Intelligent Systems Group

informatika fakultatea facultad de informática

Universidad del País Vasco Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifizialaren Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Contributions to the Analysis and Understanding of Estimation of Distribution Algorithms

by

Carlos Echegoyen

Supervised by Alexander Mendiburu, Roberto Santana and Jose A. Lozano

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country (UPV/EHU) as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, April 2012

To my parents
and Itzi

Acknowledgments

First of all, I owe the greatest thanks to my advisors, Alexander Mendiburu, Roberto Santana and Jose Antonio Lozano. Without their wise guidance, the development of this thesis would not have been possible. I am deeply indebted to them for everything that they have taught me, for the faith that they have put in my work and for the time they have devoted to it. I honestly feel very lucky to have been under the supervision of such good people... Thanks so much.

I also owe a debt of gratitude to Prof. Qingfu Zhang for allowing me to complete a stay as a visitor in his research group at the University of Essex. The scientific discussions that he shared with me were absolutely inspiring and stimulating. His hospitality and the treatment I received were extraordinary. I feel really lucky. It was an unforgettable experience... Thank you very much.

I am also in great debt to all the people in the Intelligent Systems Group with whom I have shared these four years. Their help from those early days of the PhD to the edition of the current document is invaluable. Working with them has been extremely easy and the occasional discussions certainly enriching. They are not only my co-workers, they are my friends.

I am grateful to the University of the Basque Country (UPV/EHU) for the financial support during the last four years. In addition, this work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2010-14931 (Spanish Ministry of Science and Innovation) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

The following lines are devoted to the persons that have supported me not only during my years as PhD candidate but also during my life. I hope you do not mind if I use my mother tongue to complete these lines.

Dicen que la senda que lleva a la casa de un amigo hay que recorrerla a menudo para que no se cubra de maleza. Yo he descubierto que los buenos amigos mantienen esta senda limpia para cuando vayas. Gracias a todos por vuestro apoyo. También desearía hacer una mención especial a las Hijas de la Cruz del colegio Egiluze de Irún. Creo firmemente que el mundo necesita más personas como ellas. Gracias por todo.

A Itziar, la mujer que amo y con la que he compartido estos últimos años, le agradezco de todo corazón su infinita compresión y su apoyo incondicional desde el primero al último de los días que he dedicado a este trabajo. Doy gracias por haber recorrido este camino a su lado y ya no imagino recorrer ningún otro sin ella. Sin duda, ella es parte de esta tesis.

Finalmente, expresar mi más sincero y profundo agradecimiento a mi familia y en especial a mis padres, Begoña y Paco. Mi agradecimiento hacia ellos está contenido en cada una de las palabras que componen este trabajo y en todo el esfuerzo que he invertido en su realización. A ellos les dedico esta tesis, la cual está inevitablemente impregnada con la esencia de su ejemplo y educación.

Contents

List of Figures

List of Tables

List of Algorithms

"Say, on, Sancho," said Don Quixote, "and be brief in thy discourse, for there is no pleasure in one that is long."

1

Preliminaries

1.1 Introduction

Looking for the best solutions to problems is not only a fundamental task for the development of mankind but also seems to be inherent to natural processes, and researchers have been able to see this. Proof of this is the emergence of evolutionary algorithms (EAs) to solve optimization problems regardless of the domain of application. This type of algorithms is mainly inspired by the way in which, according to Darwin (1859), the adaptation of species to the environment is accomplished by nature. Nonetheless, other sources of inspiration from nature, such as the behaviors of ant colonies (Dorigo and Stützle (2004)) or swarms (Kennedy and Eberhart (1995)), have also motivated the development of different EAs. Reciprocally, besides the inspiration of algorithms through the observation of nature, the study of such algorithms could provide us with a better understanding of nature.

EAs commonly search for the best solutions by maintaining a population of individuals (solutions) that evolves from one generation to the next. The evolution consists of selecting a set of individuals from the population and applying, to some subsets of it, recombination operators that create new solutions. A huge number of methods conforming to this framework have been developed. Therefore, the choice of the appropriate alternative for a particular application results in an important matter, as it may determine whether the problem is solved efficiently or, even, if the best solution is found at all.

Mathematically, optimization is the minimization or maximization of a given function. Hence, optimization problems can be formulated as,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{1.1}$$

where $f : S \to \mathbb{R}$ is called the objective function or fitness function, $\boldsymbol{x} = (x_1, \ldots, x_n) \in S$ represents a possible solution of the problem and S is called the search space. The optimum $\boldsymbol{x}^*$ is not necessarily unique. We will assume that S is an n-dimensional discrete search space.

This dissertation is devoted to study a relatively new class of EAs: Estimation of distribution algorithms (EDAs) (Mühlenbein and Paaß (1996)). Based on the same principles of natural selection and evolution of populations, EDAs use explicit probability distributions to lead the search to promising areas of the search space instead of applying genetic operators of crossover and mutation used in genetic algorithms (Goldberg (1989)). Throughout the thesis, we will try to shed light on different open issues regarding EDAs. The final motivation is essentially to achieve a deeper understanding of this type of algorithms and their relationship with the optimization problems. To this end, novel methodological approaches and analyses have been conducted. The basic questions that have guided the elaboration of this work can be summarized as follows.

Firstly, the learning of probabilistic models to extract the relevant information that the selected individuals can contain about the problem is a fundamental step of the algorithm. Regarding this issue, we wonder how the search and the behavior of the EDA is influenced by the accuracy of the learning method.

Secondly, one of the most interesting properties of EDAs is their ability to capture and explicitly represent interactions among the variables of the problems by means of the probabilistic models. Thus, investigating the relationship between the interactions of the problem variables and the structure of the probabilistic model is a question that arises naturally. Following this idea, we also wonder how the topology that these interactions provide determines the difficulty of the problem. More generally, the question of what makes a problem difficult for EDAs is an open question of undoubted interest.

Thirdly, a utopian goal is to know the limits of effectiveness of any search algorithm. Among other things, this type of knowledge would allow us to select the most adequate algorithm depending on the problem at hand. Coming back to more affordable issues, we wonder where the learning limits of EDAs are. We want to better understand when and why the learning step is not able to extract from the population the needed information to reach the optimum.

Fourthly, another fundamental issue regarding EDAs that we consider of special interest is to better understand how the probability of the optimum evolves during the generations. This is an essential characteristic of the algorithm which reflects how the problem is being solved. And finally, a more general issue that we keep in mind is the relationship that emerges between an EDA and the space of optimization problems. Regarding this issue, we consider the possibility of creating taxonomies of problems according to the different behaviors that an EDA can exhibit.

This introductory part will treat, as directly and briefly as possible, the theoretical background related with the dissertation. Thus, only Bayesian networks and EDAs are formally presented. Further details of any topic or scientific discipline related with the aforementioned elements, but not directly used throughout the thesis, can be consulted in different works that will be cited in the appropriate places. In turn, the specific theoretical background

that the different chapters could need, will be introduced in the corresponding points. The rest of the current chapter is organized as follows. Section 1.2 presents both Bayesian networks and the methods used to conduct learning and sampling operations. Section 1.3 introduces EDAs and some specific implementations used throughout the thesis. This section concludes with a review of different EDA implementations that have been developed in the literature. Finally, Section 1.4 presents the contents of the different chapters constituting the current dissertation.

1.2 Bayesian networks

All the algorithms considered throughout the thesis use factorizations that can be encoded by means of Bayesian networks (Pearl (1988)). Bayesian networks, also called belief networks, are a class of probabilistic graphical model (Lauritzen (1996)). This type of models have become a very popular paradigm to efficiently deal with probability distributions in modeling uncertain knowledge. One of the most important sources of the development of Bayesian networks was the field of expert systems (Neapolitan (1990); Castillo et al. (1997); Cowell et al. (1999)). In addition, over the last few years, Bayesian networks have received considerable attention from the machine learning community. As a result of this interest, many publications and tutorials have appeared (Meek (1995); Neapolitan (2003); Jensen and Nielsen (2007); Koller and Friedman (2009)). Thus, besides expert systems, the applications of Bayesian networks include classification problems (Friedman et al. (1997); Ortigosa-Hernandez et al. (2012)), optimization (Etxeberria and Larrañaga (1999); Hauschild et al. (2012)) or bioinformatics (Friedman et al. (2000); Armañanzas (2009)).

As any other probabilistic graphical model, Bayesian networks are the result of combining probability and graph theory. The graphical component of the model encodes a list of conditional independences (Dawid (1979, 1980, 1997)) associated to the probability distribution. Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be an n-dimensional discrete random variable. A Bayesian network is a graphical representation of the factorization of the joint probability distribution for $\boldsymbol{X}$, $p(\boldsymbol{X} = \boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_n)$ denotes an assignment of the variable $\boldsymbol{X}$. More specifically, a Bayesian network can be defined as a pair $(\boldsymbol{s}, \boldsymbol{\theta_s})$ (Larrañaga and Lozano (2002)) where $\boldsymbol{s}$ is a directed acyclic graph (model structure) and $\boldsymbol{\theta_s}$ is the set of parameters associated to the structure (model parameters). The structure $\boldsymbol{s}$ determines the set of conditional (in)dependences among the random variables of $\boldsymbol{X}$. According to the structure $\boldsymbol{s}$, the joint probability distribution $p(\boldsymbol{x})$ can be factorized by means of marginal and conditional probability functions. Specifically, the probability distribution factorizes according to the graph as,

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i | \boldsymbol{pa}_i) \tag{1.2}$$

where $\boldsymbol{pa}_i$ denotes a value of the variables $\boldsymbol{Pa}_i$, the parent set of X_i in the graph $\boldsymbol{s}$.

The local probability distributions of the factorization are those induced by the terms of the product that appears in Equation 1.2. We assume that these local distributions depend on the parameters $\boldsymbol{\theta_s} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. Equation 1.2 can be rewritten specifying the parameters:

$$p(\boldsymbol{x}|\boldsymbol{\theta_s}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{pa}_i, \boldsymbol{\theta}_i). \tag{1.3}$$

Assuming that the variable X_i has r_i possible values, the local distribution $p(x_i|\boldsymbol{pa}_i^j, \boldsymbol{\theta}_i)$ is an unrestricted discrete distribution:

$$p(x_i^k|\boldsymbol{pa}_i^j, \boldsymbol{\theta}_i) \equiv \theta_{ijk} \tag{1.4}$$

where $\boldsymbol{pa}_i^1, \ldots, \boldsymbol{pa}_i^{q_i}$ denote the q_i possible values of the parent set $\boldsymbol{Pa}_i$. The parameter θ_{ijk} represents the probability of variable X_i being in its k-th value, knowing that the set of its parents' variables is in its j-th value. Therefore, the local parameters are given by $\boldsymbol{\theta}_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$.

1.2.1 Bayesian network learning

In order to obtain a Bayesian network which allows us to represent and manage the uncertain knowledge of a specific domain, it is necessary to set both the structure and the parameters. The structure and conditional probabilities necessary for characterizing the Bayesian network can be provided either externally by experts, by automatic learning from datasets or by mixing both of these. We focus on the second approach. Moreover, when the model is automatically learned, it can provide us with insights into the interactions between the variables of the domain.

The learning task can be separated into two subtasks: structural learning and parameter learning. Although there are different strategies to learn the structure of a Bayesian network, we focus on the so-called score+search approach. This type of techniques deals with the structure learning as an optimization problem. Therefore, learning a Bayesian network can be enunciated as follows. Given a data set D with N cases, $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, searching the structure $\boldsymbol{s}^*$ such that,

$$\boldsymbol{s}^* = \arg\max_{\boldsymbol{s} \in \mathcal{S}^n} g(\boldsymbol{s}, D) \tag{1.5}$$

where $g(\boldsymbol{s}, D)$ is the score or metric which measures the goodness of any given structure $\boldsymbol{s}$ with respect to the data set D, and $\mathcal{S}^n$ is the set of all possible directed acyclic graphs with n nodes. Some of the most relevant and used heuristic techniques such as greedy search (Buntine (1991)), simulated annealing (Chickering et al. (1995)), genetic algorithms (Etxeberria et al. (1997)), estimation of distribution algorithms (Blanco et al. (2003); Peña et al.

(2004)) or ant colony optimization (de Campos et al. (2002)) have been applied to this task.

One of the desirable properties of a metric or score is the decomposability in presence of complete data sets. These metrics can be decomposed in sub-metrics associated to each node X_i and its parents $\boldsymbol{Pa}_i$ in the graph $\boldsymbol{s}$. Formally, any decomposable metric can be expressed as:

$$g(\boldsymbol{s}, D) = \sum_{i=1}^{n} g_D(X_i, \boldsymbol{Pa}_i) \tag{1.6}$$

where the function g_D is the sub-metric. Due to the decomposability, the local search methods are computationally more efficient because after adding an arc, we only need to evaluate the family of nodes affected by this change.

Although different learning methods are considered throughout the dissertation, a specific search algorithm will be generally used. It is Algorithm B (Buntine (1991)). This is a greedy search algorithm and the pseudocode is presented in Alg. 1.1, where A is a data structure that stores the information needed to manage the addition of the candidate arcs. Basically, Algorithm B starts with an arcless structure and, at each step, adds the arc which improves the score the most. The algorithm finishes when there is no arc whose addition improves the score.

---

1 Start with an arcless structure
2 Compute $A[X_j \to X_i] = g_D(X_i, X_j) - g_D(X_i)$ for all distinct X_i, X_j
3 **do**
4 Look for the largest $A[X_j \to X_i]$ and add that arc $X_j \to X_i$ to $\boldsymbol{s}$
5 $A[X_j \to X_i] = g_D(X_i, \boldsymbol{Pa}_i \bigcup X_j) - g_D(X_i, \boldsymbol{Pa}_i)$ for all distinct X_i, X_j not belonging to $\boldsymbol{Pa}_i$
6 $A[X_j \to X_i] = -\infty$
7 **until** Every $A[X_j \to X_i] < 0$

Alg. 1.1: Pseudocode of Algorithm B.

Regarding the implementation of the score $g(\boldsymbol{s}, D)$, different alternatives can be considered. Among the most used families of scores we can find marginal likelihood (Cooper and Herskovits (1992); Heckerman et al. (1995)), penalized log-likelihood (Schwarz (1978); Akaike (1974)) or information theory based scores (Herskovits and Cooper (1990); Lam and Bacchus (1994)). In the current dissertation we will use the Bayesian Information Criterion score (BIC) (Schwarz (1978)) based on penalized maximum likelihood. This metric is obtained as follows. Given a dataset $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, we might calculate for any Bayesian network structure $\boldsymbol{s}$ the maximum likelihood estimate $\hat{\boldsymbol{\theta}}_{\boldsymbol{s}}$ for the parameters $\boldsymbol{\theta}_{\boldsymbol{s}}$ and the associated maximized log likelihood:

$$\log p(D \mid \boldsymbol{s}, \boldsymbol{\theta_s}) = \log \prod_{w=1}^{N} p(\boldsymbol{x}_w \mid \boldsymbol{s}, \boldsymbol{\theta_s}) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} p(x_{w,i} \mid \boldsymbol{pa}_i, \boldsymbol{\theta}_i) =$$

$$\sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log(\theta_{ijk})^{N_{ijk}} \tag{1.7}$$

where N_{ijk} denotes the number of cases in D in which the variable X_i has the value x_i^k and $\boldsymbol{Pa}_i$ has its j-th value. Since the maximum likelihood estimate for θ_{ijk} is given by $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$ where $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, we obtain:

$$\log p(D \mid \boldsymbol{s}, \hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}. \tag{1.8}$$

The log-likelihood function is not used to guide the search process due to two main problems. Firstly, the log-likelihood is a monotonous increasing function with respect to the complexity of the model structure. Therefore, the use of this score to evaluate the quality of the structures during the search could lead us towards complete Bayesian networks. Secondly, as the number of parameters for each node increases, the error in the parameter estimation also increases. In order to overcome these difficulties, a penalty term is added to the log-likelihood. A general formula of the penalized log-likelihood is given by:

$$\sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - h(N) dim(S) \tag{1.9}$$

where $dim(S)$ is the dimension (number of parameters needed to specify the model) of the Bayesian network with a structure $\boldsymbol{s}$, i.e. $dim(S) = \sum_{i=1}^{n} q_i(r_i - 1)$. $h(N)$ is a non-negative penalization function. The Jeffreys-Schwarz criterion, which is usually called BIC (Schwarz (1978)), takes into account $h(N) = \frac{1}{2} \log N$. Thus, the BIC score can be written as follows:

$$BIC(\boldsymbol{s}, D) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} p(x_{w,i} | \boldsymbol{pa}_i, \hat{\boldsymbol{\theta}}_i) - \frac{1}{2} \log N \sum_{i=1}^{n} q_i(r_i - 1). \tag{1.10}$$

On the other hand, parameter learning is the numerical assessment of the parameters $\boldsymbol{\theta_s}$ that specify the conditional and marginal probability distributions of the factorization given by $\boldsymbol{s}$. Although this task can be done by means of different approaches such as the Bayesian model averaging (Santafe et al. (2006)) or the maximum a posteriori criterion (Heckerman (1995)), we use the maximum likelihood estimation. Specifically, once the structure has

been learned, the parameters of the Bayesian network are calculated by using the Laplace correction as follows:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + 1}{N_{ij} + r_i} \ . \tag{1.11}$$

1.2.2 Simulation

Once a Bayesian network is obtained, this model is able to provide us with specific probabilistic information of interest. Usually, the information that the practitioner wants to know is the probability of a certain event in the light of particular observations or evidence. The probabilities of interest are not usually stored in the Bayesian network at hand, they need to be computed. This process is known as probabilistic inference (Pearl (1988); Huang and Darwich (1996); Dechter (1999)) and, in the general case, it is an NP-complete problem (Cooper (1990)).

Simulation (also called stochastic sampling) of Bayesian networks can be considered as an alternative to the exact inference. The simulation of any probabilistic graphical model consists of obtaining a sample from the probability distribution for $\boldsymbol{X}$ that the model encodes. Then, the marginal or conditional probabilities of interest can be estimated from the sample.

For our purposes regarding EDAs, the objective of the simulation of Bayesian networks is to obtain a dataset (new population) in which the probabilistic relationships between the random variables of the model are underlying. Particularly, in order to sample the Bayesian network, we consider a forward sampling method. A variable is sampled once all its parents have been sampled. This method is known as probabilistic logic sampling (PLS) (Henrion (1988)). Alg. 1.2 shows a pseudocode of this method.

```
1   π ← Ancestral ordering of the nodes in the Bayesian network
2   for j = 1 to N
3      for i = 1 to n
4         x_{j,π(i)} ← Randomly generate a value from p(x_{π(i)}|pa_{π(i)})
5      done
6   done
```

Alg. 1.2: Pseudocode of the probabilistic logic sampling method.

1.3 Estimation of distribution algorithms

Estimation of distribution algorithms (Mühlenbein and Paaß (1996); Larrañaga and Lozano (2002); Pelikan (2005)) are a population-based optimization paradigm in the field of evolutionary computation (Eiben and Smith (2003)). Initially, a random sample of solutions is generated. These solutions are evaluated using the objective function, and a subset of candidate solutions is selected based on this evaluation. Hence, solutions with better function values have a higher chance of being selected. Then, a probabilistic model from the selected set is built and a new population is sampled from that model. The process is iterated until the optimum has been found or another termination criterion is fulfilled. The general scheme of the EDA approach is shown in Alg. 1.3.

---

1 $D_{t=0} \leftarrow$ Generate N individuals randomly
2 **do**
3 $D_t \leftarrow$ Evaluate individuals
4 $D_t^{Se} \leftarrow$ Select $M \leq N$ individuals from D_t according to a selection method
5 $p_t(\boldsymbol{x}) = p(\boldsymbol{x}|D_t^{Se}) \leftarrow$ Estimate the joint probability distribution by means of a probabilistic model
6 $D_{t+1} \leftarrow$ Sample M individuals from $p_t(\boldsymbol{x})$ and create the new population
7 $t \leftarrow t + 1$
8 **until** Stopping criterion is met

Alg. 1.3: Estimation of distribution algorithm.

EDAs arise, in part, as an alternative to genetic algorithms (GAs) (Goldberg (1989)). Instead of exchanging information between individuals through genetic operators, EDAs use machine learning methods to extract relevant features of the search space through the selected individuals of the population. The replacement of crossover and mutation operators by probabilistic models can bring some benefits. For example, EDAs reduce the number of parameters involved and hence, the tune of the algorithm could become simpler depending on the scenario of application. Nevertheless, the most important benefit could be that the structural component of the probabilistic model can provide explicit information about the interactions among the variables used to codify the problem solutions.

With the aim of finding the optimal solution $\boldsymbol{x}^*$ and solving Problem 1.1, EDAs use explicit probability distributions. At each iteration, the algorithm manages a probability distribution $p(\boldsymbol{X} = \boldsymbol{x})$ of the random variable $\boldsymbol{X}$ taking values from the search space S. Thus, each of the possible problem solutions

has an associated probability of being sampled which varies during the optimization process. The probability values assigned to the solutions are the main source in determining which one will be returned by the algorithm. Consequently, given a problem, the main goal is to get higher probability values for the highest quality solutions throughout an iterative process.

In the last decade, EDAs have acquired special relevance. Proof of this popularity is the development of new and more complex EDAs (Gámez et al. (2007); Bosman (2010); Hauschild et al. (2012)), the applications for these EDAs in different domains such as engineering (Simionescu et al. (2007)), biomedical informatics (Armañanzas et al. (2008); Santana et al. (2008a)) or robotics (Yuan et al. (2007)) and the works which study fundamental issues in order to better understand how these algorithms perform (González et al. (2002); Zhang (2004); Shapiro (2005); Hauschild et al. (2009); Echegoyen et al. (2011b)).

