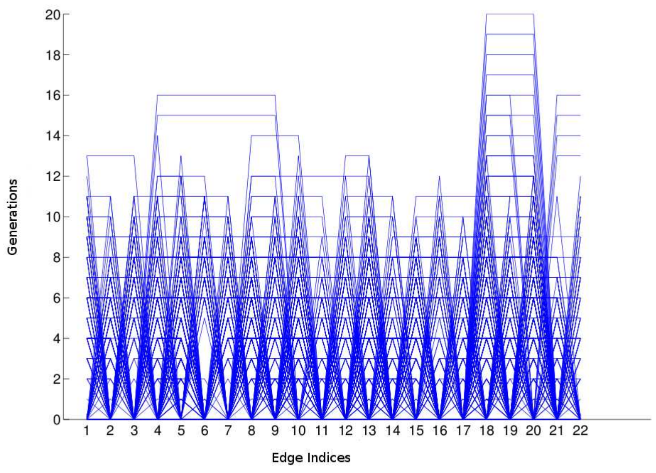

Figure 4: Parallel coordinate visualization of the most frequent edges in the structures learned by EBNA-Exact.

method can be very effective for detecting hierarchical relationships between substructures in the models. The `ViewDenDroStruct` method first computes a hierarchical tree of the selected edges based on a user defined distance (usually the inverse of the correlation of the edges in the learned structures). Then, a dendrogram displays the hierarchical tree. Like `ViewPCStruct`, **Mateda-2.0**'s `ViewDenDroStruct` method can use $const_{edg}$ and $min_{edg} > 0$ parameters to refine the search for structures.

An example of code implementing the `ViewDendroStruct` method follows:

```
viewparams{1} = [14];
viewparams{2} = [];
viewparams{3} = 60;
viewparams{4} = 2;
viewparams{5} = 'correlation';
[run_structures, maxgen, nruns] = ReadStructures('ProteinStructsExR.txt', 20);
[results] = ViewStructures(run_structures, 20, maxgen, nruns,
  'viewmatrix_method', 'ViewDendroStruct', viewparams);
```

Figure 5 shows the dendrogram of the most frequent edges (appearing at least 20 times) in the structures learned in several runs of EBNA-Exact and satisfying the condition of having at least two edges. These are the same edges as shown in Figure 4 using the parallel coordinate representation. The graph shows that some edges tend to be more correlated in the model. Frequently, couples of related edges in the tree comprise contiguously or closely ordered variables (e.g., sets $\{18-17, 19-17, 16-17\}$ or $\{6-5, 7-6\}$). This is consistent with the representation of the HP protein problem where the location of a residue in the lattice is relative to the position of the previous variables. We expect contiguous variables to be more likely to interact.

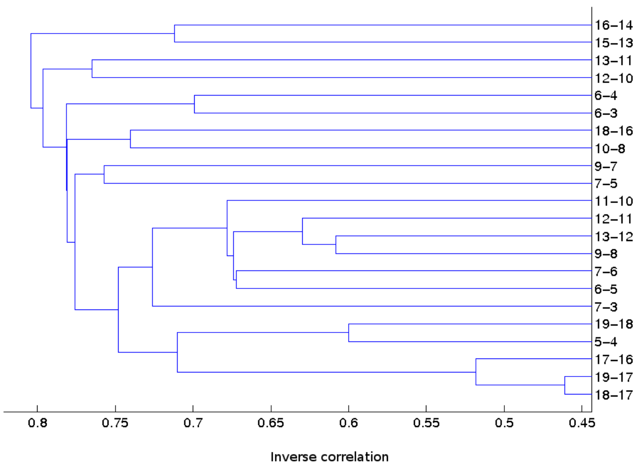

Figure 5: Dendrogram visualization of the most frequent edges in the structures learned by EBNA-Exact.