Although there is a wide variety of EDA implementations, as an example, we present below the pseudocode of the univariate marginal distribution algorithm (UMDA), the tree based estimation of distribution algorithm (Tree-EDA) and the estimation of Bayesian networks algorithm (EBNA). These algorithms will be considered in subsequent chapters of the thesis.

1.3.1 Univariate marginal distribution algorithm

The univariate marginal distribution algorithm was introduced in Mühlenbein (1998). This algorithm assumes that all the variables are independent. That is, the value of variable X_i does not depend on the state of any other variable. Then, $p(\boldsymbol{x})$ can be factorized as follows:

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i). \tag{1.12}$$

Alg. 1.4 shows the steps of the UMDA. This algorithm has been successfully applied to different problems such as feature subset selection (Blanco et al. (2001); Saeys et al. (2004)), learning of Bayesian networks from data (Blanco et al. (2003); Romero et al. (2004)), optimization of a composite video processing system (Ali and Topchy (2004)), or to solve some linear and combinatorial problems using Laplace correction (Paul and Iba (2003)).

Theoretical results derived from the UMDA (Mühlenbein and Paaß (1996)) expose its relationship with GAs, particularly with GAs that use uniform crossover. Mühlenbein and Mahnig (2001) have investigated some of the issues that explain the success of UMDA in the optimization of a wide class of functions. Other theoretical results have been obtained for UMDA in González et al. (2002) and Zhang (2004).

1 $D_{t=0} \leftarrow$ Generate N individuals randomly
2 **do**
3 $D_t \leftarrow$ Evaluate individuals
4 $D_t^{Se} \leftarrow$ Select $M \leq N$ individuals from D_t according to a selection method
5 Calculate the univariate marginal frequencies $p_t^s(x_i)$
6 $D_{t+1} \leftarrow$ Sample N individuals from $p_t(\boldsymbol{x}) = \prod_{i=1}^{n} p_t^s(x_i)$
7 $t \leftarrow t + 1$
8 **until** Stopping criterion is met

Alg. 1.4: Pseudocode for UMDA.

1.3.2 Tree-based EDA

Tree-based estimation of distribution algorithms (Baluja and Davies (1997); Santana et al. (1999); Pelikan and Mühlenbein (1999)) use factorizations that can be expressed by means of trees or forests. In particular, we will focus on the implementation presented in Santana et al. (1999). The pseudocode of this algorithm is shown in Alg. 1.5 and will be called Tree-EDA. Although other methods can also be employed, the factorization is constructed using the algorithm introduced in Chow and Liu (1968) that calculates the maximum weight spanning tree from the matrix of mutual information between pairs of variables. Additionally, a threshold for the mutual information values is used when calculating the maximum weight spanning tree in order to allow disconnected components in the structural model.

1 $D_{t=0} \leftarrow$ Generate N individuals randomly
2 **do**
3 $D_t \leftarrow$ Evaluate individuals
4 $D_t^{Se} \leftarrow$ Select $M \leq N$ individuals from D_t according to a selection method
5 Calculate the univariate and bivariate marginal frequencies $p_t^s(x_i)$ and $p_t^s(x_i, x_j)$ from D_t^{Se}
6 Calculate the mutual information and learn the tree structure
7 $D_{t+1} \leftarrow$ Sample N individuals from the tree
8 $t \leftarrow t + 1$
9 **until** Stopping criterion is met

Alg. 1.5: Pseudocode for Tree-EDA.

1.3.3 EDAs based on Bayesian networks

Throughout the dissertation, we pay special attention to EDAs that learn Bayesian networks. There are different implementations of this type of EDAs. The best known algorithms could be the following: learning factorized distribution algorithm (LFDA) (Mühlenbein and Mahnig (1999b)), Bayesian optimization algorithm (BOA) (Pelikan et al. (1999)) or estimation of Bayesian networks algorithm (EBNA) (Etxeberria and Larrañaga (1999)). We mainly focus on the EBNA implementation whose pseudocode is presented in Alg. 1.6.

---

1 $BN_{t=0} \leftarrow (\boldsymbol{s}_0, \boldsymbol{\theta}^0_{\boldsymbol{s}_0})$ where $\boldsymbol{s}_0$ is an arc-less structure and $\boldsymbol{\theta}^0_{\boldsymbol{s}_0}$ is uniform
2 $D_{t=0} \leftarrow$ Generate N individuals from BN_0
3 **do**
4 $D_t \leftarrow$ Evaluate individuals
5 $D_t^{Se} \leftarrow$ Select $M \leq N$ individuals from D_t according to a selection method
6 $\boldsymbol{s}_t \leftarrow$ Obtain a network structure
7 $\boldsymbol{\theta}^t \leftarrow$ Calculate θ^t_{ijk} using D_t^{Se} as the data set
8 $BN_t \leftarrow (\boldsymbol{s}_t, \boldsymbol{\theta}^t_{\boldsymbol{s}_t})$
9 $D_{t+1} \leftarrow$ Sample N individuals from BN_t and create the new population
10 $t \leftarrow t + 1$
11 **until** Stop criterion is met

Alg. 1.6: Pseudocode for EBNA.

In order to better understand how EDAs based on Bayesian networks perform, the characteristics of the learned probabilistic models are a rich source of information which has been studied in several works (Hauschild and Pelikan (2008); Bengoetxea (2003); Brownlee et al. (2008); Echegoyen et al. (2008); Hauschild et al. (2009); Lima et al. (2007); Mühlenbein and Höns (2006)). A straightforward form of analysis is through the explicit dependences between the variables they capture. Thus, it has been shown how different parameters of the algorithm influence the accuracy of the structural models (Lima et al. (2007)), how the dependencies of the probabilistic models change during the search (Hauschild et al. (2009)) and, how the networks learned can provide information about the problem structure (Echegoyen et al. (2007, 2008); Hauschild et al. (2009); Santana et al. (2009a)). Moreover, the structural component of the model can be used to introduce available information of the structural characteristics of the problem (Santana et al. (2008a); Echegoyen et al. (2009); Hauschild et al. (2012)).

1.3.4 Parameters of the EDAs

We have set a configuration of the EDA parameters which is often used throughout the thesis. Therefore, this standard configuration is introduced here to avoid unnecessary repetitions.

According to the main scheme of the EDA, it works with populations of N individuals. The initial population is generated according to a uniform distribution, and hence, all the solutions have the same probability of being sampled. Each iteration starts by selecting a subset of promising individuals from the population. In this step we use truncation selection with a threshold of 50%. Thus, the $N/2$ individuals with the best fitness value are selected. The next step is to learn a probabilistic model from the subset of selected individuals. This is the only step where the algorithms that we will consider differ. Once the model is built, the new population can be generated. In order to do that, N new solutions are sampled from the probabilistic model and then they are added to the N individuals of the current population. The N best individuals, among the $2N$ available, constitute the new population.

As previously commented, every EDA considered in the thesis uses factorizations that can be encoded by means of Bayesian networks. Therefore, the same approaches can be used both to obtain the corresponding parameters and to sample the new solutions. As explained in Section 1.2, the parameters are estimated by maximum likelihood and the new population is generated by PLS (see Alg. 1.2).

1.3.5 Related work

Besides the specific algorithms that we have introduced, a wide variety of EDAs have been presented in the literature. Although there are different ways of classifying EDAs, a brief overview of the different algorithms could be presented as follows. On the one hand, we can distinguish the implementations of EDAs according to the problem domain. Specific algorithms have been developed both in continuous (Larrañaga et al. (2000b); Bengoetxea et al. (2002); Bosman (2003); Bosman and Grahl (2007); Dong and Yao (2008)) and discrete domains (Larrañaga et al. (2000a); Handa (2005)). We can also distinguish particular EDA implementations for permutation based domains (Ceberio et al. (2011)). Similarly, we can find different implementations to face multi-objective problems (Okabe et al. (2004); Zhang et al. (2008)). On the other hand, EDAs can be classified according to the class or complexity of the probabilistic model they rely on. We can find EDAs that use different classes of probabilistic graphical models such as Bayesian networks (Etxeberria and Larrañaga (1999); Mühlenbein and Mahnig (1999b); Pelikan (2005)), Markov networks (Santana (2005); Shakya (2006)), Dependency networks (Gámez et al. (2007)) or Vines (Soto et al. (2012)). The introduction of factor graphs (Kschischang et al. (2001)) in EDAs has also been suggested (Mühlenbein (2012)). In turn, the researchers have developed EDAs that can

be classified according to the complexity of the dependences that they are able to manage. We can distinguish EDAs that allow univariate dependences (Baluja (1994); Mühlenbein and Paaß (1996); Harik et al. (1999)), bivariate dependencies (De Bonet et al. (1997); Cuesta-Infante et al. (2010)), multivariate dependencies (Etxeberria and Larrañaga (1999); Mühlenbein and Mahnig (1999b); Pelikan (2005)), mixture of models (Santana et al. (2001); Thierens and Bosman (2001); Peña et al. (2002)) or even combination with other strategies (Zhang et al. (2003, 2007); Bengoetxea and Larrañaga (2010)).

Regarding the theoretical aspects of EDAs, we find a narrower range of topics and contributions compared with the variety of works discussed in the previous paragraph. We point out the following major contributions without distinguishing among different domains. Firstly, the behavior of EDAs has been modeled by means of Markov Chains (Höhfeld and Rudolph (1997); González et al. (1999); Shapiro (2005)) and dynamical systems (González et al. (2001); Zhang (2004)), obtaining conclusions about the dynamics of the algorithm or about local and global convergence. Secondly, there are works that have been devoted to study the convergence to the global optimum (Mühlenbein et al. (1999); Mühlenbein and Mahnig (1999a); Zhang and Mühlenbein (2004)). And finally, the time complexity of EDAs has also been studied in Chen et al. (2010).

The previous review is not an exhaustive enumeration, it only tries to provide a general perspective. We regret any significant omission.

1.4 Outlook of the dissertation

This thesis is divided into seven chapters. After this introductory part, we start with the first contribution in Chapter 2. This chapter studies the impact that introducing an exact method to learn Bayesian networks has in the behavior of EDAs. By applying a method to learn a Bayesian network that maximizes the BIC score at each step of the algorithm, two important issues in EDAs are investigated. First, we analyze the question of whether learning more accurate (exact) models of the dependencies implies a better performance of EDAs. Secondly, we are able to study the way in which the problem structure is translated into the probabilistic model when exact learning is accomplished. In summary, the results obtained reveal that the quality of the problem information captured by the probability model can improve when the accuracy of the learning algorithm employed is increased. However, improvements in model accuracy do not always imply a more efficient search.

Chapter 3 analyzes the impact that the topology of the problem structure has in the behavior of the algorithm. In many optimization problems, regardless of the domain to which it belongs, the structural component that the interactions among variables provide can be seen as a network. Analyzing the impact that the topological characteristics of these networks can have, both in the hardness of the problem and in the structural models learned

by the EDA, provides valuable information to better understand this type of algorithms. More in particular, the chapter studies the behavior that the EDA exhibits in those problems whose structure is defined by using different network topologies which include grids, small-world networks and random graphs. In order to do that, we use several descriptors of the algorithm such as the population size, the number of evaluations or the structures learned during the search. Furthermore, we take measures from the field of complex networks such as clustering coefficient or characteristic path length in order to quantify the topological properties of the function structure and analyze their relation with the behavior of EDAs. The results show the sensitivity that the algorithm has to the topological characteristics of the problem structure. In addition, this chapter creates a link between EDAs based on Bayesian networks and the emergent field of complex networks.

In Chapter 4 the limits that the learning of Bayesian networks can have in solving optimization problems are investigated. A fundamental issue that plays a key role in understanding and developing algorithms is to be aware of which problems those search algorithms can effectively solve. In order to study the ability limit of EDAs, this chapter experimentally tests three different EDA implementations on a sequence of additively decomposable functions (ADFs) with an increasing number of interactions among binary variables. The results show that the ability of EDAs to solve problems could be lost immediately when the degree of variable interaction is larger than a threshold. We argue that this phase-transition phenomenon is closely related with the computational restrictions imposed in the learning step of this type of algorithms. Moreover, we demonstrate how the use of unrestricted Bayesian networks rapidly becomes inefficient as the number of sub-functions in an ADF increases.

In Chapter 5 we carry out an in-depth analysis of the probability distributions that the EDA generates during the search and the impact that the structure of the Bayesian network has in the behavior of the algorithm. We particularly focus on calculating the probabilities of the optimal solutions, the most probable solution given by the model and the best individual of the population at each step of the algorithm. We carry out the analysis by optimizing functions of a different nature such as Trap5, two variants of Ising spin glass and Max-SAT. The impact of the accuracy of the structural model is carried out by introducing different structures in the probabilistic models. In addition, the objective function values of the analyzed key solutions are contrasted with their probability values in order to study the connection between function and probabilistic model. The results not only show information about the internal behavior of EDAs, but also about the quality of the optimization process and setup of the parameters, the relationship between the probabilistic model and the fitness function, and even about the problem itself. Furthermore, the results allow us to discover common patterns of behavior in EDAs and propose new ideas in the development of this type of algorithms.

In Chapter 6 a theoretical study of the relationship between EDAs and the space of optimization problems is conducted. More particularly, we lay the foundations to elaborate taxonomies of problems under EDAs. By using an infinite population model and assuming that the selection operator is based on the rank of the solutions, we group optimization problems according to the behavior of the EDA. Through the definition of an equivalence relation between functions, it is possible to partition the space of problems in equivalence classes in which the algorithm has the same behavior. We show that only the probabilistic model is able to generate different partitions of the set of possible problems and hence, it predetermines the number of different behaviors that the algorithm can exhibit. As a natural consequence of our definitions, all the objective functions are in the same equivalence class when the algorithm does not impose restrictions to the probabilistic model. The taxonomy of problems is studied in depth for a simple EDA that considers independence among the variables of the problem. We provide the sufficient and necessary condition to decide the equivalence between functions and then the operators to describe and count the members of a class are developed. In addition, it is shown the intrinsic relation between univariate EDAs and the neighborhood system induced by the Hamming distance by proving that all the functions in the same class have the same number of local optima and that they are in the same ranking positions. Finally, numerical simulations are carried out in order to analyze the different behaviors that the algorithm can exhibit for the functions defined over the search space $\{0,1\}^3$.

Finally, Chapter 7 draws the general conclusions obtained during the dissertation and points out possible future works.

2

On the Impact of Learning Optimal Bayesian Networks

2.1 Introduction

In EDAs, linkage learning, understood as the ability to capture the relationships between the variables of the optimization problem, is accomplished by detecting and representing probabilistic dependencies using probability models. The ability of EDAs to learn an accurate representation of the relationships between the variables is related to the class of probabilistic models used and the methods employed to learn them. Thus, an ongoing research trend is to analyze the influence that the dependences considered by the models have in the success of the search, and their relationship with the interdependences among the variables of the problem.

In EDAs based on Bayesian networks, different works have been conducted concerning this type of issues. For instance, some papers (Santana (2002); Mühlenbein and Höns (2005)) report on the way in which the performance of EDAs can change according to the parameters that determine the learning of the probabilistic models, while other works pay attention to the question of how the features of the search space are reflected in the learned models (Bengoetxea (2003); Mühlenbein and Höns (2005); Hauschild et al. (2007); Lima et al. (2007)) or how the dependences of the model relate with interactions among problem variables (Santana et al. (2005)). The aforementioned works were published before developing the contents of this chapter (Echegoyen et al. (2007, 2008)) and it is worth noting that the EDA community maintain the interest in this research line. For instance, Hauschild et al. (2009) analyze the way in which the different components of the EDA influence the arousal of dependencies, Hauschild et al. (2012) use the probabilistic models obtained by EDAs to improve the search in similar problems in the future and Lima et al. (2011) investigate the accuracy of the models learned by the Bayesian optimization algorithm in relation to the problem structure. However, the question of analyzing the relationship between the search space and the structure of the learned probabilistic models becomes difficult due to three main reasons: i) the correct interpretation and understanding of the interdependences among

the problem variables, ii) the stochastic nature of the search and iii) the fact that methods used for learning the models are, in general, able to find only approximate, suboptimal, structures.

This chapter, which is based on the works Echegoyen et al. (2007) and Echegoyen et al. (2008), presents an alternative to study the effect that learning optimal models from the population produces in the behavior of EDAs based on Bayesian networks. Additionally, this contribution serves as a solution to extract more accurate information about the relationship between the problem structure, the search distributions and the probabilistic dependencies learned during the search. More specifically, in this chapter we implement in the EBNA algorithm (see Alg. 1.6) both an exact learning method (Silander and Myllymaki (2006)) and the approximate technique introduced in Alg. 1.1 (Algorithm B). Then, by comparing both, two important issues in EDAs are investigated. First, we analyze the question of whether learning more accurate (exact) models of the dependencies implies a better performance of EDAs. Secondly, we study the way in which the problem structure is translated into the probabilistic model when exact learning is accomplished. In addition, using exact learning allows us to investigate to what extent approximate learning algorithms are responsible for the loss in accuracy in the mapping between the problem structure and the model structure.

The chapter is organized as follows. In Section 2.2, the exact learning method is presented. Section 2.3 introduces the experimental framework and functions used to evaluate the exact and local learning methods used by the EDA. Sections 2.4 and 2.5 respectively present experimental results on the time complexity analysis and convergence reliability of the two EDA variants. Section 2.6 analyzes ways for using the Bayesian networks learned by the EDA as a source of problem knowledge and presents experimental results for several functions. The conclusions are presented in Section 2.7.

2.2 Exact Bayesian network learning

Since finding an unrestricted Bayesian network that maximizes a given score is an NP-hard problem, for a long time the goal of learning exact Bayesian networks was constrained to problems with a very reduced number of variables. The first algorithm that performed this type of learning in less than super-exponential complexity with respect to n was introduced in Koivisto and Sood (2004). For the study conducted in this chapter, we use the algorithm presented in Silander and Myllymaki (2006) to learn Bayesian networks in the EDA framework. This algorithm is feasible for $n < 33$ and it was shown to learn an optimal network for a data set of 29 variables. This was the most efficient algorithm when the contents of this chapter were developed. Currently, more efficient and scalable algorithms exist (Parviainen and Koivisto (2009); Malone et al. (2012)). By using these novel techniques, the same methodology

introduced in this chapter could be extended to other problems with higher dimensionality.

In the following, we try to summarize the basics of the exact learning algorithm used in this chapter (see Silander and Myllymaki (2006) for details). In that work, the Bayesian network structure $\boldsymbol{s}$ is defined as a vector $\boldsymbol{s} = (\boldsymbol{s}_1, .., \boldsymbol{s}_n)$ of parent sets, where $\boldsymbol{s}_i$ is the subset of $\boldsymbol{X}$ from which there are arcs to X_i. The algorithm also considers an ordering of the variables $\boldsymbol{X}$. In this order, the i-th element is denoted by ord_i. The structure $\boldsymbol{s} = (\boldsymbol{s}_1, .., \boldsymbol{s}_n)$ is said to be consistent with an ordering ord when all the parents of the node precede the node in the ordering.

Another important concept in the algorithm is the sink node. Every directed acyclic graph has at least one node with no outgoing arcs, so at least one node is not a parent of any other node. These nodes are called sinks of the network.

In this algorithm, the data set D is processed in a particular way and it uses two kinds of data tables. Given $\boldsymbol{W} \subseteq \boldsymbol{X}$, first it is defined the contingency table $CT(\mathbf{W})$ to be a list of the frequencies of different data-vectors in $D^{\mathbf{W}}$, where $D^{\mathbf{W}}$ is the data set for $\mathbf{W}$ variables. However, the main task is to calculate conditional frequency tables $CFT(X_i, \mathbf{W})$ that record how many times different values of the variable X_i occur together with different vectors $\boldsymbol{x}_j^{\mathbf{W}-\{X_i\}}$ in the data.

As discussed in Chapter 1, many popular scores used to learn Bayesian networks can be decomposed to local scores. According to the definitions used in the algorithm, the score $g(\boldsymbol{s}, D)$ can be expressed as:

$$g(\boldsymbol{s}, D) = \sum_{i=1}^{n} score(CFT(X_i, \boldsymbol{s}_i))$$

where the function *score* is applied to conditional frequency tables. Alg. 2.1 presents the main steps of the method:

1	Calculate the local scores for all $n2^{n-1}$ different (variable, variable set)-pairs
2	Using the local scores, find the best parents for all $n2^{n-1}$ (variable, variable set)-pairs
3	Find the best sink for all 2^n variable sets
4	Using the results from Step 3, find a best ordering of the variables
5	Find a best network using results computed in Steps 2 and 4

Alg. 2.1: Exact Bayesian network learning.

The first step is the main procedure and the only one for which data is needed. It starts by calculating the contingency table for all the variables $\boldsymbol{X}$

and continues calculating contingency tables for all smaller variable subsets, marginalizing variables out of the contingency table. After that, for each contingency table, the conditional frequency table is calculated for each variable appearing in the contingency table. These conditional frequency tables can then be used to calculate the local scores for any parent set given a variable. All the $n2^{n-1}$ local scores are stored in a table which will be the basis of the algorithm.

Having calculated the local scores, the best parents for X_i given a candidate set C are either the whole candidate set C itself or one of the smaller candidate sets $\{C\backslash\{c\}|c \in C\}$. This search must be computed for all 2^{n-1} variable sets (parent candidate sets) related to X_i.