4.3. Multi-objective feature subset selection and classification problem

In this section, we show how EDAs implemented in **Mateda-2.0** can be combined with classification algorithms to find a subset of features that maximizes the correct classification rate of a given classifier. EDAs have been successfully applied to feature subset selection problems (Armañanzas 2009; Inza *et al.* 2002; Mendiburu *et al.* 2006).

Feature subset selection problem (FSS)

Assume we have a dataset of instances with a set of attributes and an observed class label. The problem consists of selecting a minimal subset of the attributes that will be used as input of a predefined classifier, giving a correct classification rate as high as possible. There are two objectives to be fullfiled: (1) Maximize the correct classification rate given by the classifier, and (2) Minimize the number of selected features. Therefore, the problem is posed as a multi-objective optimization problem and we intend to obtain a good approximation of the Pareto set of solutions.

We use the lung cancer data set (Hong and Yang 1991) from the UCI machine learning repository (Asuncion and Newman 2007). The dataset describes 3 types of pathological lung cancers. It contains 56 attributes which correspond to the results on various medical tests carried out on a patient. All predictive attributes are nominal, taking on integer values 0-3. There are only 32 instances. Class distribution of lung cancer dataset contains 9 examples from class 1, 13 examples from class 2 and 10 examples from class 3.

Optimization of the FSS problem using evolutionary algorithms

Each solution is represented using a binary vector where $x_i = 1$ means that the corresponding feature is included in the classifier. A wrapper approach over the k-nearest neighbor (k-NN) algorithm ($k = 1$) is used to evaluate the goodness of each solution. The performance of a predictive model is estimated using the classifier's correct classification rate averaged over

10×10 cross-validation runs.

Since the dataset contains missing data, in a preliminary step, we use an imputation method that replaces the missing entries with the corresponding value from the nearest-neighbor column using the Euclidean distance (Troyanskaya *et al.* 2001). The correct classification rate is computed using the imputed dataset.

To compute a lower bound of the classifier accuracy, we calculate the correct classification rate when all the features are included. It is 0.5219. We expect that the optimization algorithms will find solutions that improve this value. Recent results (Polat and Güneş 2008) have shown that it is possible to achieve 100% classification for the lung dataset.

We use EBNA and a GA with one-point crossover and mutation to compute two different Pareto sets. The GA has been included to show that **Mateda-2.0** may be used to implement not only EDAs but also other classes of evolutionary algorithms. The following code implements the two evolutionary algorithm approaches to the FSS problem:

```
filenamedata = 'lungcancer.dat';
global AttributesMatrix
global ProblemClasses
AuxMatrix = load(filenamedata);
NumbVar = size(AuxMatrix, 2) - 1;
AttributesMatrix = knnimpute(AuxMatrix(:, 2:NumbVar+1));
ProblemClasses = AuxMatrix(:, 1);
F = 'FindClassifier';
PopSize = 250;
cache  = [1, 1, 1, 1, 1];
maxgen =  30;
selparams(1:2) = {0.5, 'ParetoRank_ordering'};
Card = 2*ones(1, NumbVar);
edaparams{3} = {'selection_method', 'truncation_selection', selparams};
edaparams{4} = {'replacement_method', 'best_elitism', {'ParetoRank_ordering'}};
edaparams{5} = {'stop_cond_method', 'max_gen', {maxgen}};
for Exp=1:30,
  filename = ['NewLungResults_Alg1_Exp_', num2str(Exp), '.mat'];
  BN_params(1:7) = {'k2', 5, 0.05, 'pearson', 'bayesian', 'no', Card};
  edaparams{1} = {'learning_method', 'LearnBN', BN_params};
  edaparams{2} = {'sampling_method', 'SampleBN', {PopSize, 1}};
  [AllStat, Cache] = RunEDA(PopSize, NumbVar, F, Card, cache, edaparams)
  eval(['save ',  filename,   ' AllStat Cache']);
  filename = ['NewLungResults_Alg2_Exp_', num2str(Exp), '.mat'];
  crossover_params = {fix(PopSize/2) + 1};
  edaparams{1} = {'learning_method', 'LearnOnePointCrossover',
    crossover_params};
  edaparams{2} = {'sampling_method', 'SampleOnePointCrossoverPlusMut',
    {PopSize, 1}};
  [AllStat, Cache] = RunEDA(PopSize, NumbVar, F, Card, cache, edaparams)
  eval(['save ', filename, ' AllStat Cache']);
end,
```