Step 3 of the algorithm is based on the following observation: The best network G^* for a variable set W must have a sink s. As G^* is a network with the highest score, sink s must have incoming arcs from its best possible set of parents. In this way, the rest of the nodes and the arcs must form the best possible network for variables $W\backslash\{s\}$. Therefore, the best sink for $\mathbf{W}$, $sink^*(\mathbf{W})$, is the node that maximizes the sum between the local score for s and the score for the network S without node s.

When we have the best sinks for all 2^n variable sets, it is possible to yield the best ordering ord^* in reverse order. Then, for each position from n to 1, in ord_i^* we have to store the best sink for the set $\bigcup_{j=i+1}^{n}\{ord_j^*(\boldsymbol{X})\}$.

Having a best ordering and a table with the best parents for any candidate set, it is possible to obtain a best network consistent with the given ordering. For the i-th variable in the optimal ordering, the best parents from its predecessors are picked.

The implementation of the algorithm presented in Alg. 2.1 is provided by the authors[1]. The computational complexity of the algorithm is $o(n^2 2^{n-2})$. The memory requirement of the method is 2^{n+2} bytes and the disk-space requirement is $12n2^{n-1}$ bytes. The computational cost of this learning method strongly conditions the experiments conducted in this chapter.

2.3 Experimental framework and function benchmark

To investigate the impact of exact learning in the behavior of Bayesian network based EDAs, we compare two EBNA versions. The two versions of the algorithm only differ in the method used to learn the Bayesian network at each step. The rest of parameters are set as explained in Section 1.3.4. The first EBNA version, called EBNA-Local, implements Algorithm B. The second version, called EBNA-Exact, implements the exact learning method explained in the previous section.

[1] The C++ code of this implementation is available from http://www.cs.helsinki.fi/u/tsilande/sw/bene/download/

We used three different criteria to compare the algorithms. The time complexity, the convergence reliability and the way in which probabilistic dependencies are represented in the structure of the Bayesian network. In this section, we introduce a set of functions that represent different classes of problems and which are used in the following sections to test the behavior of EDAs.

2.3.1 Function benchmark

Let $u(\boldsymbol{x}) = \sum_{i=1}^{n} x_i$, $f(\boldsymbol{x})$ be a unitation function such that, $\forall \boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n, f(\boldsymbol{x}) = f(\boldsymbol{y})$ if $u(\boldsymbol{x}) = u(\boldsymbol{y})$. A unitation function is defined in terms of its unitation value $u(\boldsymbol{x})$, or in a simpler way u.

- Function $OneMax$:

$$OneMax(\boldsymbol{x}) = \sum_{i=1}^{n} x_i = u(\boldsymbol{x}). \tag{2.1}$$

- Unitation functions are also useful for the definition of a class of functions where the difficulty is given by the interactions that arise among subsets of variables. One example of this class of functions are deceptive functions (Goldberg (1989)):

$$Deceptive3(\boldsymbol{x}) = \sum_{i=1}^{\frac{n}{3}} f_{3dec}(x_{3i-2}, x_{3i-1}, x_{3i}) \tag{2.2}$$

  where f_{3dec} can be defined in terms of the function u as:

$$f_{3dec}(u) = \begin{cases} 0.9 \; for \; u = 0 \\ 0.8 \; for \; u = 1 \\ 0.0 \; for \; u = 2 \\ 1.0 \; for \; u = 3 \end{cases}$$

- Function $SixPeaks$ is a modification of the $FourPeaks$ problem (Baluja and Davies (1997)) and it can be defined mathematically as:

$$SixPeaks(\boldsymbol{x}, t) = max\{tail(0, \boldsymbol{x}), head(1, \boldsymbol{x}), tail(1, \boldsymbol{x}), head(0, \boldsymbol{x})\} + \mathcal{R}(\boldsymbol{x}, t) \tag{2.3}$$

  where

  $tail(b, \boldsymbol{x})$ = number of contiguous trailing b's in $\boldsymbol{x}$
  $head(b, \boldsymbol{x})$ = number of contiguous leading b's in $\boldsymbol{x}$

$$\mathcal{R}(\boldsymbol{x}, t) = \begin{cases} n & \text{if } (tail(0, \boldsymbol{x}) > t \text{ and } head(1, \boldsymbol{x}) > t) \text{ or} \\ & \quad (tail(1, \boldsymbol{x}) > t \text{ and } head(0, \boldsymbol{x}) > t) \\ 0 & \text{otherwise} \end{cases}$$

The goal is to maximize the function. For an even number of variables this function has 4 global optima, located at the points:

$$(\overbrace{0,\ldots,0}^{t+1},1,\ldots,1)\ (0,\ldots,0,\overbrace{1,\ldots,1}^{t+1})\ (\overbrace{1,\ldots,1}^{t+1},0,\ldots,0)\ (1,\ldots,1,\overbrace{0,\ldots,0}^{t+1})$$

These points are very difficult to reach because they are isolated. On the other hand, two local optima $(0,0,\ldots,0)$, $(1,1,\ldots,1)$ are very easily reachable. The value of t was set to $\frac{n}{2}-1$.

- The *Parity* function is a simple k-bounded additively separable function that has been used to investigate the limitations of linkage learning by probabilistic modeling. It can be seen as a generalization of the XOR function. In this case we will work with the concatenated parity function (CPF). It is said that this problem is hard for EDAs based on Bayesian networks (Coffin and Smith (2007)). The *Parity* function can be defined as:

$$Parity(\boldsymbol{x}) = \begin{cases} C_{even} & \text{if } u \text{ is even} \\ C_{odd} & \text{otherwise} \end{cases}$$

where C_{even} and C_{odd} are parameters of the function. The CPF is defined as m concatenated parity sub-functions of size $k=5$,

$$CPF(\boldsymbol{x}) = \sum_{i=1}^{m} parity(x_{5i-4}, x_{5i-3}, \ldots, x_{5i}). \tag{2.4}$$

As in Coffin and Smith (2007), we use $C_{odd}=5$ and $C_{even}=0$. Notice that there are 2^{n-m} solutions where the function reaches the global optima.

- Function *Cuban*5 (Mühlenbein et al. (1999)) is a non-separable additive function. The second best value of this function is very close to the global optimum.

$$Cuban5(\boldsymbol{x}) = F^5_{cuban1}(s_0) + \sum_{j=0}^{m}(F^5_{cuban2}(s_{2j+1}) + F^5_{cuban1}(s_{2j+2})) \tag{2.5}$$

where the s_i are substrings of $\boldsymbol{x}$ containing 5 consecutive variables and $n = 4(2m+1)+1$. The sub-functions that constitute this function *Cuban*5 can be defined as follows.

$$F^3_{cuban1}(x,y,z) = \begin{cases} 0.595 \text{ for } 000 \\ 0.200 \text{ for } 001 \\ 0.595 \text{ for } 010 \\ 0.100 \text{ for } 011 \\ 1.000 \text{ for } 100 \\ 0.050 \text{ for } 101 \\ 0.090 \text{ for } 110 \\ 0.150 \text{ for } 111 \end{cases}$$

$$F^5_{cuban1}(x,y,z,v,w) = \begin{cases} 4F^3_{cuban1}(x,y,z) & \text{if } y = v \text{ and } z = w \\ 0 & \text{otherwise} \end{cases}$$

$$F^5_{cuban2}(x,y,z,v,w) = \begin{cases} u & \text{for} \quad w = 0 \\ 0 & \text{for } x = 0, w = 1 \\ u - 2 & \text{for } x = 1, w = 1 \end{cases}$$

where x, y, z, v, w are the variables belonging to any substring s_i.

2.3.1.1 The HP protein model

In our experiments we also use a class of coarse-grained protein folding model called the hydrophobic-polar (HP) model (Dill (1985)).

Under specific conditions, a protein sequence folds into a native 3-dimensional structure. The problem of determining the protein native structure from its sequence is known as the protein structure prediction problem. To solve this problem, a protein model is chosen and an energy is associated to each possible protein fold. The search for the protein structure is transformed into the search for the optimal protein configuration given the energy function.

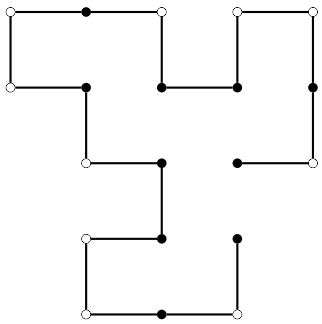


Fig. 2.1. An optimal solution of the HP model for sequence $HPHPPHHPHPPHPHHPPHPH$. The optimal energy corresponding to this sequence is -9.

The HP model considers two types of residues: hydrophobic (H) residues and hydrophilic or polar (P) residues. In this model, a protein is considered as

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). The total number of topological neighboring positions in the lattice (z) is called the lattice coordination number. Fig. 2.1 shows one possible configuration of sequence $HPHPPHHPHPPHPHHPPHPH$ in the HP model.

A solution $\boldsymbol{x}$ can be interpreted as a walk in the lattice, representing one possible folding of the protein. We use a discrete representation of the solutions. For a given sequence and lattice, X_i will represent the relative move of residue i in relation to the previous two residues. Taking as a reference the location of the previous two residues in the lattice, X_i takes values in $\{0, 1, \ldots, z-2\}$, where $z-1$ is the number of movements allowed 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. If the encoded solution is self-intersecting, it can be repaired or penalized during the evaluation step using a recursive repairing procedure introduced in Cotta (2003). Therefore, values for X_1 and X_2 are meaningless. The locations of these two residues are fixed.

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 function that returns the total energy of a solution will be called *Protein* function. The HP problem consists of finding the solution that minimizes the total energy. More details about the representation and function can be found in Santana et al. (2008a).

2.4 Time complexity analysis

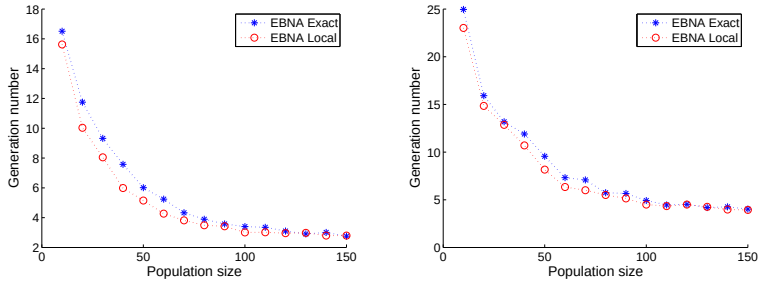
In this section, the time complexity analysis will refer to the study of the average number of generations needed by EBNA-Local and EBNA-Exact to find the optimum.

Experiments were conducted for three functions, *OneMax*, *Deceptive*3 and *SixPeaks*. In the first function, there are no interactions between the variables. In the rest, interactions arise between variables that belong to the same definition set of the function.

In order to analyze the average number of generations to find the optimum needed by EBNA-Local and EBNA-Exact, we start with a population of 10 individuals. The population size is increased by 10 until a maximum population size of 150 is reached. For each possible combination of function, number of variables n, and population size N, 50 successful runs are recorded. For each execution of the algorithm, a maximum of 10^5 evaluations are allowed.

For the *OneMax* function, we conducted experiments for $n \in \{15, 20\}$. In order to increase the accuracy of the curves shown Fig. 2.2, for $n = 15$ we exceptionally conducted 100 experiments. In general, this number of runs is

not feasible due to the computational cost of the exact learning. The results of the experiments for $n = 15$ are shown in Fig. 2.2 (a) and the results for $n = 20$ are shown in Fig. 2.2 (b).



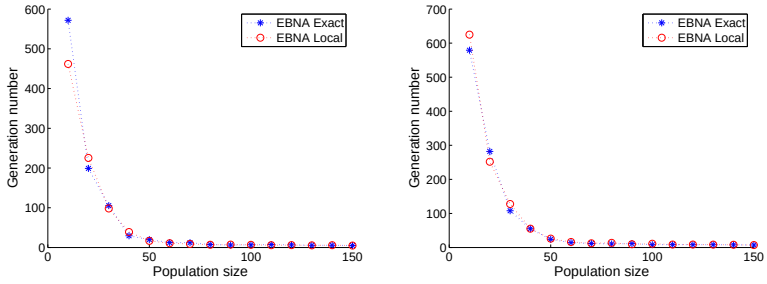
a) b)

Fig. 2.2. Time complexity analysis for function $OneMax$, (a) $n = 15$ and (b) $n = 20$.

The analysis of Fig. 2.2 reveals that both algorithms exhibit the same time complexity pattern. However, EBNA-Exact needs, in general, a higher number of evaluations than EBNA-Local to find the optimal solution for the first time. The difference in the number of generations is less evident when the population size approaches to 150. For this simple function, it seems that the error in the learning of the model introduced by the approximate learning algorithm is beneficial for the search. It could be possible that the structures learned by the exact method are overfitted to the specific set of selected individuals and then, the algorithm loses some desirable properties such as generalization ability or diversity. Although this point deserves an specific study, it will be discussed later when the structures learned by the algorithm are analyzed.

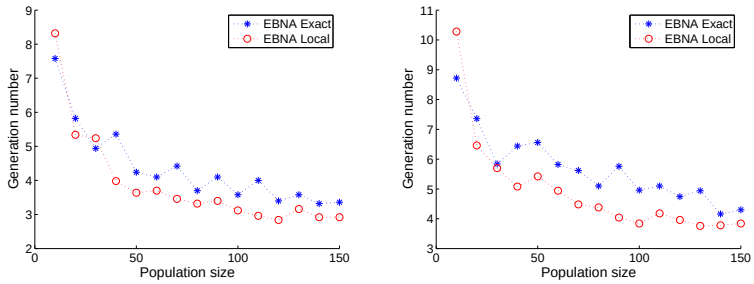
The results for the function $Deceptive3$ are shown in Fig. 2.3. In this figure we can observe that both algorithms practically have identical curves. For this function, and for the values of n investigated, the influence of the exact learning is not relevant. We can anticipate that the structures that the algorithms learn during the search are similar. For this function, both algorithms need a high number of generations to reach the optimum when the smallest population sizes are used. However, after $N = 50$ approximately, the time complexity of the algorithm stabilizes. These results suggest that the algorithm is needing a population size larger than certain threshold in order to be effective.

The results for the $SixPeaks$ function are presented in Fig. 2.4. It can be observed that EBNA-Local is able to constantly reach the optimum earlier than EBNA-Exact. From this analysis, we can say that the accuracy of the exact learning does not seem beneficial to solve $SixPeaks$ function. As previously commented, we argue that the main reason for this behavior of



a) b)

Fig. 2.3. Time complexity analysis for the function *Deceptive*3, (a) $n = 15$ and (b) $n = 18$.



a) b)

Fig. 2.4. Time complexity analysis for function *SixPeaks*, (a) $n = 14$ and (b) $n = 16$.

the EBNA-Exact is related to the overfitting phenomenon. The exact algorithm seems to be highly sensitive to the specific solutions in each population and therefore, it is describing the random errors of the samples. Nevertheless, this fact seems to have different consequences depending on the problem. The different behaviors that the algorithms have exhibited here will be further analyzed when discussing the structures of the models learned during the search.

2.5 Convergence reliability

In the analysis of the convergence reliability, we focus on the critical population size needed by the EDAs to achieve a predefined convergence rate. In the experiments conducted, the goal was to determine the minimum population size needed by the two different variants of EBNA to find the optimum in 20 consecutive runs. We investigated the behavior of the algorithms for

functions $Cuban5$ ($n = 13$), $SixPeaks$ ($n \in \{10, 12, 14\}$) and $Deceptive3$ ($n \in \{9, 12, 15\}$).

The algorithm begins with a population size $N = 16$ which is doubled until the optimal solution has been found in 20 consecutive runs. The maximum number of evaluations allowed is 10^4. For each objective function and value of n, the final results are the average of 25 experiments. Table 2.1 shows the mean and standard deviation of the critical population size found.

Table 2.1. Mean, standard deviation and p-value of the critical population size for different functions and number of variables.

Function	n	EBNA-Exact		EBNA-Local		T-Test
		mean	std	mean	std	p-value
$Cuban5$	13	118.40	53.07	109.44	57.26	0.57
$SixPeaks$	10	**153.60**	52.26	215.04	109.11	0.014
$SixPeaks$	12	**209.92**	110.11	389.12	249.19	0.019
$SixPeaks$	14	**312.32**	133.64	604.16	318.97	0.001
$Deceptive3$	9	**135.68**	38.40	168.96	60.94	0.025
$Deceptive3$	12	**168.96**	60.94	261.12	86.50	0.001
$Deceptive3$	15	**220.16**	58.66	296.96	95.79	0.001

Table 2.1 shows that, for function $Cuban5$, EBNA-Exact requires a slightly higher population size than EBNA-Local. The picture drastically changes for functions $SixPeaks$ and $Deceptive3$, for which EBNA-Exact needs a notably smaller population size. This difference is particularly evident for function $SixPeaks$. Another observation is that the standard deviation of EBNA-Local is always higher than that of EBNA-Exact. Since the only difference between EBNA-Exact and EBNA-Local is the learning method, the difference of behaviors seems to be due to the ability of EBNA-Exact to learn a more accurate model of the dependencies. Therefore, at least for functions $SixPeaks$ and $Deceptive3$, learning a more accurate model determines a better performance of EBNA in terms of convergence reliability.

To determine if the population sizes obtained for each algorithm are significantly different, we have carried out a Student's t-test over the two sets of 25 population sizes for each function and value of n. In the last column of Table 2.1, the probability values of the test are reported.

If we consider a significance level of 0.05, we would have to reject the null hypothesis for all cases except for $Cuban5$, where there are not significant differences. A possible explanation of the similar behavior achieved by both algorithms for $Cuban5$ will be presented in the next section, where the structures of the probabilistic models learned by the algorithms are studied. Moreover, for $SixPeaks$, using the highest value of n, and for $Deceptive3$, using the two highest problem sizes, the difference between the algorithms is statistically significant at the 0.01 level.

2.6 Problem-knowledge extraction from Bayesian networks

The objective of this section is to investigate the difference between the structures learned using exact and approximate learning algorithms and the relationship of that structures with the interdependences among the variables that the function defines. Throughout the dissertation, when dealing with additively decomposable functions (ADFs), we assume that the variables belonging to the same sub-functions interact i.e. there is an interaction between any pair of variables of the same sub-function. We also analyze the changes in the pattern (number and type) of the dependencies captured by the algorithms during their evolution.

2.6.1 Probabilistic models as a source of knowledge about the problem

Although the main objective in EDAs is to obtain a set of optimal solutions, the analysis of the models learned by the algorithms during the evolution can reveal previously unknown characteristics of the problem. There is a variety of information that can be obtained from the analysis of the models. Just to cite a few examples, it could be possible to extract:

- A description of sets of dependent or interacting variables.
- Probabilistic information about most likely configurations for subsets of the variables of the problem which can be translated into most-probable partial solutions of the problem.
- Evidence on the existence of different types of problem symmetry.
- Identification of conflicting partial solutions in problems with frustration.
- In addition, by considering the change of the models during the evolution (a dynamical perspective), it is also possible to identify patterns in the formation of optimal structures.

In this regard, a central problem is the design of methods for extracting and interpreting this information from the models. There are a number of approaches that have been proposed to deal with this issue. We identify three main sources of information:

1. The structure of the Bayesian network: By inspecting the topological characteristics of the graphs (e.g. most frequent arcs or conditional independences), we identify structural relationships between the variables.
2. The probabilistic tables of the Bayesian networks: By analyzing the probability associated to variables linked in the network, it is possible to identify both promising and poor configurations of the partial solutions.
3. Most probable configurations given the network: These are the solutions with the highest probability given the model. Thus, they condense the structural and parametrical information stored by the Bayesian network and have not necessarily been generated during the evolution of the EDA.

In this chapter we focus on the analysis of network structures.

2.6.2 Analysis of the structures learned by EBNA

In order to investigate the type of dependencies learned by EBNA-Exact and EBNA-Local, we saved the structures of the Bayesian networks learned during the evolutionary process for functions $Deceptive3$, $SixPeaks$, $Cuban5$, CPF and $Protein$.

In the following experiments, we start by running EBNA-Local and EBNA-Exact until obtaining 30 executions in which the optimum was found. The stopping criterion is a maximum number of 10^5 evaluations. In each of these experiments, the structures of the Bayesian networks learned in each generation are stored. From these structures, the frequency in which each arc appeared in the Bayesian network was calculated. Since we are not interested in the direction of the dependencies, we consider the two arcs that involve the same pair of variables. The matrices that store this information are called frequency matrices.

Two different ways of showing the information contained in the frequency matrices are used. The first way to represent the frequencies is using images where lighter color indicate a higher frequency. As another means to visualize the patterns of interactions, we use contour maps in which dependencies with a similar frequency are joined with lines. In this way, it is possible to identify areas of similar strength of dependency. In addition, the number of contours is a parameter that can be tuned to focus the attention on the set of the strongest dependencies.

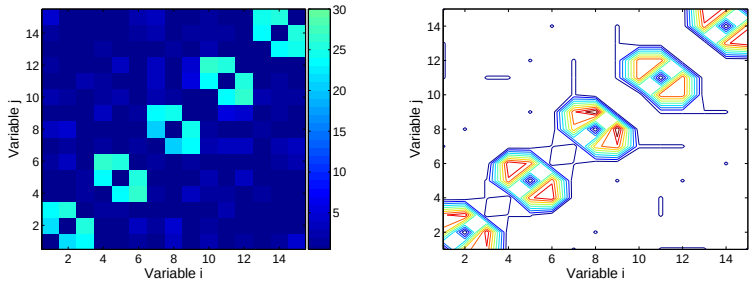
In the following, for each function and variant of EBNA employed, two figures are shown. The first figure shows the image graph of the dependencies learned by the model in the last generation and contained in the corresponding frequency matrix. The second figure shows the contour graph corresponding to a matrix that stores all the arcs learned by all the models during the evolution. We call this second matrix the cumulative frequency matrix. In order to fairly compare both algorithms using the contour figures, we normalized the frequencies of the arcs by the highest value among the two cumulative matrices learned by each algorithm. The normalized values are later discretized in ten levels. Thus, the contour lines refer to the same levels of frequencies.

2.6.2.1 Results for *Deceptive*3 and *SixPeaks* functions

In this section, the results for $Deceptive3$ with $n = 15$ and $SixPeaks$ with $n = 16$ are presented. We discuss the behavior exhibited by these functions and analyze some patterns identified in the structures of the models learned. We also try to link this behavior with previous results shown in this chapter.

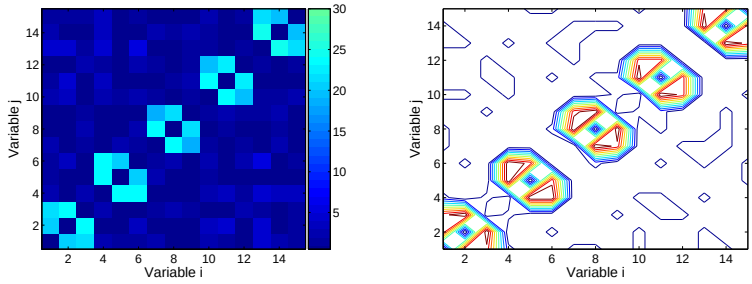
We start by using a population size of 150, which was the highest population size used on the complexity experiments shown in previous sections. Fig. 2.5 and 2.6 respectively show the frequency matrices corresponding to

EBNA-Local and EBNA-Exact for function *Deceptive*3. Both algorithms are able to capture the dependencies corresponding to the problem interactions. This fact may explain the similar behavior exhibited in the time complexity experiments.



a) b)

Fig. 2.5. Frequency matrices calculated from the models learned by EBNA-Local for function *Deceptive*3 with $N = 150$ (a) Last generation (b) All generations.

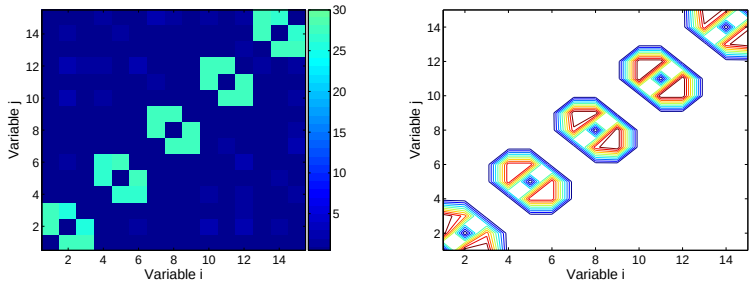


a) b)

Fig. 2.6. Frequency matrices calculated from the models learned by EBNA-Exact for function *Deceptive*3 with $N = 150$ (a) Last generation (b) All generations.

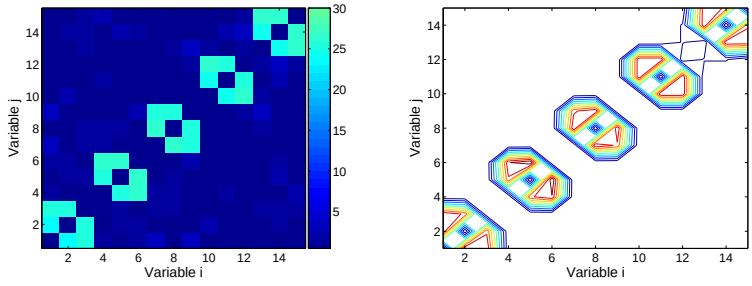
It can be noticed that the models include a number of dependences which are not explicitly determined by the function structure. This is particularly evident for the EBNA-Exact algorithm and is explained by the fact that exact learning is more sensitive to the overfitting of the data when the population size is small. Therefore, we increase the population size to $N = 500$ and repeat the same experiment for this function. The frequency matrices obtained are shown in Fig. 2.7 and 2.8. They reveal the effect of increasing the population size in the dependencies learned. It can be appreciated that spurious correla-

tions have almost disappeared from the models. Both algorithms are able to learn more accurate models with a population size of $N = 500$. This fact is particularly evident in the contour figures.



a) b)

Fig. 2.7. Frequency matrices calculated from the models learned by EBNA-Local for function *Deceptive*3 with $N = 500$ (a) Last generation (b) All generations.

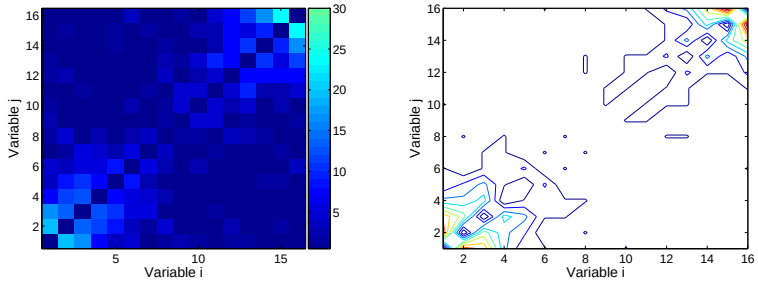


a) b)

Fig. 2.8. Frequency matrices calculated from the models learned by EBNA-Exact for function *Deceptive*3 with $N = 500$ (a) Last generation (b) All generations.

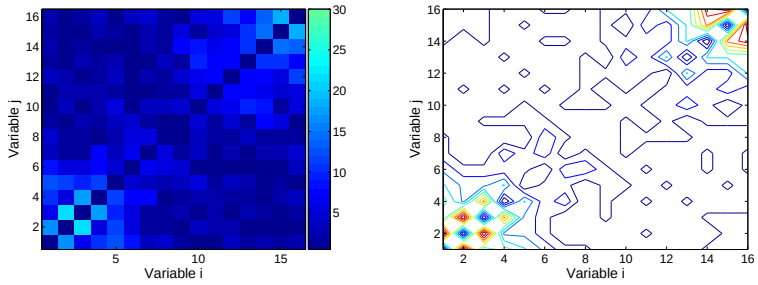
We conduct a similar analysis for function *SixPeaks*. Fig. 2.9 and 2.10 respectively show the frequency matrices calculated for EBNA-Local and EBNA-Exact with a population size of $N = 150$. It can be seen that both algorithms are unable to learn the accurate structure. As in the case of the *Deceptive*3 function, EBNA-Exact seems to learn more spurious dependencies than EBNA-Local. This fact is specially evident in the contour plot (Fig. 2.10(b)). In this case, the patterns of dependencies is spread along the matrix while dependencies learned by EBNA-Local are grouped around the

diagonal. This fact may explain the better results achieved by EBNA-Local in the time complexity experiments done for this function.



a) b)

Fig. 2.9. Frequency matrices calculated from the models learned by EBNA-Local for function $SixPeaks$ with $N = 150$ (a) Last generation (b) All generations.



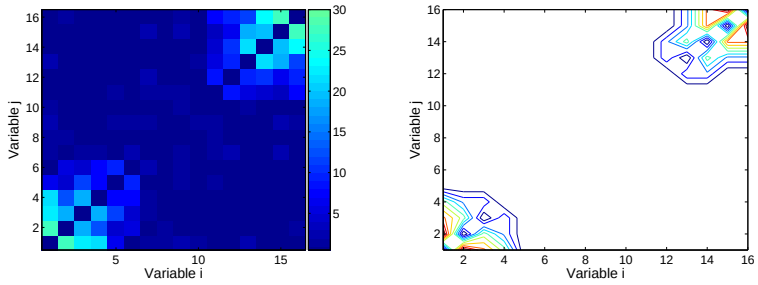
a) b)

Fig. 2.10. Frequency matrices calculated from the models learned by EBNA-Exact for function $SixPeaks$ with $N = 150$ (a) Last generation (b) All generations.

Insufficient population size might be the main reason to explain the poor quality in the mapping between the function structure and the model structure. Therefore, we repeat the experiment using a population size $N = 500$. Results are shown in Fig. 2.11 and 2.12.

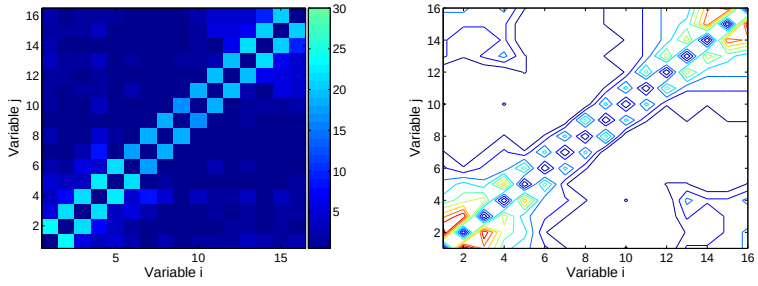
The images reveal that, by increasing the population size, EBNA-Exact is able to learn a very accurate structure. The model learned captures all the short-order dependencies of the function. This fact is corroborated by inspecting the contour graph in Fig. 2.12(b) where there is evidence that exact learning has gains in accuracy with respect to a smaller population size. On the other hand, EBNA-Local does not achieve a similar improvement. Further-

more, the accuracy of the approximation is lower than when a population size $N = 150$ was used, as can be seen by comparing Fig. 2.9 and 2.11. Here, the approximate technique could be experiencing certain learning limits. These issues will be treated in Chapter 4.



a) b)

Fig. 2.11. Frequency matrices calculated from the models learned by EBNA-Local for function $SixPeaks$ with $N = 500$ (a) Last generation (b) All generations.



a) b)

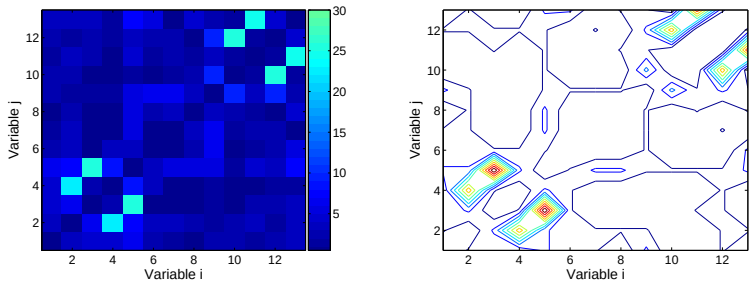
Fig. 2.12. Frequency matrices calculated from the models learned by EBNA-Exact for function $SixPeaks$ with $N = 500$ (a) Last generation (b) All generations.

2.6.2.2 *Cuban5* function

We analyze the $Cuban5$ function with $m = 1$ and $n = 13$. For $m = 1$, $Cuban5$ is equal to the sum of three subfunctions:

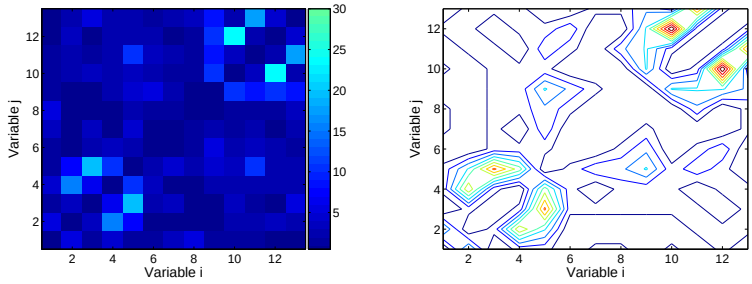
$$Cuban5(\mathbf{x}) = F^5_{cuban1}(s_0) + F^5_{cuban2}(s_1) + F^5_{cuban1}(s_2). \tag{2.6}$$

The interactions are determined by two different functions, F^5_{cuban1} and F^5_{cuban2}. Therefore, we expect $Cuban5$ to exhibit a different pattern of interactions than those previously analyzed. As in previous experiments, we start with a population size $N = 150$. The frequency matrices corresponding to EBNA-Local and EBNA-Exact are respectively shown in Fig. 2.13 and 2.14.



a) b)

Fig. 2.13. Frequency matrices calculated from the models learned by EBNA-Local for function $Cuban5$ with $N = 150$ (a) Last generation (b) All generations.



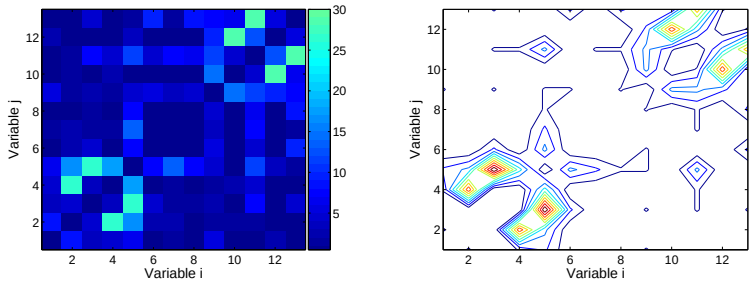
a) b)

Fig. 2.14. Frequency matrices calculated from the models learned by EBNA-Exact for function $Cuban5$ with $N = 150$ (a) Last generation (b) All generations.

It can be seen in the images calculated from the frequency matrices of the last generation that only some of the dependencies determined by function F^5_{cuban1} are captured by both algorithms. However, the cumulative frequencies clearly show the existence of dependencies related to function F^5_{cuban2}. There are no significant differences between both EDAs.

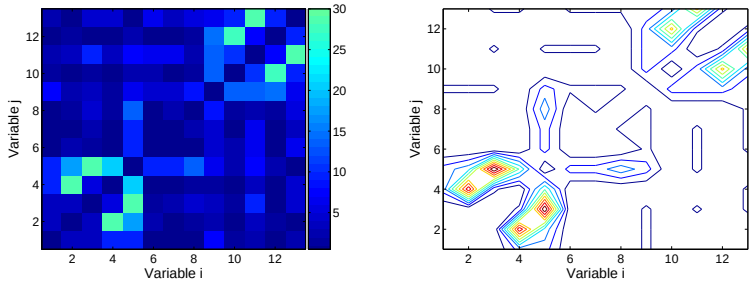
The frequency matrices obtained by increasing the population size to $N = 500$ are shown in Fig. 2.15 and 2.16. In this case, the dependencies determined

by function F^5_{cuban2} are easier to recognize in the frequency matrices of the last generation. However, although the population size is larger, for this function both algorithms have learned a similar structure. This fact could explain the absence of significant differences between both algorithms in the study of convergence reliability presented in Section 2.5.



a) b)

Fig. 2.15. Frequency matrices calculated from the models learned by EBNA-Local for function $Cuban5$ with $N = 500$ (a) Last generation (b) All generations.



a) b)

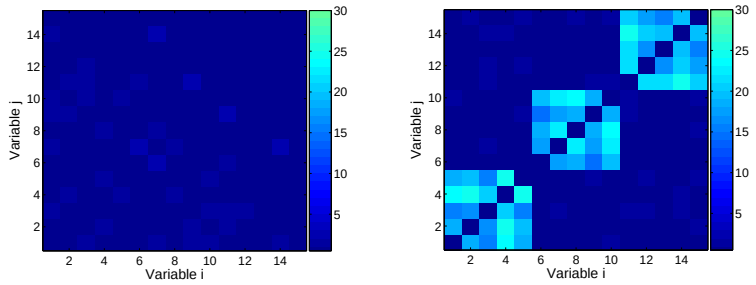
Fig. 2.16. Frequency matrices calculated from the models learned by EBNA-Exact for function $Cuban5$ with $N = 500$ (a) Last generation (b) All generations.

2.6.2.3 Results for the *CPF* function

The CPF function represents another interesting class of functions. It has been shown that Bayesian network based EDAs such as BOA are deceived by this function. Being CPF an additive separable function of bounded complexity, BOA has an exponential scaling for it. Furthermore, in Coffin and Smith

(2007) it is shown that increasing the population size does not always produce an improvement in the algorithm's behavior. Authors point to the fact that the learning algorithm used by BOA may fail to detect the higher order type of interactions that occurs in the CPF function. Nevertheless, Chen and Yu (2009) claim that the compact genetic algorithm, which does not conduct linkage learning, is able to solve this problem within a polynomial number of function evaluations to the problem size.

We will investigate whether there are differences between exact and local learning for the CPF function with parameters: $n = 15$, $k = 5$, $c_{odd} = 5$ and $c_{even} = 0$. For these parameters, the optimum can be reached in 2^{12} different points. As a consequence, it is likely that EBNA reaches the global solution in the first generation. On the other hand, the limitations of the exact learning algorithm do not allow to deal with a higher number of variables. Therefore, we slightly change the experimental design for this specific function. We will only analyze the models learned in the first generations of the EDAs, disregarding whether the optimum has been found or not. The models have been calculated using 30 independent experiments. Regarding the population size, we started with $N = 150$. For this and some higher values of the population size, none of the algorithms was able to recover any type of structure. However, the differences appear with $N = 1000$. The frequency matrices calculated for EBNA-Local and EBNA-Exact are shown in Fig. 2.17.



a) b)

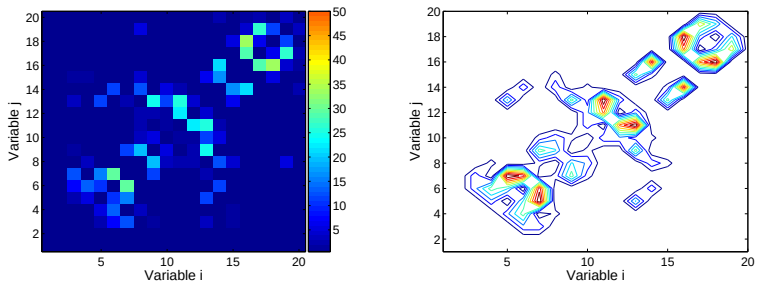
Fig. 2.17. Frequency matrices calculated from the models learned in the first generation of the EDAs for function CPF with $N = 1000$ (a) EBNA Local (b) EBNA-Exact.

Surprisingly, EBNA-Exact was able to recover an almost perfect structure while EBNA-Local was not. These results reveal that, for problems such as CPF, an accurate learning of the model might be essential to recover the correct structure of the problem. It also shows that the population size required to discover the problem structure is higher than in previous additive functions considered. In addition, with a sufficient populations size, the exact learning seems to overcome the limitations exhibited by the approximate learning.

2.6.2.4 HP protein model

The HP model has been used as a benchmark for studying different issues related with the behavior of EDAs (Santana et al. (2008a); Chen et al. (2009)). It is a non-binary, non-decomposable problem for which extensive investigation using evolutionary and other heuristic algorithms have been conducted (see Cutello et al. (2007) and references therein). We use one instance of the HP model to investigate the impact of exact learning. Fig. 2.1 shows one optimal folding for the chosen sequence $HPHPPHHPHPPHPHHPPHPH$.

In the evaluation of the HP model, two variants are considered. In the first one, infeasible individuals are assigned a penalty. In the second variant, individuals are first repaired and after that the *Protein* function is used to evaluate them. In all the experiments conducted for this function, $N = 200$ and 50 independent experiments of EBNA-Local and EBNA-Exact were run.

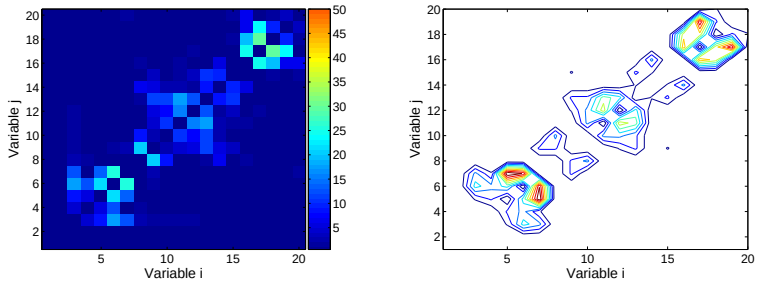


a) b)

Fig. 2.18. Frequency matrices calculated from the models learned by EBNA-Local for the *Protein* function, repairing procedure with $N = 200$ (a) Last generation (b) All generations.

Since the *Protein* function is not decomposable, a detailed description of the problem structure is not available and we can not contrast the dependencies learned with a perfect model of the interactions. However, previous research on the application of EDAs to the HP problem (Santana et al. (2008a)) has shown that important dependencies between adjacent variables arise. These dependencies are in part determined by the codification used, in which each residue's position depend on the position of the previous two. Thus, the objective of our experiments is twofold. Firstly, to compare the class of models learned by EBNA-Local and EBNA-Exact. Secondly, to investigate the effect that the application of the repair mechanism has in the number and patterns of the interactions learned by the EDAs.

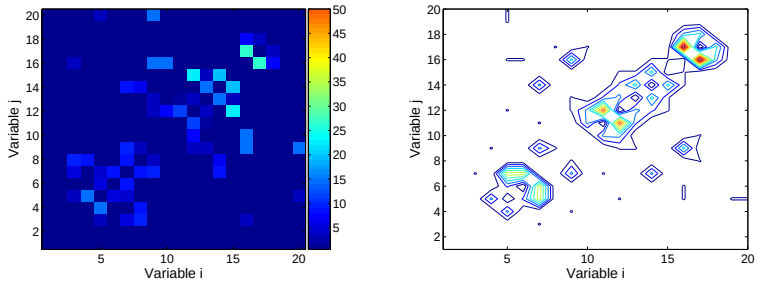
Fig. 2.18 and 2.19 respectively show the frequency matrices learned by EBNA-Local and EBNA-Exact when the repairing procedure is applied.



a) b)

Fig. 2.19. Frequency matrices calculated from the models learned by EBNA-Exact for the *Protein* function, repairing procedure with $N = 200$ (a) Last generation (b) All generations.