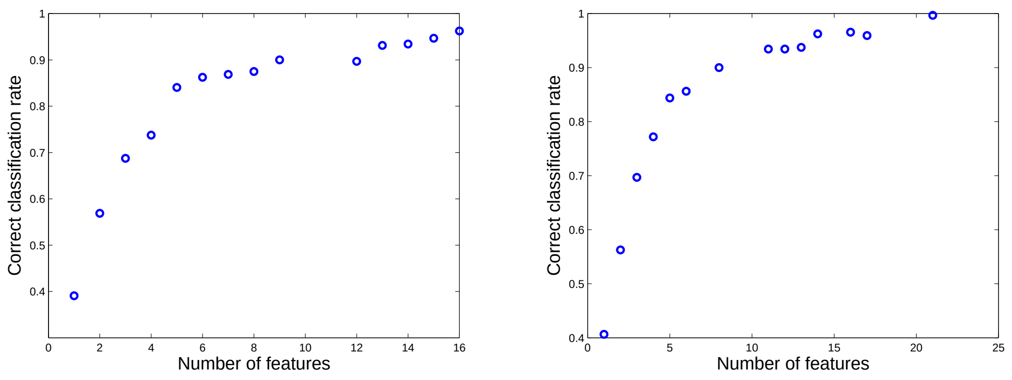

Figure 6: Pareto set approximations of the FSS problem computed with the solutions found by: EBNA (left), GA (right).

All the populations generated by the algorithms are saved. In a subsequent step the selected sets learned in all the generations of all the runs are used to compute the Pareto set approximations. The obtained solutions are displayed in Figure 6. It can be seen that few features are sufficient to obtain accuracies over 0.9, clearly outperforming the classifier that uses all the features. The best classifiers achieved by both evolutionary algorithms have a correct classification rate very close to 1.

The results achieved by both optimization algorithms are then compared using the $\mathcal{C}$ metric. This metric serves to evaluate which of two Pareto set approximations is better. It computes the proportion of decision vectors in each set that are weakly dominated by decision vectors in the other. The value $C(A, B) = 1$ means that all decision vectors in B are weakly dominated by A. The opposite, $C(A, B) = 0$, represents the situation when none of the points in B is weakly dominated by A. Function `ComputeC_Metric` implements the computation of this Pareto set approximation measure in **Mateda-2.0**. Let PS_{EBNA} and PS_{GA} respectively be the Pareto sets found by EBNA and the GA. The results of the $\mathcal{C}$ metric show that $C(PS_{\text{EBNA}}, PS_{\text{GA}}) = 0.4286$ and $C(PS_{\text{GA}}, PS_{\text{EBNA}}) = 0.6429$.

5. Conclusions

In this paper we have presented **Mateda-2.0**, a MATLAB package for optimization using EDAs. Its main characteristics are summarized as follows:

- It can be used to optimize single and multi-objective problems.
- It has a highly modular implementation where each EDA component (either added by the user or already included in **Mateda-2.0**) is implemented as an independent program.
- Available implementations include learning and sampling methods of undirected and directed probabilistic graphical models for problems with discrete and continuous variables.
- The knowledge extraction and visualization module can extract, process, and display

information from the probabilistic models learned and the populations generated by the EDA.

- It has a variety of seeding, local optimization, repairing, selection and replacement methods.
- It includes statistical analysis of different measures of the EDA evolution.
- It has an extended library of functions and testbed problems.

We expect these programs to help find new applications of EDAs to practical problems. The knowledge extraction and visualization methods introduced should be useful for extending the use of probabilistic modeling in optimization, particularly for revealing unknown information in black-box optimization problems. In the future, we also intend to incorporate new methods for dealing with highly complicated, mixed, constrained, and other difficult problems (Santana *et al.* 2009c).

Acknowledgments

This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2008-06815-C02-01, TIN- 2008-06815-C02-02, TIN2007-62626 and Consolider Ingenio 2010 - CSD2007-00018 projects (Spanish Ministry of Science and Innovation), the CajalBlueBrain project, and the COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

Akaike H (1974). "A New Look at the Statistical Identification Model." *IEEE Transactions on Automatic Control*, pp. 716–723.

Armañanzas R (2009). *Consensus Policies to Solve Bioinformatic Problems through Bayesian Network Classifiers and Estimation of Distribution Algorithms.* Ph.D. thesis, Department of Computer Science and Artificial Intelligence, University of the Basque Country. URL `http://www.sc.ehu.es/ccwbayes/members/ruben/RubenPhD.pdf`.

Armañanzas R, Inza I, Santana R, Saeys Y, Flores JL, Lozano JA, Van de Peer Y, Blanco R, Robles V, Bielza C, Larrañaga P (2008). "A Review of Estimation of Distribution Algorithms in Bioinformatics." *BioData Mining*, **1**(6).

Asuncion A, Newman DJ (2007). "UCI Machine Learning Repository." URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Baluja S, Davies S (1997). "Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space." In *Proceedings of the 14th International Conference on Machine Learning*, pp. 30–38. Morgan Kaufmann, San Francisco, CA.

Bengoetxea E (2003). *Inexact Graph Matching Using Estimation of Distribution Algorithms.* Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications. Paris, France. URL `http://www.sc.ehu.es/acwbecae/ikerkuntza/these/`.

Besag J (2000). "Markov Chain Monte Carlo for Statistical Inference." *Working Paper 9*, Center for Statistical and Social Science, University of Washington.

Bosman PA, Thierens D (2000a). "Expanding from Discrete to Continuous Estimation of Distribution Algorithms: The IDEA." In *Parallel Problem Solving from Nature – PPSN VI 6th International Conference*. Springer-Verlag, Paris, France. Lecture Notes in Computer Science 1917.

Bosman PA, Thierens D (2000b). "IDEAs Based on the Normal Kernels Probability Density Function." *Technical Report UU-CS-2000-11*, Utrecht University. URL `http://www.cs.uu.nl/research/techreps/repo/CS-2000/2000-11.pdf`.

Bosman PA, Thierens D (2002). "Multi-Objective Optimization with Diversity Preserving Mixture-Based Iterated Density Estimation Evolutionary algorithms." *International Journal of Approximate Reasoning*, **31**(3), 259–289.

Brownlee S, McCall J, Zhang Q, Brown D (2008). "Approaches to Selection and Their Effect on Fitness Modelling in an Estimation of Distribution Algorithm." In *Proceedings of the 2008 Congress on Evolutionary Computation CEC-2008*, pp. 2621–2628. IEEE Press, Hong Kong.

Cotta C (2003). "Protein Structure Prediction Using Evolutionary Algorithms Hybridized with Backtracking." In J Mira, JR Alvarez (eds.), *Artificial Neural Nets Problem Solving Methods*, volume 2687 of *Lecture Notes in Computer Science*, pp. 321–328. Springer-Verlag.

Crescenzi P, Goldman D, Papadimitriou CH, Piccolboni A, Yannakakis M (1998). "On the Complexity of Protein Folding." *Journal of Computational Biology*, **5**(3), 423–466.

de la Ossa L, Sastry K, Lobo FG (2006). "χ-ary Extended Compact Genetic Algorithm in C++." *IlliGAL Report 2006013*, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL. URL `http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2006013.pdf`.