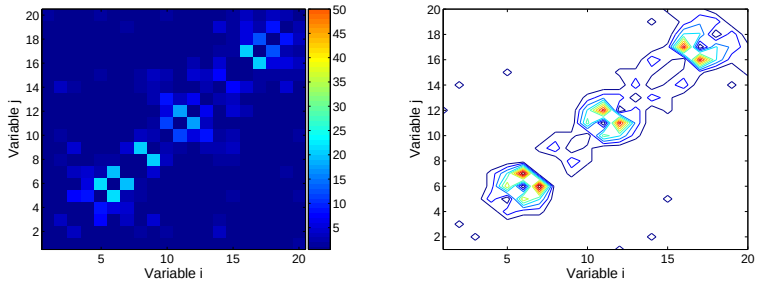
Fig. 2.20 and 2.21 show frequencies corresponding to the variant in which the repairing procedure is not applied.



a) b)

Fig. 2.20. Frequency matrices calculated from the models learned by EBNA-Local for the *Protein* function, without repairing procedure with $N = 200$ (a) Last generation (b) All generations.

An analysis of the figures reveal that EBNA-Exact learns a pattern of interactions more localized around the diagonal representing the dependencies between adjacent variables. The dependencies found by EBNA-Local are more spread-out, away from the diagonal. We also observe some differences due to the application of the repairing procedure. These differences are particularly noticeable from the analysis of the contour graphs. Taking as an accuracy criterion the connectedness of the adjacent variables in the problem representation, we see that repairing helps EBNA-Local to learn more accurate structures. Without repairing, the pattern of interactions is more fragmented.



a) b)

Fig. 2.21. Frequency matrices calculated from the models learned by EBNA-Exact for the *Protein* function, without repairing procedure with $N = 200$ (a) Last generation (b) All generations.

However, repairing does not help EBNA-Exact, which is able to recover a more connected structure without the application of the repairing procedure.

2.7 Conclusions

In this work we have accomplished a detailed analysis of the use of exact learning of the Bayesian network structure in the study of EDAs. We have conducted systematic experiments of different kinds for several functions.

Results show that learning the optimal Bayesian network at each generation does not necessarily improve the performance of the EDA. We argue that this is mainly due to the overfitting phenomenon. In the analysis of the structural models, we have seen that the exact method tends to learn more dependences not explicitly described in the formulation of the function than the approximate technique, especially when a small population size is used. It indicates that the structures obtained by exact learning could be describing the random errors of the samples along with the underlying relationships among the problem variables. Nevertheless, the impact of this phenomenon depends on the characteristics of the problem. Although in Wu and Shapiro (2006); Lima et al. (2008, 2011) overfitting is studied in the context of EDAs, we believe that more research is needed to understand this issue. In particular, the overfitting that takes place when an exact structural learning is accomplished could be a subject for further research.

In general, the results show that the type of learning algorithm (whether exact or approximate) may produce significant differences in the models learned and in the performance of the EBNA. In particular, when the exact learning is provided with enough information through the selected individuals, it is able to learn accurate structural models closely related to the structure of the problem. However, the approximate technique is not always able to

learn accurate structures even when a large population size is used. This fact suggests the existence of certain learning limits related to the approximate techniques. This issue will be studied more in-depth in Chapter 4. In addition, this fact is important because usually Bayesian networks learned using approximate algorithms are thought to accurately reflect the dependencies that arise in the population. As the example of the CPF function illustrates, this might not be the case for functions with a particular type of high order dependencies.

On the other hand, we have shown that whenever the size of the problem is manageable, exact learning of Bayesian networks is a more appropriate option for theoretical analysis of the probabilistic dependencies. We have shown that the analysis of the probabilistic models can reveal the effect that some EDA components, such as repairing procedures, have in the arousal of dependencies. By using exact learning, we have shown the critical effect that an inadequate population size may have to capture accurate probabilistic models.

3

Optimization Problems Defined on Complex Networks

3.1 Introduction

This chapter follows the line opened in the previous chapter regarding the relationship among problem structure, EDA behavior and the models learned by the algorithm. Nevertheless, although the current analysis is related to similar concepts and motivations, a completely different methodological approach is developed. The study conducted in this chapter is mainly driven by the topological characteristics of the structures that the interactions among the variables of the problems provide. Taking as reference the emergent field of complex networks, we generate a wide spectrum of networks that will serve as problem structures. Then, the impact that the topological characteristics of those networks have, both in the hardness of the optimization problem and in the behavior of the EDA, is analyzed.

Nowadays, the study of networks pervades many scientific disciplines, from biology to statistical physics or society (Strogatz (2001); Albert and Barabási (2002)). For several years, the topological properties of real-world networks have been described through the theory of random graphs (Erdös and Rényi (1959)). However, advances in this field have shown that networks from very different domains display certain patterns of order and self-organization. Specifically, the topological characteristics of many real-world networks are midway between those of regular lattices and random graphs (Watts and Strogatz (1998); Albert and Barabási (2002)). In order to quantify these characteristics, concepts and measures such as path length, clustering coefficient or degree distribution have been proposed (Albert and Barabási (2002)). These subjects are studied in the field of complex networks (Erdös and Rényi (1959); Watts and Strogatz (1998); Albert et al. (2000)) and constitute a suitable framework for the study of complex systems (Dorogovtsev et al. (2008)). In the same manner, networks emerge from the structural component provided by the interactions among variables in many optimization problems. Moreover, the complexity of these interactions can dramatically influence the hardness of the problem (Dorogovtsev et al. (2008); Echegoyen et al. (2012)).

Therefore, the study of the topological characteristics of the function structure and their impact in the optimization techniques is a relevant issue in the development of more efficient and accurate optimization tools.

As discussed in the previous chapter, a natural link arises between the structures used by EDAs based on Bayesian networks and the structure formed by the interactions among the variables in the optimization problem. In this chapter, we study the impact that varying the topological characteristics of the function structure has in EBNA implementing Algorithm B. The functions are defined through a network structure and interaction strengths for each couple of variables (edges in the network). Thus, throughout a procedure based on Watts and Strogatz (1998), we gradually change the properties of the function structure in order to analyze EDAs in different topologies such as regular grids, small-worlds and random graphs.

In particular, the current chapter is mainly focused on the following two questions. i) How does the hardness of the problem change according to the topological characteristics of its structure? and ii) how are the structures learned by EBNA in order solve the problems depending on the topological properties of the problem structures? In order to interpret the behavior of the algorithm in terms of problem difficulty, we deal with the population size that allows the algorithm to reach the optimum with reliability. Thus, it is considered that the greater the reliable population size, the higher the difficulty of the problem will be. By using this population size and also the number of evaluations to reach the optimal solution for the first time, we try to shed light on the first question. As regards the second question, we analyze the structures learned by EBNA both by using measures taken from the field of complex networks and studying the similarity between EBNA structures and the original function structure.

The remainder of the chapter is organized as follows. Section 3.2 introduces definitions of graphs and details about the creation of optimization functions from a class of complex networks. Section 3.3 explains the experimental design. Section 3.4 presents and discusses the results of the different experiments. Section 3.5 discusses relevant previous works related with this chapter. Finally, Section 3.6 draws the conclusions obtained during the study.

3.2 The function benchmark: From regular grids to small-worlds and random graphs

The functions considered in this chapter are defined on networks where the variables interact in pairs according to the edges. Formally, a network is represented by a graph. Thus, let $G = (V, E)$ be an undirected graph (UG), where $V = \{1, .., n\}$ is a set of vertices which correspond to index variables in the function, and the set of edges E is a subset of the set $V \times V$ of pairs of distinct vertices. For each network or graph we define an additively decomposable function as follows,

$$f(\boldsymbol{x}) = \sum_{(i,j)\in E} g_{ij}(x_i, x_j) \tag{3.1}$$

where there is an interaction function g_{ij} for each edge in the graph.

In this chapter, the following graph-related definitions will be used. Two nodes connected by an edge are called adjacent or neighbors. We denote the number of neighbors or degree of the i-th node by k_i. The shortest path length between nodes i and j is denoted by d_{ij}. The following two concepts are considered to analyze the topological characteristics of networks. Firstly, the *small-world* concept which describes the fact that in most real-world networks there exists a relatively short path between any two nodes. The characteristic path length L is proposed in Watts and Strogatz (1998) as a measure of this property. L is defined as the number of edges in the shortest path between two vertices, averaged over all pairs of vertices. Thus, the characteristic path length is calculated as $L = \frac{1}{n(n-1)} \sum_{i\neq j} d_{ij}$. Secondly, the *clustering* concept is related to the cliqueness of the network. The tendency to cluster is quantified by the clustering coefficient C (Watts and Strogatz (1998)). For a node i having k_i neighbors, there could be $k_i(k_i - 1)/2$ possible edges between the neighbors (without including node i). The clustering coefficient of node i, denoted by C_i, is the ratio between the number ε_i of edges that exist in the graph among its k_i neighbors (without including node i) and the total number of possible edges between the neighbors. Then, we have $C_i = \frac{2\varepsilon_i}{k_i(k_i-1)}$. The clustering coefficient of the whole network is the average of all C_i's.

The way in which the set of network topologies are generated for this chapter is based on Watts and Strogatz (1998). That work explores models of networks which can be found between regular grids and random graphs. They claim that the networks of this middle ground are able to reproduce characteristics of real-world networks which are present in biological, technologic or social domains. This type of networks is called small-world networks and its main property is that nodes are grouped into clusters with small path lengths between any pair of nodes. In order to interpolate between regular and random networks, the authors propose a random rewiring procedure. They start by considering a ring of n vertices, each connected to its k nearest neighbors by undirected edges. At each step of the procedure, the probability p of rewiring edges of the ring is progressively increased from $p = 0$ (regularity) to $p = 1$ (chaotical).

In the current chapter, we consider a slightly different model (Santana et al. (2008b)). The main difference is that we start from a 2-dimensional grid with periodic boundary conditions. This allows us to represent systems disposed on regular grids such as Ising spin glass (Ising (1925)) which represents a classical challenge in optimization (Pelikan and Goldberg (2003); Hauschild et al. (2009)) and it is NP-complete in its general form (Barahona (1982)). The details of our function benchmark are given in the next section.

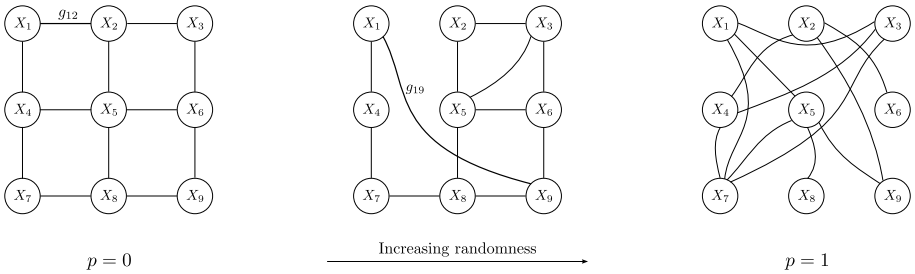


Fig. 3.1. Different stages of the rewiring procedure.

3.3 Experimental design

3.3.1 Functions

In order to generate the set of functions, we start from a toroidal 2-dimensional grid denoted by G_0. The number of nodes is $n = m \times m$ where $m = 7$ is the dimension of the grid. In G_0, each node has 4 neighbors, being the total number of edges $2n$. Hence, we have $2n$ interaction functions g_{ij}. Since we work with binary variables, each interaction function g_{ij} has four possible values which are uniformly chosen from $(0, 1)$.

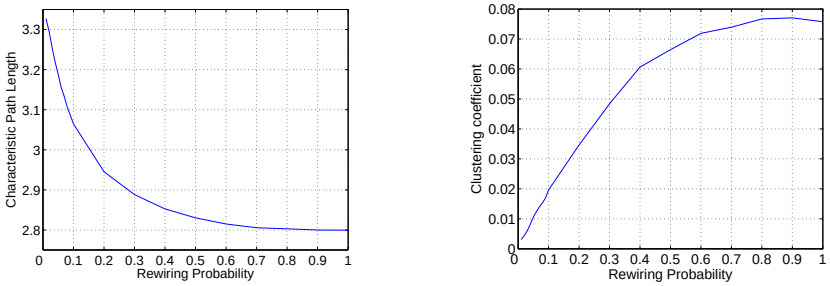
From G_0, a collection of graphs is generated by rewiring the original edges with probability p. To generate a rewired graph G_p from G_0, each edge is visited and a decision about rewiring is made with probability p. If the edge is rewired, the variables linked by the corresponding edge are modified but the function values are kept intact. Duplicate edges are forbidden. This rewiring procedure is illustrated in Fig. 3.1. The original grid (without periodic boundaries for simplicity) is shown on the left of the figure, a possible rewired graph is shown in the middle and a random graph is represented on the right. To explain how the function values are managed, we focus on the interaction denoted by g_{12} in the regular grid, and by g_{19} after rewiring. We can see in Table 3.1 that this interaction has the same function values before and after rewiring, although they are associated to different variables.

Since the transition to small-network topologies is said to start at small p values, we generate networks for values of $p \in \{0.01, 0.02, \ldots, 0.1\}$. In addition, and in order to observe the behavior of EBNA when the topology becomes a random graph, p is further increased by generating networks for $p \in \{0.2, 0.3, \ldots, 1.0\}$. For each value of p, 100 different graphs are generated by rewiring the edges from G_0 with probability p. The total number of structures generated starting from G_0 is 1900. In addition, as we have created 10 different functions, the total number of considered instances is 19000.

In Figure 3.2 we show different topological characteristics of the networks generated by the aforementioned procedure. We can see how the characteristic

Before			After	
x_1, x_2	g_{12}	$\rightarrow$	x_1, x_9	g_{19}
0, 0	0.2		0, 0	0.2
0, 1	0.9		0, 1	0.9
1, 0	0.3		1, 0	0.3
1, 1	0.5		1, 1	0.5

Table 3.1. Function values of an interaction before and after rewiring.



(a) Path length (b) Clustering coefficient

Fig. 3.2. Different measures taken from the networks generated by the rewiring procedure. (a) Characteristic path length and (b) clustering coefficient.

path length L (Figure 3.2(a)) falls off steeply at low values of p, but the decrease slows as p increases. Finally, L stabilizes at $L = 2.8$ around $p = 0.7$. On the contrary, the average clustering coefficient C (Figure 3.2(b)) steadily increases to $p = 0.9$ when it reaches the maximum value.

3.3.2 EBNA runs

Each of the generated problem instances has been solved by EBNA. The learning step is carried out by using Algorithm B (see Alg. 1.1). The rest of parameters are set as explained in Section 1.3.4. All runs for this work have been conducted using a population size given by the bisection method (Pelikan (2005)). This procedure returns a critical population size to reach the optimum with high reliability. The bisection method has two phases. The first phase establishes the bounds for the search of the final population size that the method will return. Starting from an initial size N (in our case $N = 16$), it is doubled until the EDA can find the optimal solution for, in this case, 10 out of 10 independent runs. The population size that passes this test is denoted by N_{max}. Thus, the upper bound for the second phase of the bisection method is N_{max} and the lower bound is $N_{min} = N_{max}/2$, where the test failed. Note that this first phase corresponds to the procedure carried out in Chapter 2 to conduct the analysis of convergence reliability.

Having established this range, the second phase refines the reliable population size N_{max} obtained in the first phase. In order to do so, the bisection searches a population size between N_{max} and N_{min} for which the EDA keeps reaching the optimum for 10 out of 10 runs in the following way. The midpoint $N_{mid} = (N_{max} + N_{min})/2$ is considered. If the EDA achieves 10 consecutive successful runs with N_{mid}, then N_{max} is decreased to N_{min}, otherwise N_{min} is increased to N_{mid}. This procedure continues until N_{min} and N_{max} are within a predefined threshold distance of each other. For the experiments of this chapter, this threshold is set to 10. After applying the bisection method for a given objective function, we have that N_{max} is close to being the smallest population size for which a specific EDA can reliably optimize the problem. We calculate this critical size for each problem instance and the final population size is the average over 10 successful bisection runs.

We only work with runs for which the optimal solution is reached. The stopping criterion for EBNA is to find the optimum for the first time. For each function, one EBNA run has been carried out with the population size given by bisection. All results shown are the average of 10 functions and 100 structures for a given rewiring probability p.

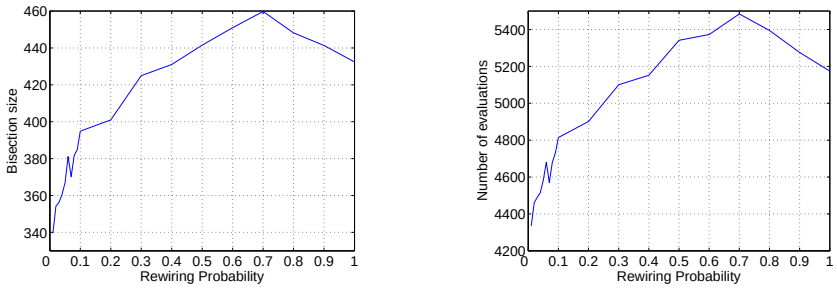
3.4 Empirical results

The experiments are divided into two groups. In the first one, we observe the performance of EBNA through the population size needed to reach the optimum with reliability (bisection size) and the number of evaluations to reach the optimum for the first time. In the second one, we show and discuss the influence of the topological characteristics of the function in the structures learned by EBNA and their relationship with the original structure of the function.

3.4.1 Analysis of the difficulty of the problems

In Fig. 3.3 we can see that the population size to reach the optimum with reliability and the number of evaluations to reach that solution for the first time (when bisection size is used) change in a closely related manner according to the rewiring probability. This indicates that the performance of EBNA is sensitive to the topological characteristics of the function structure. Since the curves of both charts have a similar shape, we can deduce that the number of generations remains practically constant during the rewiring procedure. It is the population size which varies depending on the randomness of the function structures. The use of larger population sizes suggests that the algorithm needs more information to learn the adequate models to solve the problems.

By comparing Fig. 3.3 with the plots in Fig. 3.2, we can observe that the bisection size and number of evaluations are related with the measures taken from the original networks of the function. On the one hand, as the



(a) Bisection sizes (b) Number of evaluations

Fig. 3.3. Evolution of the bisection size and the number of evaluations to reach the optimum throughout the different functions generated for each probability of rewiring.

clustering coefficient increases, obtaining the optimum becomes harder for EBNA because it needs a larger population and higher number of evaluations. However, there is not a full relation. When p is 0.7 or larger, the behavior changes: the clustering coefficient tends to stabilize while the bisection size and number of evaluations clearly decrease. After conducting several tests on the structures of the functions, we conjecture that this is due to a mixture of different factors which is difficult to detect in a single measure. This is an important issue that deserves a specific analysis.

On the other hand, there is an inverse relation between the characteristic path length (Fig. 3.2(a)) and the behavior of EBNA (Fig. 3.3). This inverse relation is especially remarkable from $p = 0$ to $p = 0.1$ because it is the range in which the path length clearly decreases while bisection size and number of evaluations increases dramatically. This fact indicates that the rewiring of a few edges in the regular grids sharply increases the hardness of the problem for EBNA.

Clearly, the easiest problems are those whose structures are regular grids whereas the hardest problems are in the zone of random graphs. The transition to small-world networks seems to moderately increase the difficulty of the problems. Interestingly, this type of "real-world" networks carry a medium complexity for EBNA.

3.4.2 Analysis of the learned structures

In this section we want to analyze the structures learned by EBNA from a global point of view. Thus, the descriptors used are averaged for all the structures learned during the run. We must take into account that the learned structures can vary from the first to the last generation and occasionally can be empty at the end of the runs due to the convergence of the population

(Echegoyen et al. (2012)). Moreover, some of the structures can be disconnected and, therefore, the distance between variables in different connected components of the network can not be defined. In this case, the couple of variables belonging to different connected components are not taken into account when calculating the characteristic path length.

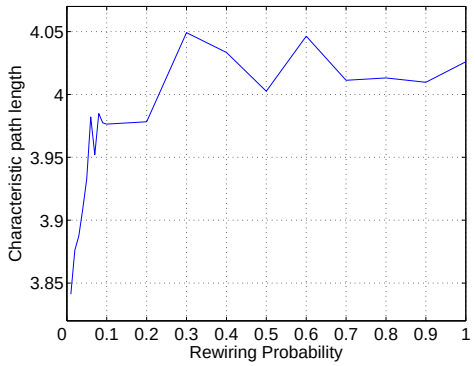
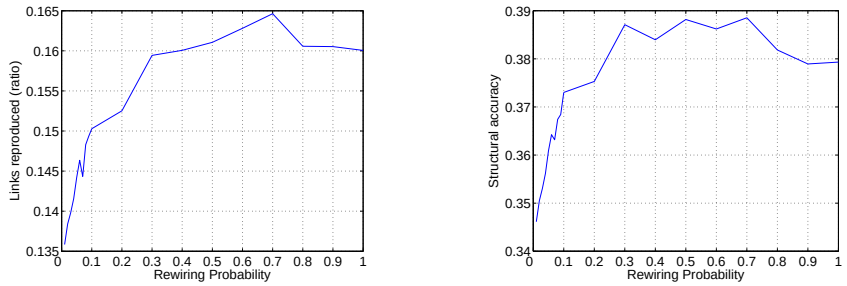


Fig. 3.4. Characteristic path length taken from the whole set of structures learned by EBNA in the optimization of the functions generated for each probability of rewiring.