Dill KA (1985). "Theory for the Folding and Stability of Globular Proteins." *Biochemistry*, **24**(6), 1501–1509.

Echegoyen C, Santana R, Lozano JA, Larrañaga P (2008). "The Impact of Probabilistic Learning Algorithms in EDAs Based on Bayesian Networks." In *Linkage in Evolutionary Computation*, Studies in Computational Intelligence, pp. 109–139. Springer-Verlag.

Etxeberria R, Larrañaga P (1999). "Global Optimization Using Bayesian Networks." In A Ochoa, MR Soto, R Santana (eds.), *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pp. 151–173. Havana, Cuba.

Everitt B, Hand D (1981). *Finite Mixture Distributions*. Chapman and Hall, London.

Frey BJ, Dueck D (2007). "Clustering by Passing Messages between Data Points." *Science*, **315**, 972–976.

Geiger D, Heckerman D (1996). "Knowledge Representation and Inference in Similarity Networks and Bayesian Multinets." *Artificial Intelligence*, **82**(1-2), 45–74.

Greenwood GW, Shin JM (2002). In GB Fogel, DW Corne (eds.), *Evolutionary Computation in Bioinformatics*, chapter On the Evolutionary Search for Solutions to the Protein Folding Problem, pp. 115–136. Morgan Kaufmann, San Francisco, CA.

Grosset L, Riche R, Haftka RT (2006). "A Double-Distribution Statistical Algorithm for Composite Laminate Optimization." *Structural and Multidisciplinary Optimization*, **31**(1), 49–59.

Hart WE, Istrail SC (1996). "Fast Protein Folding in the Hydrophobic-Hydrophilic Model within Three-Eights of Optimal." *Journal of Computational Biology*, **3**(1), 53–96.

Hauschild M, Pelikan M, Lima C, Sastry K (2007). "Analyzing Probabilistic Models in Hierarchical BOA on Traps and Spin Glasses." In D Thierens et al (ed.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2007*, volume I, pp. 523–530. ACM Press, London, UK.

Hauschild M, Pelikan M, Sastry K, Goldberg DE (2008). "Using Previous Models to Bias Structural Learning in the Hierarchical BOA." *MEDAL Report No. 2008003*, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL). URL http://medal.cs.umsl.edu/files/2008003.pdf.

Henrion M (1988). "Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling." In JF Lemmer, LN Kanal (eds.), *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence*, pp. 149–164. Elsevier.

Holland JH (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.

Hong ZQ, Yang JY (1991). "Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane." *Pattern Recognition*, **24**(4), 317–324.

Hu L, Zhou C, Sun Z (2006). "Biped Gait Optimization Using Spline Function Based Probability Model." In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 830–835. Orlando, Florida.

Inselberg A (2009). *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer-Verlag, New York.

Inza I, Larrañaga P, Sierra B (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter Feature Subset Selection by Estimation of Distribution Algorithms, pp. 265–290. Kluwer Academic Publishers, London.

Johnson SC (1967). "Hierarchical Clustering Schemes." *Psychometrika*, **2**, 241–254.

Kindermann R, Snell JL (1980). *Markov Random Fields and Their Applications*. American Mathematical Society, Providence, Rhode Island.

Kirkpatrick S, Gelatt CDJ, Vecchi MP (1983). "Optimization by Simulated Annealing." *Science*, **220**, 671–680.

Krasnogor N, Hart WE, Smith J, Pelta DA (1999). "Protein Structure Prediction with Evolutionary Algorithms." In W Banzhaf, J Daida, AE Eiben, MH Garzon, V Honavar, M Jakiela, RE Smith (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pp. 1596–1601. Morgan Kaufmann, San Francisco, CA.

Larrañaga P (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter A Review on Estimation of Distribution Algorithms, pp. 55–98. Kluwer Academic Publishers, Boston/Dordrecht/London.