The results for the clustering coefficient of the networks learned by EBNA are not shown because this measure hardly varies during the rewiring procedure. In fact, 63% of structures learned in the whole set of runs have a clustering coefficient equal to 0. It indicates that the algorithm does not need complex Bayesian network to solve the problems. Nevertheless, the characteristic path length of the networks learned by EBNA provides valuable information (Fig. 3.4). First of all, note that the changes in the characteristic path length are in the same scale of values as in the original problem structures (Fig. 3.2(a)). However, the behavior of both curves is opposite. We can observe the clear increase of the path length in the EBNA structures from $p = 0$ to $p = 0.1$ according to the decrease of this measure in the function structure (when the small-world networks are created). This result indicates that the rewiring of a few edges in the original grid produces a visible change in the path length of the EBNA structures.

Taking into account the results of bisection and number of evaluations shown in Fig. 3.3, we could interpret that the difficulty of the problems between $p = 0$ and $p = 0.1$ noticeably increases. These results combined with the path length presented in Fig. 3.4 suggest that, in order to address the difficulty of the problems with small-world topologies, EBNA learns structures with longer paths i.e. the algorithm tends to link more variables which

probably form small chains or trees. Interestingly, this increase of the paths stabilizes after passing the zone of the small-word networks.



(a) Ratio of links reproduced (b) Accuracy of the EBNA structures

Fig. 3.5. Two different measures to analyze the similarity between the structures learned by EBNA and the structure of the function.

In order to study more in-depth the degree of similarity between the structures learned by EBNA and the structure of the function, the following analysis is carried out. To this analysis the direction of the arcs in the Bayesian networks is not taken into account. Thus, given $G_f = (V_f, E_f)$ the graph of the original function structure and $G_s = (V_s, E_s)$ the undirected version of the EBNA structure at each step, we calculate i) the ratio $|E_f \cap E_s|/|E_f|$ of edges that match between both graphs over the total edges in the graph of the function and ii) the ratio $|E_f \cap E_s|/|E_s|$ of edges that match between both graphs over the total edges in the EBNA structure. The first measure indicates the proportion of dependences that EBNA captures from the function. The second measure indicates the proportion of these dependences in relation to the total number of edges learned by the algorithm. This measure is called *model structural accuracy* and has been previously used to analyze structures in EDAs based on Bayesian networks (Lima et al. (2008)). Fig. 3.5(a) and 3.5(b) respectively show the first and second aforementioned measures.

In Fig. 3.5(a), the curve describing the proportion of links that EBNA learns from the whole set of interactions in the function is closely related with the curves of population size and number of evaluations shown in Fig. 3.3. As we are using the bisection size in the runs, Fig. 3.5(a) gives the proportion of interactions among the function variables that EBNA needs to solve the problem with reliability. There seems to be a direct relationship among the topology of the problem structures, the difficulty of the problems and the dependences that the algorithm needs to learn in order to solve that problems.

The accuracy of the EBNA structures shown in Fig. 3.5(b) displays a similar behavior, especially in the first steps of the rewiring procedure. Nevertheless, the dependences that have been learned by the model and that are

not present in the function should be carefully interpreted. These probabilistic dependences could be useful to avoid critical independences between problem variables. In fact, dependences of the model that are not explicitly in the function could be more decisive to solve the problem than other dependences that match with the function structure.

According to both measures, we can see that, depending on the topological characteristics of the function structure, EBNA needs a different degree of precision in order to solve the problem with reliability. Finally, note that in all the figures of this section, the maximum increase occurs from $p = 0$ to $p = 0.1$, the range where the small-world networks are created.

3.5 Related work

The mapping between the problem structure and the dependences captured in the probabilistic models learned by EDAs has been a subject of research since the inception of EDAs (Bengoetxea (2003); Echegoyen et al. (2007); Hauschild et al. (2007); Mühlenbein and Höns (2005)). Measures of structural accuracy such as the one used in this chapter has been also proposed (Lima et al. (2008)).

It has been previously shown (Santana et al. (2008b)), that the behavior of the maximum loopy belief propagation (Pearl (1988)), an algorithm based on the use of probabilistic models and that can be applied for optimization, is sensitive to a variation in the path length and the clustering coefficient of the problem instances. The effect of other topological features of the problem structure, e.g. vertex degree distribution, in the characteristics of the evolutionary search has been also investigated for genetic algorithms (Khor (2009)) and EDAs (Santana et al. (2010)). However, the influence of the path length and clustering coefficient has not been previously analyzed in this class of algorithms. We also notice that it is possible to indirectly change the topological characteristics of the problem structure by varying other parameters of the function. The effects of these types of changes for EDAs, e.g. connectivity patterns in random additive functions (Gao and Culberson (2005)) or the overlapping parameter in the NK-landscapes (Pelikan et al. (2009)), have been investigated.

The field of complex networks is also an inspiration in the development of evolutionary algorithms. For instance, certain works (Payne and Eppstein (2006); Whitacre et al. (2008)) try to mimic in the population the self-organizing networks observed in nature to improve the behavior of the algorithms.

In relation to the work presented here, we also conducted a preliminary study of the impact that the function values have in the performance of the algorithm and in the structures learned. As previously commented, different works have analyzed the impact that the problem structure has in the algorithm. However, the influence of the function values has not been taken into

account. Furthermore, the synergy of these values with the structure could entail dramatic changes in the properties of the problem and in the ability of the algorithm to solve it. One of the experiments that we carried out is explained next to illustrate the idea. Based on the same range of function structures presented in this chapter, we considered three different classes of function values. In this case, the function values were generated as in the Ising spin glass problem, which will be formally introduced in Chapter 5. Then, we considered that i) all the interactions of the problem are positive, ii) the sign of the interactions is generated at random, iii) all the interactions are negative. The results showed that, on average, the most difficult problems are obtained when all the interactions are negative. In this regard, looking for the function values that, given a structure, generate the most difficult problems could be useful to shed light on these topics.

3.6 Conclusions

In this chapter, we have analyzed how EDAs based on Bayesian networks are influenced by the changes in the topological characteristics of the function structure. The range of function structures is generated following a rewiring procedure which is able to produce networks from regular grids to small-worlds and random graphs. In order to analyze the relationship between problem structure and EBNA behavior, we have introduced measures such as clustering coefficient or characteristic path length taken from the field of complex networks. In particular, we raised two main questions in Section 3.1 which are discussed below in light of the results obtained.

- How does the hardness of the problem change according to the topological characteristics of its structure?

In this chapter, the element that allows us to interpret the behavior of the algorithm in terms of problem difficulty is the population size given by bisection. Despite the limited number of variables used in these experiments, the influence of the topological characteristics of the function structure in EBNA is appreciable. Thus, problems with the same number of explicit interactions between variables can vary their difficulty according to the way in which those interactions are disposed. We have observed that the difficulty of the problems tends to increase with the randomness of the function structure. In fact, the problems disposed on regular grids are clearly the easiest of the whole range of problems.

- How do the structures learned by EBNA depend on the topological properties of the problem structures?

We have seen that the rewiring of a few edges from the original grid is enough to produce an increase of the path length in the networks learned

by EBNA. It suggests that the algorithm needs to link more variables in order to solve the problems after the first stage of the rewiring process. In addition, the number of function interactions that the algorithm needs to capture in order to reach the optimum tends to increase with the randomness of the problem structure. This reveals a relationship among the topology of the problem structure, the difficulty of the problems and the dependences that the algorithm needs to learn in order to solve the problems.

In general, an important fact must be pointed out: the results show a critical change in the behavior of the algorithm when the small-world networks are created (from $p = 0$ to $p = 0.1$). Thus, the rewiring of a few edges not only provokes the emergence of small-world networks but also provokes a remarkable change in the behavior of the EDA. After this stage, the results suggest that the hardness of the functions changes more smoothly during the rewiring procedure. The relationship between real-world networks and problem difficulty for evolutionary algorithms could constitute an interesting study. In fact, many optimization problems in the real world can be related with complex networks. We believe that EDAs based on graphical models can be an appropriate tool both to solve and understand this type of problems due to their underlying connection throughout the networks.

As a final note, the increase in difficulty that the problems show in the transition to small-world networks (the networks that are supposed to be related with real-world networks) could remind one of the thesis posed in Kauffman (1993). According to his assertion, the systems (and in our context the possible optimization problems associated to these systems) that emerge in nature could be in a boundary between order and chaos, a state that maximizes the complexity of the systems and simultaneously maximizes the effectiveness of the evolutionary processes taking place in the systems.

4

On the Limits of Effectiveness

4.1 Introduction

Which classes of problems a search algorithm can solve, and which classes the algorithm is not able to solve, constitute a fundamental issue that plays a key role in understanding and developing algorithms. Nowadays, there exist a vast amount of heuristic optimization techniques and new proposals are being continuously published. However, little work is usually conducted with the aim of providing a deeper understanding on where and why the different proposed algorithms are effective. In EDA research, the development of new and more sophisticated algorithms also constitutes a main topic (Brownlee et al. (2009); Lima et al. (2011); Hauschild et al. (2012); Soto et al. (2012)) although the range of problems in which they are both effective and ineffective remains practically unknown. Nevertheless, some previous works (Gao and Culberson (2005); Coffin and Smith (2007); Chen and Yu (2009)) are related to these issues. We argue that studying the limits of performance of any search algorithm is a crucial task in order to achieve a more robust and efficient spectrum of techniques.

In this chapter, we study the ability limit of EDAs to effectively solve problems in relation to the number of interactions among the variables (Echegoyen et al. (2011b)). More in particular, we numerically analyze the learning limits that different EDA implementations encounter to solve problems on a sequence of additively decomposable functions (ADFs) in which new sub-functions are progressively added. The study is carried out in a worst-case scenario where the sub-functions are defined as deceptive functions. As in previous chapters, we focus on the learning step of the algorithm. We argue that the limits for this type of algorithm are mainly imposed by the probabilistic model they rely on. To study the impact that the complexity of the probabilistic model has in the performance of the algorithm, we use three different implementations which were introduced in Chapter 1. Firstly, an EDA that assumes independences between the variables (UMDA), secondly, an EDA that learns a tree-like structure at each generation (Tree-EDA) and

finally, an EDA that learns Bayesian networks (EBNA). Since the population size is a critical parameter in EDAs and especially when Bayesian networks are learned, we use different population sizes for the experiments.

According to the results, the ability of EDAs to solve the generated problems is lost dramatically after a certain number of sub-functions in the objective function. This threshold of performance shows a marked phase-transition effect that clearly delimits the frontiers of effectiveness. The results also show that the ability to learn structures is crucial to extend the limits of successful EDA applicability. Nonetheless, EBNA shows a dramatic loss of performance due to the impossibility of the structural learning method to build more complex models. In addition, according to the results, the learning of unrestricted Bayesian networks to solve the problems rapidly entails a high computational cost. Thus, the complexity of the networks needed to solve the problems shows an exponential increase as the number of sub-functions in an ADF increases. Beyond the limitations of the approximate learning methods, the results suggest that, in general, the use of Bayesian networks can entail strong computational restrictions to overcome the limits of applicability shown in this chapter.

The remainder of the chapter is organized as follows. Section 4.2 introduces the definition of the fitness function and the procedure used to progressively increase the number of interactions. This section also analyzes some properties of the functions generated. In Section 4.3, exact factorizations for the set of functions are built and analyzed. Section 4.4 explains and discusses the results of the experiments. Section 4.5 draws the conclusions obtained during the study.

4.2 Fitness functions

In order to investigate the behavior of EDAs as the complexity of the function increases, we deal with additively decomposable functions (ADFs). This type of functions are widely used throughout the dissertation. It is well known that many optimization problems studied over the years can be modeled by using ADFs. The model of function used in this chapter, in which new subfunctions are progressively added, could be thought as a system that increases its complexity with the time due to the creation of new interactions among the variables.

4.2.1 Definitions

Although in previous chapters we have dealt with ADFs, they have not been defined in a general form. Let $S = \{0,1\}^n$ be the search space, a fitness function $f : S \to \mathbb{R}$ is additive if it can be represented as a sum of subfunctions of lower dimension,

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{c}_i \in \mathcal{C}} f_i(\boldsymbol{c}_i) \tag{4.1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ and $\mathcal{C} = \{\boldsymbol{c}_1, .., \boldsymbol{c}_l\}$ is a collection of distinct subsets $\boldsymbol{c}_i \subseteq \{x_1, \ldots, x_n\}$. In addition, we assume that no set $\boldsymbol{c}_i$ can be a subset of any other set $\boldsymbol{c}_j$ with $i \neq j$. This type of functions is also characterized by its order k, which is the size of the largest subset in $\mathcal{C}$.

In this chapter, we use specific instances of this general class of functions. Firstly, all the subsets in $\mathcal{C}$ have three variables ($k = 3$). Therefore, $\mathcal{C}$ consists of any collection of distinct sub-sets taken from all the $C(n,k) = \binom{n}{k}$ possible sub-sets of variables. Secondly, all the sub-functions f_i are the same deceptive function f_{3dec} (Goldberg (1989)) that was previously used in Chapter 2. Given $u(\boldsymbol{y}) = \sum_{i=1}^{k} y_i$ where $\boldsymbol{y} \in \{0,1\}^k$, this function can be defined as,

$$f_{3dec}(\boldsymbol{c}_i) = \begin{cases} 0.9 \; for \; u(\boldsymbol{c}_i) = 0 \\ 0.8 \; for \; u(\boldsymbol{c}_i) = 1 \\ 0.0 \; for \; u(\boldsymbol{c}_i) = 2 \\ 1.0 \; for \; u(\boldsymbol{c}_i) = 3 \end{cases}$$

From the point of view of the EDA analysis, the benefit of the fitness functions that we propose is twofold. Firstly, independently of the number of sub-functions, we always know the global optimum, which is the solution $\boldsymbol{x}$ of all ones. Secondly, the deceptive approach creates strong interdependences among the three variables belonging to each sub-function.

Regarding the f_{3dec} functions, when they are disposed without overlapping among the set of variables, we obtain the function *Deceptive*3 (Goldberg (1989)) used in Chapter 2. This specific function was proposed in the context of genetic algorithms with the aim of analyzing their limitations. Thus, in the current chapter, this function will constitute a useful reference in order to analyze the limits of performance in EDAs. Nowadays, deceptive or trap separable functions are widely used to test evolutionary algorithms.

4.2.2 Implementation of the functions

In order to progressively increase the complexity of the functions, a simple procedure is proposed. Basically, we generate a sequence of objective functions in which each new function adds one more sub-function to the previous one. This sequence of functions is given by the ordered set $\mathcal{C} = \{\boldsymbol{c}_1, .., \boldsymbol{c}_l\}$. This set is a collection of l distinct subsets of variables randomly selected according to a uniform distribution from all the $C(n,k)$ possible combinations. Although we could introduce in $\mathcal{C}$ all the $C(n,k)$ distinct subsets, it will not be necessary to reach learning limits. Thus, the s-th objective function in the sequence sums s sub-functions which are applied to the corresponding first s subsets of variables in $\mathcal{C}$. The s-th function can be expressed as,

$$f_s(\boldsymbol{x}) = \sum_{i=1}^{s \leq l} f_{3dec}(\boldsymbol{c}_i). \tag{4.2}$$

Nevertheless, the ordered set $\mathcal{C}$ has a restriction. The union of the first n/k subsets cover the whole set of n variables without overlapping, forming the function *Deceptive*3. Note that in the functions from $s = 1$ to $s = n/k$ some of the variables do not have any sub-function assigned. In order to complete these functions, we directly apply the previously defined function u over the set of variables $\{\boldsymbol{x} \setminus \bigcup_{i=1}^{s} \boldsymbol{c}_i\}$ without sub-function assignment. This stage is especially useful to analyze the univariate and bivariate EDA. Furthermore, this separable function is useful as a problem difficulty reference.

Finally, due to the random nature of the set $\mathcal{C}$, we have created for the experiments 100 different random instances of this type of sets and the results shown are the average computed from them. The whole set of experiments consists of three different problem sizes ($n \in \{24, 48, 72\}$) and the maximum number of sub-functions given by $\mathcal{C}$ is $l = 200$. In addition, for each possible function, we carry out 10 independent EDA runs. The number of runs per instance has been restricted due to the computational cost of the experiments.

4.2.3 Degree of interaction and problem difficulty

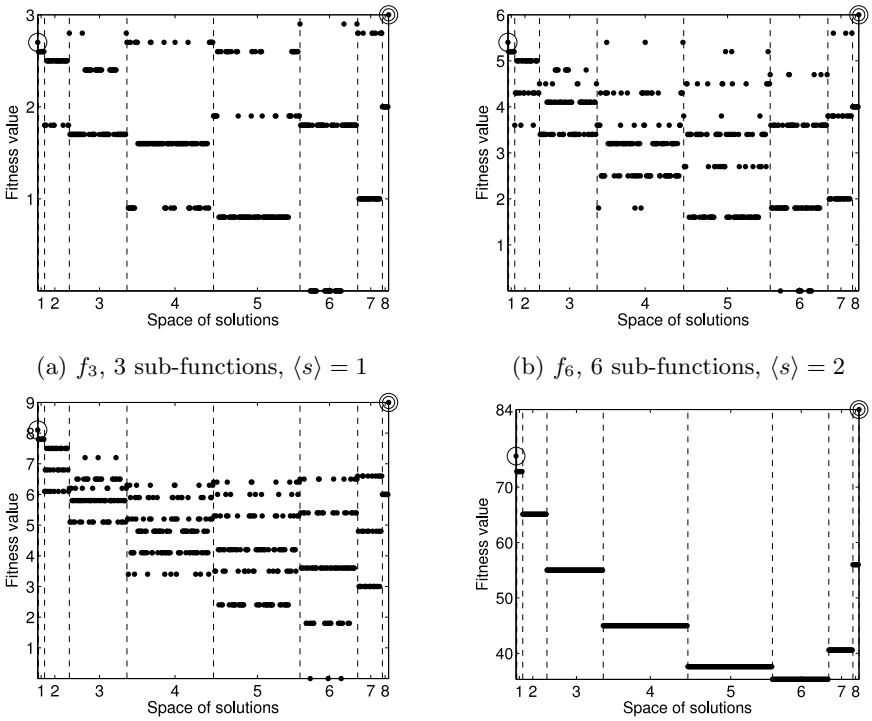
The degree of interaction can be seen as a concept related with the interdependences that arise among the variables of a problem. Although there could be many different ways to measure this notion, in the context of the present work, we assume that the degree of interaction is simply given by the number of sub-functions included in the objective function.

Additionally, in order to provide a more intuitive measure of the degree of interaction, we also take into account the number of sub-functions each variable appears in. For example, in the separable deceptive function each variable belongs to only one sub-function. In general, given s sub-functions of size k over n variables, we calculate the expected number of sub-functions assigned to each variable. It is given by,

$$\langle s \rangle = s\frac{k}{n}. \tag{4.3}$$

In order to illustrate how the landscapes of the functions change according to the number of sub-functions added, we present a simple example with $n = 9$ variables in Fig. 4.1. This figure shows the function values that four different objective functions assign to all possible solutions of the search space. The solutions are grouped by the number of ones in the x-axis with the aim of providing more intuitive plots. The vertical dashed lines enclose the groups of solutions with the same number of ones. For example, the area that corresponds to the number 4 (x-axis) contains all the $C(9, 4)$ solutions with 4 ones. Thus, the area that corresponds to the number 0 or number 9 in the x-axis only includes one solution. Both these solutions play a crucial role in

the class of functions we propose, and therefore, the assignment of all zeros is highlighted with a circle and the optimum (all ones) with two concentric circles.



(a) f_3, 3 sub-functions, $\langle s \rangle = 1$

(b) f_6, 6 sub-functions, $\langle s \rangle = 2$

(c) f_9, 9 sub-functions, $\langle s \rangle = 3$

(d) f_{84}, 84 sub-functions, $\langle s \rangle = 28$

Fig. 4.1. Different snapshots that show how the landscape of the functions with $n = 9$ changes according to the number of deceptive sub-functions. The space of solutions in the x-axis is grouped and labeled by the number of ones. The vertical dashed lines enclose the different groups of solutions that have the same number of ones. The function values of the solutions are in the y-axis. The assignment of all zeros is highlighted with a circle and the optimum (all ones) with two concentric circles. (a) Landscape of the function f_3 that adds 3 sub-functions and has $\langle s \rangle = 1$. (b) Landscape of the function f_6 that adds 6 sub-functions and has $\langle s \rangle = 2$. (c) Landscape of the function f_9 that adds 9 sub-functions and has $\langle s \rangle = 3$. (d) Landscape of the function f_{84} that adds all the 84 possible sub-functions and has $\langle s \rangle = 28$.

Fig. 4.1(a) corresponds to the landscape of a separable deceptive function and Fig. 4.1(d) corresponds to a function that sums all $C(n, k)$ possible sub-functions. In turn, Fig. 4.1(b) and 4.1(c) show the landscapes for two intermediate functions in which each variable exactly belongs to 2 and 3 sub-functions.