Larrañaga P, Calvo B, Santana R, Bielza C, Galdiano J, Inza I, Lozano JA, Armañanzas R, Santafé G, Pérez A, Robles V (2006). "Machine Learning in Bioinformatics." *Briefings in Bioinformatics*, **7**, 86–112.

Larrañaga P, Etxeberria R, Lozano JA, Peña JM (1999). "Optimization by Learning and Simulation of Bayesian and Gaussian Networks." *Technical Report EHU-KZAA-IK-4/99*, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

Larrañaga P, Lozano JA (eds.) (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London.

Lauritzen SL (1996). *Graphical Models*. Clarendon Press, Oxford.

Leray P, Francois O (2004). "**BNT** Structure Learning Package: Documentation and Experiments." *Technical report*, Laboratoire PSI, INSA Rouen, FRE CNRS 2645. URL http://bnt.insa-rouen.fr/programmes/BNT_StructureLearning_v1.3.pdf.

Lima CF, Pelikan M, Goldberg DE, Lobo FG, Sastry K, Hauschild M (2007). "Influence of Selection and Replacement Strategies on Linkage Learning in BOA." In *Proceedings of the 2007 Congress on Evolutionary Computation CEC-2007*, pp. 1083–1090. IEEE Press.

Mateo JL, de la Ossa L (2007). "**LiO** an Easy and Flexible Library of Metaheuristics." *Technical Report DIAB-06-04-1*, Department of Computing Systems, Escuela Politecnica Superior de Castilla La Mancha, Albacete, Spain.

Meila M, Jordan MI (2000). "Learning with Mixtures of Trees." *Journal of Machine Learning Research*, **1**, 1–48.

Mendiburu A, Miguel-Alonso J, Lozano JA, Ostra M, Ubide C (2006). "Parallel EDAs to Create Multivariate Calibration Models for Quantitative Chemical Applications." *Journal of Parallel Distributed Computation*, **66**(8), 1002–1013.

Metropolis N, Rosenbluth A, Teller A, Teller E (1953). "Equations of State Calculations by Fast Computing Machines." *Journal of Chemical Physics*, **21**(6), 1087–1091.

Mühlenbein H (1997). "The Equation for Response to Selection and Its Use for Prediction." *Evolutionary Computation*, **5**(3), 303–346.

Mühlenbein H, Höns R (2005). "The Estimation of Distributions and the Minimum Relative Entropy Principle." *Evolutionary Computation*, **13**(1), 1–27.

Mühlenbein H, Mahnig T (2001). "Evolutionary Synthesis of Bayesian Networks for Optimization." In M Patel, V Honavar, K Balakrishnan (eds.), *Advances in Evolutionary Synthesis of Intelligent Agents*, pp. 429–455. MIT Press, Cambridge, Mass.

Mühlenbein H, Mahnig T, Ochoa A (1999). "Schemata, Distributions and Graphical Models in Evolutionary Optimization." *Journal of Heuristics*, **5**(2), 213–247.

Mühlenbein H, Paaß G (1996). "From Recombination of Genes to the Estimation of Distributions I. Binary Parameters." In HM Voigt, W Ebeling, I Rechenberg, HP Schwefel (eds.), *Parallel Problem Solving from Nature – PPSN IV*, volume 1141 of *Lectures Notes in Computer Science*, pp. 178–187. Springer-Verlag, Berlin.

Murphy K (2001). "The Bayes Net Toolbox for MATLAB." *Computer Science and Statistics: Proceedings of Interface*, **33**, 331–350. URL http://code.google.com/p/bnt/.

Nilsson D (1998). "An Efficient Algorithm for Finding the *M* Most Probable Configurations in Probabilistic Expert Systems." *Statistics and Computing*, **2**, 159–173.

Pearl J (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California.

Pelikan M (2000). "The Bayesian Optimization Algorithm (BOA) with Decision Graphs." *IlliGAL Report No. 2000025*, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL. URL http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2000025.ps.Z.