In general, we can observe, without using specific measures of problem difficulty for evolutionary algorithms (Naudts and Kallel (2000)), that the optimum tends to be isolated as the number of sub-functions increases. Therefore, it will be more difficult to find useful information in the populations to guide the algorithm towards the optimum. After the third snapshot (Fig. 4.1(c)), the assignments of all zeros for $\boldsymbol{x}$ is the second best solution and it seems to have a greater basin of attraction as more sub-functions are added. In addition, the neighbors of this solution, in terms of Hamming distance, tend to have higher function values than the neighbors of the optimum. On the other hand, we can see in Fig. 4.1(c) that there are solutions close (in terms of Hamming distance) to the optimum that keep high quality function values, and therefore, they can contain valuable information about the optimal solution. Even in the last snapshot (Fig. 4.1(d)), some traces of information about the optimum remain in the solutions with more than 6 ones.

4.3 Exact factorizations for the objective functions

In order to create the first relationship between the set of functions proposed in this chapter and the EDAs that use Bayesian networks, we will consider the concept of exact factorization. In the context of EDAs, this type of factorizations were first introduced in Mühlenbein et al. (1999) and employed in the factorized distribution algorithm (FDA), which was proposed by the same authors. The exact factorizations are associated to objective functions and can be defined as follows.

Definition 4.1 *Any factorization of a probability distribution $p(\boldsymbol{x})$ is said to be an exact factorization for a function $f(\boldsymbol{x})$ if i) the factorization can be represented by means of a junction tree and ii) it does not imply any conditional independence not verified by the Boltzmann distribution*

$$p(\boldsymbol{x}) = \frac{e^{f(\boldsymbol{x})}}{\sum_{\boldsymbol{y} \in S} e^{f(\boldsymbol{y})}} \tag{4.4}$$

associated to the objective function $f(\boldsymbol{x})$.

Of course, exact factorizations can be expressed in terms of Bayesian networks and a complete directed acyclic graph provides a trivial exact factorization. Nevertheless, we are interested in factorizations of minimum complexity.

As previously mentioned, we assume that the objective function $f(\boldsymbol{x})$ is an ADF as defined in Equation 4.1. Then, associated to this function, we can build an undirected graph $G_{ADF} = (V, E)$ as follows. The vertices of the graph represent the variables of the ADF. Two vertices are connected by an edge if and only if the corresponding variables are contained in the same sub-function (see Fig. 4.2(a) and 4.2(b) for an example).

It can be proved (Lauritzen (1996); Mühlenbein et al. (1999)) that any conditional independence that can be read from G_{ADF} by means of the U-separation criterion (see Castillo et al. (1997)) is verified in the Boltzmann distribution of Equation 4.4. Thus, an exact factorization in terms of marginal and conditional probability functions for $f(\boldsymbol{x})$ can be obtained by computing the *junction tree* (see Castillo et al. (1997)) of G_{ADF}. Nevertheless, we are only interested in the complexity of the exact factorizations, not in their specific factors. In particular, we consider the order of the exact factorization, which is given by the largest clique after triangulating G_{ADF} i.e. after adding a chord to every cycle of length greater than 3. The number of variables of the largest clique in the triangulated graph corresponds to the number of variables belonging to the largest factor of the exact factorization. Of course, it is desirable to find a triangulation where the size of the largest clique is minimized in order to obtain exact factorizations of minimum complexity. Unfortunately, this problem is NP-hard (Yannakakis (1981); Arnborg et al. (1987)). To carry out the triangulation step, we apply an heuristic method called *minimum size* (see Kjaerulff (1990) for details of triangulation algorithms). Fig. 4.2 shows an example with an ADF, the associated graph G_{ADF} and the corresponding triangulated graph. We can see that the largest clique of this last graph is of size 3. Therefore, the order of the exact factorization will be equal to 3.

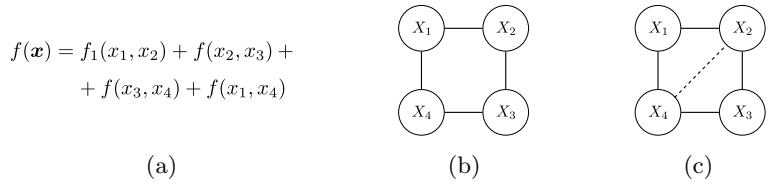


Fig. 4.2. Main elements involved in calculating the order of exact factorizations. (a) Objective function. (b) Undirected graph G_{ADF}. (c) Graph triangulated.

Roughly speaking, it could be said that exact factorizations are a sufficient condition to reach the optimum by means of EDAs based on Bayesian networks, as long as a large enough population size is used. Exact factorizations provide an upper bound for the model complexity that this type of algorithms may need to solve a problem (Gao and Culberson (2005)). Fortunately, introducing exact factorizations in the EDA is not a necessary condition to reach the optimum. In the context of this chapter, we believe that considering the complexity of this type of factorizations can be useful to better understand the limits of Bayesian network EDAs as the degree of interaction among the problem variables increases. In Gao and Culberson (2005), this type of factorizations are used to study the space complexity of EDAs in relation to the number of variables n.

In this section, for each function generated, we calculate the order of a possible associated exact factorization. As commented above, this calculation is an approximation to the exact factorization of minimum complexity. Nonetheless, this measure provides valuable information about the complexity of the probabilistic models that an EDA could need to solve the generated problems in the worst case. The orders of the exact factorizations for our sequence of functions are presented in Fig. 4.3. This figure shows the order in relation to the average number of sub-functions assigned to each variable for the three problem sizes. We start from $\langle s \rangle = 1$, that corresponds with the *Deceptive*3 function. The results imply an exponential increase of the number of parameters associated to exact factorizations and therefore, the complexity of these models quickly becomes prohibitive. Even with complete knowledge about the formulation of the function, the construction of this type of probabilistic models encounters important computational restrictions by only adding a reduced number of sub-functions. In addition, if these factorizations were used in an EDA, the population size should be increased at least proportionately to the complexity of the factorizations in order to obtain a robust learning of the parameters (Mühlenbein (2012)).

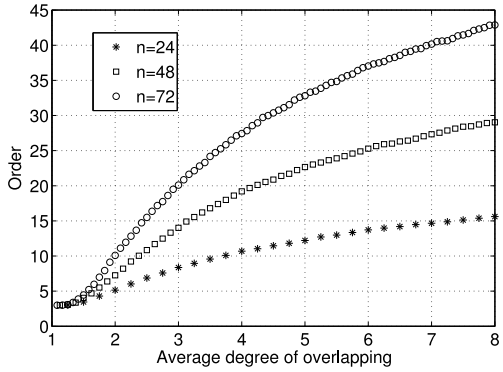


Fig. 4.3. Order of the exact factorizations built from the set of fitness functions with $n \in \{24, 48, 72\}$. The order of the factorization is depicted as a function of the average degree of overlapping $\langle s \rangle$.

4.4 Experiments

We use three EDA implementations that differ only in the structural model as explained in Chapter 1. Besides the EDA based on Bayesian networks (see Alg. 1.6 for EBNA), we consider the univariate marginal distribution algorithm (see Alg. 1.4 for UMDA) and the tree based estimation of distribution algorithm (see Alg. 1.5 for Tree-EDA).

4.4.1 Descriptors and parameters

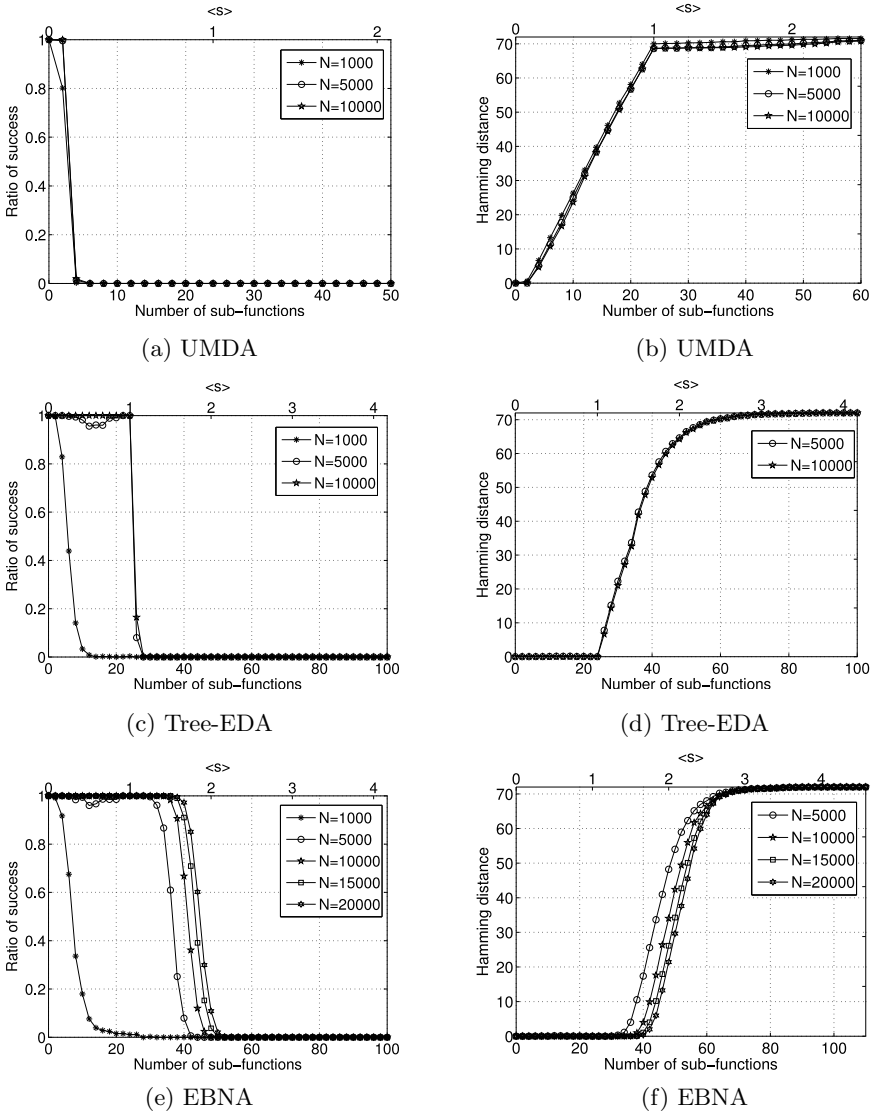
We use two descriptors in order to measure the performance and effectiveness of the aforementioned EDAs as the degree of variable interaction in the problem increases. Firstly, we calculate the ratio of successful runs. This measure represents the proportion of problems solved by the EDAs for each level of difficulty, which is given by the number of sub-functions in the objective function. Secondly, we calculate the Hamming distance between the best solution given by the algorithm and the optimum. In addition, we record the order of the Bayesian networks learned by EBNA at each generation. As introduced in Section 4.3, the order of the Bayesian network is given by the number of variables in the largest factor of the corresponding factorization, i.e. the maximum number of parents plus the child. The results that we present only take into account the maximum order among the networks learned during each run. This provides a measure of the computational cost of the search. All the results are shown both in relation to the number of sub-functions and the average number of sub-functions assigned to the variables $\langle s \rangle$.

As discussed in previous chapters, the population size is a crucial parameter of the algorithm. In this chapter, we use five different population sizes $N \in \{1000, 5000, 10000, 15000, 20000\}$. Some preliminary results, not presented here, support the fact that the population size has a higher influence on EDAs based on Bayesian networks than in simpler EDAs. Therefore, in order to avoid unnecessary experiments, the last two sizes are only used for EBNA. Finally, the stopping criterion of the algorithms is a fixed number of n generations, that is, as many as the number of variables in the problem.

4.4.2 Results

In this section, we summarize the most relevant results obtained throughout the study. Fig. 4.4 shows the ratio of successful runs and Hamming distance to the optimum for each type of algorithm. We only present the results for the problems with $n = 72$ variables. The behavior of the algorithms is similar for the three problem sizes ($n \in \{24, 48, 72\}$) that we have considered. However, as the number of variables increases, the patterns are more evident and clearer. Additional results regarding the rest of problem sizes can be found in Echegoyen et al. (2011a).

In general, through the descriptors used in Fig. 4.4, we clearly observe the manner in which the performance of different EDA implementations collapses. The curves show a phase-transition effect after a certain degree of interaction in the generated problems. Nevertheless, this effect is particularly noticeable in the curves of ratio of successful runs which fall off from 1 to 0 by only adding a few number of sub-functions. Thus, in UMDA and Tree-EDA, the difference between total success and complete failure is in 2 sub-functions. When EDA learns Bayesian networks, although its performance also suffers a sudden collapse, the transition from 1 to 0 in the ratio of successful runs



(a) UMDA (b) UMDA

(c) Tree-EDA (d) Tree-EDA

(e) EBNA (f) EBNA

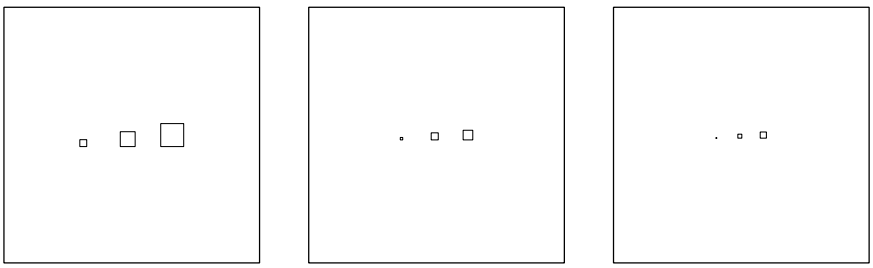
Fig. 4.4. Ratio of successful runs and Hamming distance to the optimum for the different EDA implementations with different population sizes. The x-axis at the bottom shows the number of sub-functions. The x-axis at the top shows the corresponding average number of sub-functions assigned to each variable represented by $\langle s \rangle$. (a) Ratio of success in UMDA. (b) Hamming distance in UMDA. (c) Ratio of success in Tree-EDA. (d) Hamming distance in Tree-EDA. (f) Ratio of success in EBNA. (g) Hamming distance in EBNA.

is slightly more progressive. Regarding the Hamming distance, it shows a more progressive change, which provides complementary information about the quality of the solutions. For example, although in Fig. 4.4(a) the ratio of success can be equal to 0 with only 4 sub-functions, UMDA returns, at this point, solutions which are close to the optimum (Fig. 4.4(b)).

From the results presented in Fig. 4.4, we also obtain insights into the impact that the probabilistic models and the population size have on the algorithm to solve problems with increasing degree of interaction. As expected, the probabilistic model used in the algorithm has a decisive influence on the range of problems that it is able to solve. Thus, UMDA starts to lose its reliability when the objective functions have two sub-functions, Tree-EDA is able to exactly reach the level of *Deceptive*3 function and EBNA fails between the separable function and the functions with $\langle s \rangle = 2$ (around 40 sub-functions). Moreover, according to our results, the higher the ability of an EDA to manage more complex structural models, the higher the influence of the population size is. Whereas UMDA is hardly influenced by this parameter, it is critical to obtain a robust behavior of Tree-EDA and EBNA. As shown in Fig. 4.4(c) and 4.4(e), the lowest population size ($N = 1000$) is clearly insufficient to achieve a competent performance of these algorithms. The greatest impact of the population size occurs in EBNA and it is reflected in the separation between the curves shown in Fig. 4.4(e) and 4.4(f). Nevertheless, even when EDA learns Bayesian networks, this parameter shows a limited utility to overcome a certain threshold of sub-functions. This indicates that, although the population size seems to be crucial to obtain a robust behavior, increasing this parameter is not an efficient solution to solve the problems as the degree of interaction increases. In this regard, we observe in Fig. 4.4(e) and 4.4(f) that the different curves tend to be closer as the size of the population increases.

From the curves of Hamming distance, another observation can be made. That is, after a certain degree of interaction in the problem, all EDAs return the same solution, which is in the assignment of all zeros. Then, we could say that, due to efficiency reasons, UMDA is the best option to face the problems after this critical threshold of difficulty. From this critic's point of view, and taking into account the whole range of functions that can be generated from $s = 0$ to $s = C(n, k)$, EBNA provides better results in a reduced sub-space. Note that Fig 4.4 only shows the behavior of the EDAs throughout the first levels of difficulty. With $n = 72$ and $k = 3$, we could introduce in the fitness function up to $C(72, 3) = 59640$ sub-functions. Therefore, the charts only represent a small portion of this number. In order to illustrate the proportion of the whole space of problems that the different EDAs were able to solve with reliability (they reach the optimum in 95% of the runs), we provide Fig. 4.5. In this case, the results for the three problem sizes are presented to observe the progression as n increases. The area of the biggest squares represent the whole range of problems from 0 to $C(n, k)$ sub-functions. Inside each of the three big squares, there are three small squares (from left to right they correspond to UMDA, Tree-EDA and EBNA) which represent the proportion of the whole

space of problems that each EDA was able to solve with reliability. This is an intuitive result that is useful to suggest the scope for improvement that exists for this kind of algorithms and to challenge other techniques.



(a) $n = 24$ (b) $n = 48$ (c) $n = 72$

Fig. 4.5. Intuitive representation of the proportion of the space of problems that the different EDAs have been able to reliably solve. We assume that two functions with the same number of sub-functions represent the same level of difficulty and, therefore, the same allocation in the space of problems. The area of the biggest squares represents the number of all possible levels of difficulty from 0 sub-functions to $C(n, k)$. The three squares inside the big squares (from left to right UMDA, Tree-EDA and EBNA) represent the size of the space of problems that each EDA was able to solve with a reliability of 95% for each problem size. (a) Number of variables $n = 24$ and maximum number of sub-functions $C(24, 3) = 2024$, (b) $n = 48$ and $C(48, 3) = 17296$ and (c) $n = 72$ and $C(72, 3) = 59640$.

Through the results presented in Fig 4.4, we have seen how the performance of different EDAs suddenly collapses after a certain degree of interaction among the problem variables. It seems clear that UMDA and Tree-EDA fail due to the lack of ability to learn structures. However, in the case of EBNA, it is worth conducting a more in-depth analysis of the causes of its collapse. To do that, we take into account the complexity of the Bayesian networks learned during the run. As previously mentioned, we use the order of the factorizations given by these probabilistic models to measure their complexity. We show in Fig. 4.6 the average of the maximum orders obtained during each run. The points in this chart indicate the maximum complexity that the algorithm has managed in each search.

Note that the effect of phase transition observed in Fig. 4.4(e) and 4.4(f) is completely related with Fig. 4.6. Thus, the algorithm starts to fail dramatically after the peaks in the curves shown in Fig. 4.6, i.e. when the algorithm is not able to build more complex models. It is possible to check that both events occur when the objective functions approximately have 40 sub-functions (in the case of the largest population sizes). Fig. 4.6 suggests that the complexity of the Bayesian networks that the algorithm needs to manage in order to solve the problems exponentially increases with the number of sub-functions. Note

that, when the learning method is unable to build the adequate structures to solve more problems, the algorithm still spends important computational resources on learning ineffective models.

To what extent the behavior discussed above depends on the structural learning algorithm is an issue that deserve an in-depth analysis. We hypothesize that, in this worst-case scenario, we will probably obtain a similar scene for other approximate learning techniques.

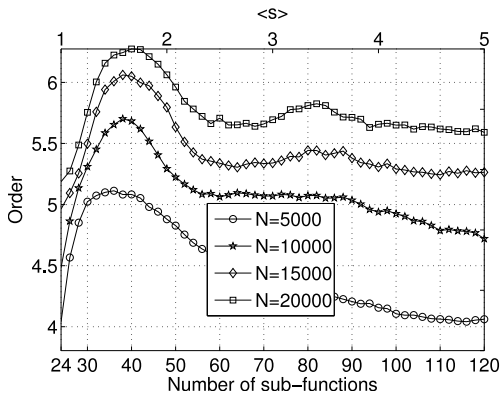


Fig. 4.6. Order of the Bayesian networks learned by EBNA. For this chart, we only take into account the Bayesian network with the maximum order during each run. The x-axis at bottom shows the number of sub-functions. The x-axis at top shows the corresponding average number of sub-functions assigned to each variable given by $\langle s \rangle$.

The complexity of the learned structures (Fig. 4.6) can be put into relation to the complexity of the exact factorizations (Fig. 4.3). This last result is shown in Fig. 4.7. The curve depicted in Fig. 4.7(a) corresponds to the order of the exact factorizations which was already shown in Fig. 4.3. Nevertheless, this time the curve is only for $n = 72$ and is presented in relation to the number of sub-functions. We have added a circular mark in order to approximately indicate the area in which the collapse of EBNA is located. In Fig 4.7(b), the order of the exact factorizations and the order of the Bayesian networks learned by the EDA are put together. The dashed line represents the curve of the exact factorizations. In this figure we can see that, when the algorithm is able to optimize the functions, the maximum complexity of the networks learned is equal to or greater than the complexity of the exact factorizations. However, when the EDA is not able to reach the complexity of the exact factorizations, it no longer solves more problems. The dashed line of the exact factorizations perfectly separates, for every population size, the problems solved from the problems unsolved.

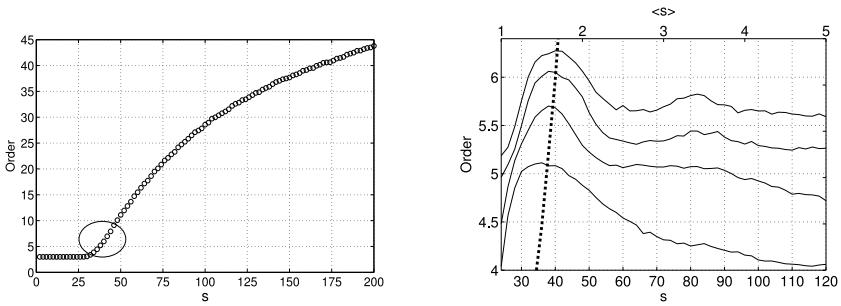


Fig. 4.7. (a) Order of the exact factorizations built from the set of fitness functions with $n = 72$. In this case, the order of the factorizations is depicted in relation to the number of sub-functions. A circular mark approximately indicates the range of functions in which the collapse of EBNA is taking place. (b) Order of the Bayesian networks learned by the EDA (Fig.4.6) along with the order of the exact factorizations in this stage (dashed line).