Pelikan M (2005). *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*, volume 170 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, New York.

Pelikan M, Goldberg DE (2000). "Genetic Algorithms, Clustering, and the Breaking of Symmetry." In M Schoenauer, K Deb, G Rudolph, X Yao, E Lutton, JJ Merelo, HP Schwefel (eds.), *Parallel Problem Solving from Nature – PPSN VI 6th International Conference*, pp. 385–394. Springer-Verlag, Paris, France. Lecture Notes in Computer Science 1917.

Pelikan M, Goldberg DE, Cantú-Paz E (1999). "BOA: The Bayesian Optimization Algorithm." In W Banzhaf, J Daida, AE Eiben, MH Garzon, V Honavar, M Jakiela, RE Smith (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, volume I, pp. 525–532. Morgan Kaufmann Publishers, San Francisco, CA.

Pelikan M, Mühlenbein H (1999). "The Bivariate Marginal Distribution Algorithm." In R Roy, T Furuhashi, PK Chawdhry (eds.), *Advances in Soft Computing – Engineering Design and Manufacturing*, pp. 521–535. Springer-Verlag, London.

Pelikan M, Sastry K, Goldberg DE (2006). "Implementation of the Dependency-Tree Estimation of Distribution Algorithm in C++." *MEDAL Report No. 2006010*, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL). URL http://medal.cs.umsl.edu/files/2006010.pdf.

Pereira FB, Machado P, Costa E, Cardoso A, Ochoa A, Santana R, Soto MR (2000). "Too Busy to Learn." In *Proceedings of the 2000 Congress on Evolutionary Computation CEC-2000*, pp. 720–727. IEEE Press, La Jolla Marriott Hotel La Jolla, California, USA.

Polat K, Güneş S (2008). "Computer Aided Medical Diagnosis System Based on Principal Component Analysis and Artificial Immune Recognition System Classifier Algorithm." *Expert Systems with Applications*, **34**(1), 773–779.

Santana R (2005). "Estimation of Distribution Algorithms with Kikuchi Approximations." *Evolutionary Computation*, **13**(1), 67–97.

Santana R, Echegoyen C (2009). "MATLAB Toolbox for Estimation of Distribution Algorithms (**Mateda-2.0**)." URL `http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/MATEDA.html`.

Santana R, Echegoyen C, Mendiburu A, Bielza C, Lozano JA, Larrañaga P, Armañanzas R, Shakya S (2009a). "**MATEDA**: A Suite of EDA Programs in MATLAB." *Technical Report EHU-KZAA-IK-2/09*, Department of Computer Science and Artificial Intelligence, University of the Basque Country. URL `http://www.sc.ehu.es/ccwbayes/technical.htm`.

Santana R, Larrañaga P, Lozano JA (2006). "Mixtures of Kikuchi Approximations." In J Fürnkranz, T Scheffer, M Spiliopoulou (eds.), *Proceedings of the 17th European Conference on Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Artificial Intelligence*, pp. 365–376. Springer-Verlag.

Santana R, Larrañaga P, Lozano JA (2008a). "Adaptive Estimation of Distribution Algorithms." In C Cotta, M Sevaux, K Sörensen (eds.), *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pp. 177–197. Springer-Verlag.

Santana R, Larrañaga P, Lozano JA (2008b). "Protein Folding in Simplified Models with Estimation of Distribution Algorithms." *IEEE Transactions on Evolutionary Computation*, **12**(4), 418–438.

Santana R, Larrañaga P, Lozano JA (2009b). "Learning Factorizations in Estimation of Distribution Algorithms Using Affinity Propagation." *Evolutionary Computation*. Accepted for publication.

Santana R, Larrañaga P, Lozano JA (2009c). "Research Topics on Discrete Estimation of Distribution Algorithms." *Memetic Computing*, **1**(1), 35–54.

Santana R, Ochoa A, Soto MR (2001). "The mixture of Trees Factorized Distribution Algorithm." In L Spector, E Goodman, A Wu, W Langdon, H Voigt, M Gen, S Sen, M Dorigo, S Pezeshk, M Garzon, E Burke (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pp. 543–550. Morgan Kaufmann Publishers, San Francisco, CA.

Sastry K, Pelikan M, Goldberg DE (2006). "Efficiency Enhancement of Estimation of Distribution Algorithms." In M Pelikan, K Sastry, E Cantú-Paz (eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pp. 161–186. Springer-Verlag.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **7**(2), 461–464.

Shachter R, Kenley C (1989). "Gaussian Influence Diagrams." *Management Science*, **35**, 527–550.

Shakya S, McCall J (2007). "Optimization by Estimation of Distribution with DEUM Framework Based on Markov Random Fields." *International Journal of Automation and Computing*, **4**(3), 262–272.

Shakya S, McCall J, Brown D (2005). "Using a Markov Network Model in a Univariate EDA: An Empirical Cost-Benefit Analysis." In HG Beyer, UM O'Reilly (eds.), *Proceedings of Genetic and Evolutionary Computation Conference GECCO-2005*, pp. 727–734. ACM Press, Washington, D.C.

Shakya S, Santana R (2008). "An EDA Based on Local Markov Property and Gibbs Sampling." In M Keijzer (ed.), *Proceedings of the 2008 Genetic and Evolutionary Computation Conference (GECCO)*, pp. 475–476. ACM, New York.

Silander T, Myllymaki P (2006). "A Simple Approach for Finding the Globally Optimal Bayesian Network Structure." In *Proceedings of the 22th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2006)*, pp. 445–452. Morgan Kaufmann Publishers, San Francisco, CA.

Simionescu PA, Beale D, Dozier GV (2007). "Teeth-Number Synthesis of a Multispeed Planetary Transmission Using an Estimation of Distribution Algorithm." *Journal of Mechanical Design*, **128**(1), 108–115.

Spirtes P, Glymour C, Scheines R (1993). *Causation, Prediction and Search*, volume 81 of *Lecture Notes in Statistics*. Springer-Verlag, New York.

The MathWorks, Inc (2007). *MATLAB – The Language of Technical Computing, Version 7.5*. The MathWorks, Inc., Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Tierney L (1994). "Markov Chains for Exploring Posterior Distributions." *The Annals of Statistics*, **22**, 1701–1762.

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB (2001). "Missing Value Estimation Methods for DNA Microarrays." *Bioinformatics*, **17**(6), 520–525.

Ward MO (2002). "A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization." *Information Visualization*, **1**(3/4), 194–210.

Yuan B, Orlowska ME, Sadiq SW (2007). "Finding the Optimal Path in 3D Spaces Using EDAs – The Wireless Sensor Networks Scenario." In B Beliczynski, A Dzielinski, M Iwanowski, B Ribeiro (eds.), *Adaptive and Natural Computing Algorithms, 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11-14, 2007, Proceedings, Part I*, volume 4431 of *Lecture Notes in Computer Science*, pp. 536–545. Springer-Verlag.

Zhang Q, Li H (2007). "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition." *IEEE Transactions on Evolutionary Computation*, **11**(6), 712–731.

Zhang Q, Zhou A, Jin Y (2008). "RM-MEDA: A Regularity Model Based Multiobjective Estimation of Distribution Algorithm." *IEEE Transactions on Evolutionary Computation*, **12**(1), 41–63.

Affiliation:

Roberto Santana
Universidad Politécnica de Madrid.
Campus de Montegancedo. 28660.
Boadilla del Monte, Madrid, Spain.
E-mail: roberto.santana@upm.es

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 35, Issue 7 *Submitted:* 2009-04-15
July 2010 *Accepted:* 2009-12-17