This result suggests that, in order to continue solving more difficult problems, the algorithm should reach the complexity of the exact factorizations. If so, this fact would imply deep computational restrictions to solve the problems by means of Bayesian networks, independently of the implementation of the learning technique. Regarding this issue, we could consider the hypothetical behavior that an EDA with exact learning could exhibit. Assuming that the population size is increased accordingly, it is possible that an exact algorithm could learn the needed probabilistic information to solve problems as the number of interactions increases. We showed in Chapter 2 that the exact learning is able to capture the structure of the problem as long as the population size is large enough. Nevertheless, in this hypothetical case, the limits of applicability would be due to efficiency reasons.

These conjectures lead us to pose a last issue regarding the population. We argue that, as the number of interactions in the function increases, a given population size $N << 2^n$ can only provide useful information to solve the problems to a certain degree of interaction, independently of the search algorithm. The relationship between the complexity of problem and the minimum population size needed to contain useful information to reach the optimum is an issue that can be taken into account for further research.

4.5 Conclusions

In this chapter, we have analyzed the limits of performance that different EDA implementations encounter as the degree of interaction among the variables of the problem increases. We base the analysis on the use of additive decomposable functions in which new sub-functions with the same deceptive values are

progressively added. Thus, the degree of interaction can be directly measured by the number of sub-functions that the objective function includes. Moreover, we use the separable deceptive function as a reference of problem difficulty in order to provide more intuitive results. In the experiments, we have dealt with three different EDA implementations. Since these algorithms only differ in the probabilistic model used, the results show the impact that introducing more complex models has in order to solve a wider range of problems. We have also used different population sizes. This parameter has been critical in order to achieve a robust behavior in EDAs based on Bayesian networks. However, the results suggest that, in general, increasing this parameter is not efficient to solve more complex problems.

We have discovered that, in the worst-case scenario, the performance of the algorithm collapses with an effect of phase transition as the number of sub-functions in the objective function increases. The threshold in which EBNA fails is between the separable deceptive functions and the objective functions with $2n/k$ sub-functions. The reason for the failure of the algorithm is in the probabilistic models. When the EDA is not able to learn more complex models to solve more difficult problems the algorithm collapses. In the first stages of the sequence of functions, the complexity of the networks learned by the algorithm tends to rise exponentially in order to reach the optimum. However, after a certain threshold, the learning is not able to build the adequate models to solve the problem and then, the algorithm fails dramatically. It suggests that, after a certain critical degree of interaction, the learning of Bayesian networks might not be able to recover the information needed to reach the optimum from the population. In addition, the relationship between the structures learned by the EDA and the exact factorization suggests strong computational limitations due to the exponential growth of the structural complexity needed to solve the problems. In order to make a step forward in solving this type of problems, the use of other classes of probabilistic models such as factor graphs (Kschischang et al. (2001); Mühlenbein (2012)) or the development of algorithms based on hybridizations could be a promising alternative (Robles et al. (2006); Zhang et al. (2005, 2007)).

The limits of effectiveness shown in this chapter are directly related to the learning step of the algorithm. However, these limits do not necessarily have only one source. We can identify three different perspectives from which the learning limits in EDAs could be discussed: i) Limitations of the learning methods either because the structure is given a priori or because they learn approximate structures of bounded complexity. ii) Even by using an ideal or exact learning algorithm, there could exist efficiency limits due to an exponential rise of the structural complexity needed to solve the problems as the number of interactions increases. iii) Limits due to the population either because of the lack of information that it contains to solve the problem or because this parameter should be exponentially increased to provide a robust learning (Mühlenbein (2012)).

In summary, we have explored the concept of boundaries of EDA effectiveness in relation to the degree of interaction of the problem. The final objective of this research trend is not to find the overall best algorithm or discredit any technique but to better understand which algorithms are the best for which problems.

5

Analyzing the Probability of the Optimum and the Most Probable Solution

5.1 Introduction

Outside the extreme experimental framework specifically designed in the previous chapter in order to reach and study the limits of performance in EDAs, this type of algorithms has been successfully applied to solve a wide variety of challenging optimization problems. In fact, they are nowadays a strong alternative for solving problems from different domains such as engineering (Simionescu et al. (2007); Yu et al. (2006)), biomedical informatics (Armañanzas et al. (2008); Santana et al. (2008a); Armañanzas et al. (2011)) or robotics (Yuan et al. (2007)). Nevertheless, despite their promising applicability, there is a wide variety of open questions (Santana et al. (2009b)) regarding the behavior of this type of algorithms.

Both in previous chapters and other works devoted to the study of the behavior of EDAs, the structural component of the probabilistic models used by the algorithm has been the main source of information (Bengoetxea (2003); Mühlenbein and Höns (2006); Lima et al. (2007); Hauschild and Pelikan (2008); Brownlee et al. (2008); Echegoyen et al. (2008); Hauschild et al. (2009)). In this chapter, we propose a novel methodology based on a quantitative analysis of the probabilistic models. Ultimately, the particular probability values assigned to the solutions during the search are the raw material from which EDAs obtain their results. Therefore, studying such probabilities should provide useful information to better understand the behavior of this type of algorithm. Following this criterion, our quantitative analysis of EDAs is based on monitoring the probability of certain distinguished solutions during the search: i) the optimal solution of the function, ii) the solution with the highest probability in the distribution and iii) the best individual in each generation. In order to complete the quantitative analysis, we also record the fitness function values for the solutions ii) and iii) during the search.

The proposed analysis is carried out when EDAs that use Bayesian networks are applied to classic benchmark problems such as separable trap functions, 2D Ising spin glass and maximum satisfiability. The EDA can use dif-

ferent structural models which can be learned from the population or created by reproducing interactions among the variables of the problem. We also use different population sizes in order to analyze the influence of this parameter in the algorithm. Furthermore, we take into account both successful (the optimum is reached) and unsuccessful runs (the optimum is not reached).

Throughout this study we shed light on basic questions of great interest that still remain open in EDAs such as:

- How does the probability assigned by the probability distributions to the optimal solution evolve during the search?

This first question plays a key role in this work and it is related with a number of current assumptions in the application of EDAs. For example, whether, in order to solve a problem, it is a necessary condition that the probability associated by the algorithm to the optimal solution steadily increases at each generation or whether the highest probability is assigned to the optimum during the search.

- How does the accuracy of the information about the problem contained in the structural model influence the internal behavior of EDAs?

This question is addressed in order to better know the relation between the interactions of the problem and the dependences of the probabilistic model. By using different structures in EDAs, we study the effect that introducing more interactions in the structural model has on the behavior of EDAs in general and in the previously mentioned assumptions about the probability of the optimum in particular.

Some works (Baluja (2006); Hauschild et al. (2009, 2012)) have considered different means of introducing a priori knowledge of the problem into the algorithm in order to improve the efficiency and efficacy of EDAs. Understanding the impact of this type of practices in the internal behavior of EDAs is also a very important issue in their application to real problems.

- How does the function value for the most probable solution given by the probabilistic models evolve during the search?

A contribution of this work is the exact calculation and analysis of the solution with the highest probability in the distributions estimated at each generation. Thus, by using its fitness function value, we can study how the probabilistic model captures the properties of the function. It would be desirable that the function value of the solution with the highest probability increased during the search.

Although the main target of this work is to provide insights about these key issues, the results obtained show a different perspective of EDAs that is able to reveal constant patterns in their behavior. Furthermore, both the probability and function values analyzed are able to capture the quality of the probabilistic models in terms of their use within EDAs. Throughout the

analysis proposed, it is also possible to better understand how the convergence of the algorithm occurs and even detect multimodality in the problems solved.

The rest of the chapter is organized as follows. Section 5.2 introduces abductive inference in Bayesian networks. Section 5.3 explains the experimental design. Sections 5.4, 5.5, 5.6 and 5.7 discuss Trap5, Gaussian Ising, $\pm J$ Ising and Max-SAT problems respectively, quantitatively analyzing the behavior of EDAs for each problem when different structural models and population sizes are used. Section 5.8 discusses relevant previous works. Finally, Section 5.9 draws the conclusions obtained during the study.

5.2 Abductive inference in Bayesian networks

In general, abductive reasoning tries to find the hypothesis that would best explain a set of facts or observations. In the probabilistic network context, the abductive inference (Gámez (2004)) consists of finding the maximum a posteriori probability state of the network variables, given some evidence (observed variables).

The total abductive inference involves all the problem variables and is defined as follows. Given a probability distribution over the vector of random variables $\boldsymbol{X}$ and the evidence $\boldsymbol{e}$, that is an instance of the observed variable set $\boldsymbol{E} \subseteq \{X_1, \ldots, X_n\}$, we want to obtain the assignment $\boldsymbol{x}^*_{\boldsymbol{U}}$ to the unobserved variables $\boldsymbol{X_U} = \{X_1, \ldots, X_n\} \setminus \boldsymbol{E}$ such that,

$$\boldsymbol{x}^*_{\boldsymbol{U}} = \arg\max_{\boldsymbol{x_U}} p(\boldsymbol{x_U}|\boldsymbol{e}). \tag{5.1}$$

Usually $\boldsymbol{x}^*_{\boldsymbol{U}}$ is known as the *most probable explanation*.

However, when this technique is applied to the probability distributions associated to Bayesian networks in EDAs, there is no evidence. In this case, the objective is to look for the assignment $\boldsymbol{x}^*$ with the highest probability for the whole vector of variables $\boldsymbol{X}$. Knowing that $P(\boldsymbol{X_U}|\boldsymbol{E}) = P(\boldsymbol{X}|\boldsymbol{E})$ and having an empty evidence set $\boldsymbol{E} = \emptyset$, Equation 5.1 can be directly converted into our target,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}, \hat{S}) \tag{5.2}$$

where $\hat{S}$ is the structure of the model which has been learned from the population and $\hat{\boldsymbol{\theta}}$ represents the parameters of the probabilistic model estimated by maximum likelihood. In our context of EDAs, $\boldsymbol{x}^*$ is called the *most probable solution* (MPS). As it is proven in Shimony (1994), this kind of inference is an NP-hard problem. Therefore, its exact resolution is only feasible in problems of limited length.

In this work, the point with the highest probability is calculated using probability propagation in junction trees (Castillo et al. (1997)) or variable elimination techniques (Dechter (1999)), as they are implemented in Bayes Net Toolbox (Murphy (2001)).

5.3 Experimental design

The experiments were mainly designed with the aim of shedding light on the questions and assumptions mentioned in Section 5.1. Specifically, at each generation of the EDA, we record the probability and fitness values of distinguished solutions of the search space: the optimum, the most probable solution and the best individual in the population. By varying the problem size, using different structural modes and different population sizes, different scenarios are created to complete the analysis.

In order to show the relation between the probabilities of our distinguished solutions and the diversity of the population, we hereby introduce additional information that is not obtained from the probabilistic model but directly from the population itself. Particularly, at each step of the algorithm, we calculate the accumulated entropy of the population by means of adding the entropy of each variable belonging to the function,

$$H(\boldsymbol{X}) = -\sum_{i=1}^{n}\sum_{j=1}^{r_i} p(x_i^j)\cdot \log_2 p(x_i^j). \tag{5.3}$$

This metric shows how the population managed by the algorithm loses diversity and converges. Some works have already studied these types of measurements in EDAs (Ochoa and Soto (2006); Ocenasek (2006)).

In the following section, we will explain the different problems, structural models and parameters used for the experiments. In Santana et al. (2010), the necessary tools to reproduce the experiments or to carry out similar analysis are implemented.

5.3.1 Problems

The whole set of problems is based on additively decomposable functions. A general difintion of this type of functions was presented in Section 4.2.1 of the previous chapter.

With the aim of covering a wide spectrum of applications and observe the behavior of EDAs in different scenarios, we chose the following four test problems: Trap5, Gaussian Ising, $\pm J$ Ising and Max-SAT. The exact details of each problem will be introduced in the following sections. These problems are selected for several reasons. Firstly, they have different numbers of optimal solutions in order to investigate the influence of multimodality in the behavior of EDAs. The first two problems have just one optimal solution and the last

two problems have several optimal solutions. Secondly, all of them are optimization problems which have been widely used to analyze EDAs (Hauschild et al. (2009); Pelikan and Goldberg (2003); Brownlee et al. (2007)). Finally, all the problems have a different nature. Trap5 (Deb and Goldberg (1994)) is a deceptive function designed in the context of genetic algorithms (Goldberg (1989)) aimed at finding their limitations. It is a separable function and in practice it can be easily optimized if the structure is known. Gaussian Ising and $\pm J$ Ising come from statistical physics domains and are instances of the Ising model proposed to analyze ferromagnetism (Ising (1925)). The variables are disposed on a grid and the interactions do not allow to decompose the problem into independent subproblems of bounded order (Mühlenbein et al. (1999)). It is a challenge in optimization (Hauschild et al. (2009); Pelikan and Goldberg (2003)) and in its general form is NP-complete (Barahona (1982)). Max-SAT is a variation for optimization of a classic benchmark problem in computational complexity, the propositional satisfiability or SAT. In fact, SAT was the first problem proven to be NP-complete (Cook (1971)) in its general form. An instance of this problem can contain a very high number of interactions among variables and in general, it can not be efficiently divided into subproblems of bounded size in order to reach the optimum. With the exception of the function Trap5, we have dealt with 100 instances for each type of problem.

Regarding the information stored at each generation, when the functions with just one optimum are optimized, we only record our three distinguished solutions. However, in the functions with several optimal solutions, the analyzed EDAs reach a subset of those optima and the analysis of the probability of the optimum is extended. Thus, we calculate the probabilities during the search for all optima reached by the alorithm in the last generation. It leads us to see how the probability is distributed when there are different optimal solutions. In order to gain clarity in the results, we only show the maximum and minimum probabilities assigned by the probability distribution to the reached optimal solutions at each generation.

5.3.2 Structural models

In the literature, several works have discussed the influence of the structure of the probabilistic model in the behavior of EDAs (Echegoyen et al. (2007, 2008)) and the impact of using a priori knowledge of the problem in the search (Hauschild et al. (2009, 2012)). In this chapter, we also take into account these important issues. Therefore, in addition to deal with automatic learning techniques, we propose to include some manageable structural models related with the problem structure to analyze the changes produced in the internal behavior of EDAs. We do not consider the use of exact factorizations because they are not relevant for the purposes of this chapter. Moreover, it has been shown (Pelikan (2005); Hauschild et al. (2009)) that the structures learned by EDAs to solve the Ising problem are far from the complexity of the exact fac-

torization. And for MaxSAT, this type of factorizations are computationally intractable.

Specifically, based on the EBNA implementation, we use the following two approaches as regards the structural models. On the one hand, we use Algorithm B to obtain a new structure from the selected individuals at each generation. On the other, we use two fixed structures related to the nature of the function, and thus, only parametric learning is carried out. Since all the functions are ADFs, an intuitive and straightforward way to create a related structural model is by means of linking variables belonging to the same subfunction with arcs. The first structure tries to reproduce all interactions among variables that can be directly observed from the formulation of the problem. As a general method, we connect two variables (representing nodes in the graph) by an edge in the structure if the corresponding variables are contained in the same sub-function. Then, by taking a directed acyclic version of this undirected graph, we obtain a Bayesian network structure which will be called *dense structure*. The second structure also reproduces interactions among the variables of the function but only considers bivariate dependences. This structure has less information but is always related with the nature of the problem. It will be called *bivariate structure*.

It must be pointed out that our aim is to study the influence that different structural models have over the probability values rather than demonstrating their quality and accuracy.

5.3.3 Parameter configuration

The sample size is very important in order to learn Bayesian networks (Friedman and Yakhini (1996)) and, hence, in the behavior of EDAs based on this type of models. This fact has been proved in previous chapters. Thus, we deal with two different population sizes in order to analyze their influence in the algorithm. Firstly, we use the bisection method (Pelikan (2005)) to determine an adequate population size to reach the optimum (with high probability). This size is denoted by m. The details of the bisection method were introduced in Chapter 3. The stopping criterion for bisection is to obtain the optimum in 10 out of 10 independent runs. The final population size is the average over 20 successful bisection runs. Due to computational restrictions, the maximum population size has been limited to 2^{14}. The population size m is always obtained from EBNA executions with Algorithm B. The second population size is half of the bisection, $m/2$. With this size we try to create a more realistic scenario in which achieving the optimum is less likely. This also allows us to analyze in detail the probability of the optimum when it is not reached.

In addition to population size, different problem dimensions have also been taken into account. Particularly, we have used $n = \{50, 75, 100\}$ for Trap5 and Max-SAT, and $n = \{8 \times 8, 9 \times 9, 10 \times 10\}$ for both types of Ising. The upper bound has been set to 100 variables due to the high memory requirements

needed to calculate the most probable solution. Increasing the number of variables would require the use of approximate inference techniques (Kschischang et al. (2001)), spoiling the correctness of the results.

The stopping criterion is a fixed number of iterations and it is independent from obtaining the optimum. Each execution will run n generations, that is, as many as the number of problem variables. This number of generations is enough to observe the convergence of the analyzed probability values.

5.3.4 Details of the experiments

The analyzed probability values are reported in logarithmic scale in order to smoothen the probability slopes and better observe their behavior from the beginning of the run. The number of runs which have reached the optimum at each generation is indicated with bars on the top of the charts, where the probability values are shown. Although we have made runs with a fixed number of generations, the charts presented were cut when all runs have reached the optimum or the curves are stabilized.

Concerning the total number of executions, for each Bayesian network learning approach and population size, we carried out 50 independent runs for Trap5 and 5 independent runs for each of the 100 instances in the rest of the problems. All the runs belonging to 100 different instances of a given problem are put together and analyzed as a whole in order to provide a general view of the behavior of EDAs.

Analyzing the wide set of results collected throughout the experiments, we have observed that EDAs show the same patterns of behavior independently of the problem dimension. Therefore, we will focus on problem sizes of 100 variables. For the sake of simplicity, in this chapter we only present the most relevant results. However, for the interested reader, the complete analysis is available on the website of the Intelligent Systems Group[1].

5.4 EDA behavior solving Trap5

5.4.1 Trap5 description

Our first function, Trap5 (Deb and Goldberg (1994)), is an additively separable (non overlapping) function with a unique optimum. It divides the vector $\boldsymbol{x}$ of n variables, into disjoint subvector $\boldsymbol{x}_I = (x_{5I-4}, x_{5I-3}, x_{5I-2}, x_{5I-1}, x_{5I})$ of 5 consecutive variables. As previously done, we use the function $u(\boldsymbol{y}) = \sum_{i=1}^{k} y_i$ where $\boldsymbol{y} \in \{0,1\}^k$ to define the function Trap5 as,

$$\text{Trap5}(\boldsymbol{x}) = \sum_{I=1}^{\frac{n}{5}} trap_5(\boldsymbol{x}_I) \tag{5.4}$$

[1] http://www.sc.ehu.es/ccwbayes/members/carlos/eda_probs/

where $trap_5$ is defined as,

$$trap_5(\boldsymbol{x}_I) = \begin{cases} 5 & \text{if } u(\boldsymbol{x}_I) = 5 \\ 4 - u(\boldsymbol{x}_I) & \text{otherwise} \end{cases} \tag{5.5}$$

This function has one global optimum in the assignment of all ones for $\boldsymbol{x}$ and a large number of local optima, $2^{n/5} - 1$.

Trap5 function has been used in previous works (Hauschild et al. (2009)) to study the structure of the probabilistic models in EDAs based on Bayesian networks, as well as studying the influence of different parameters (Lima et al. (2007)). It is important to note that this function is difficult to optimize if the probabilistic model is not able to identify interactions between variables (Echegoyen et al. (2009)).

5.4.2 Structures related to the problem

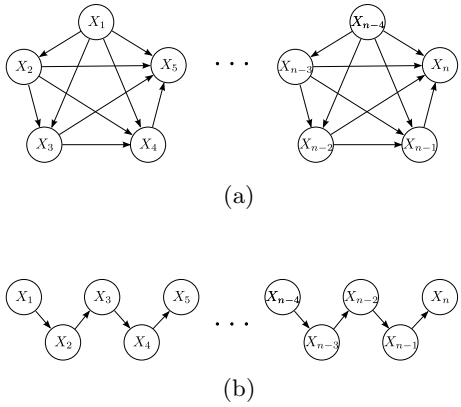


Fig. 5.1. Fixed structural models related with the dependences among the variables in Trap5. (a) Dense structure. (b) Bivariate structure.

In this section we propose two fixed structures related with the Trap5 function. The dense structure is created by linking all the variables in each sub-function $trap_5$. Thus, by providing direction to the arcs without creating cycles, we obtain the Bayesian network structure shown in Fig. 5.1(a). With this structure, there are no independences between variables of the same subgroup and variables from different partitions are independent. Exceptionally, for this type of separable functions, this intuitive method to introduce information of the problem into the structural model provides exact factorizations. However, this is not the case for the rest of the problems.

As regards the bivariate structure, it is formed by a chain for each subgroup of 5 variables. As can be seen in Fig. 5.1(b), the graph contains the

minimum number of arcs necessary to connect all the variables belonging to each partition.

5.4.3 Using structural learning

In this section, we present and discuss the results obtained when EBNA, using Algorithm B, tries to optimize the Trap5 function.

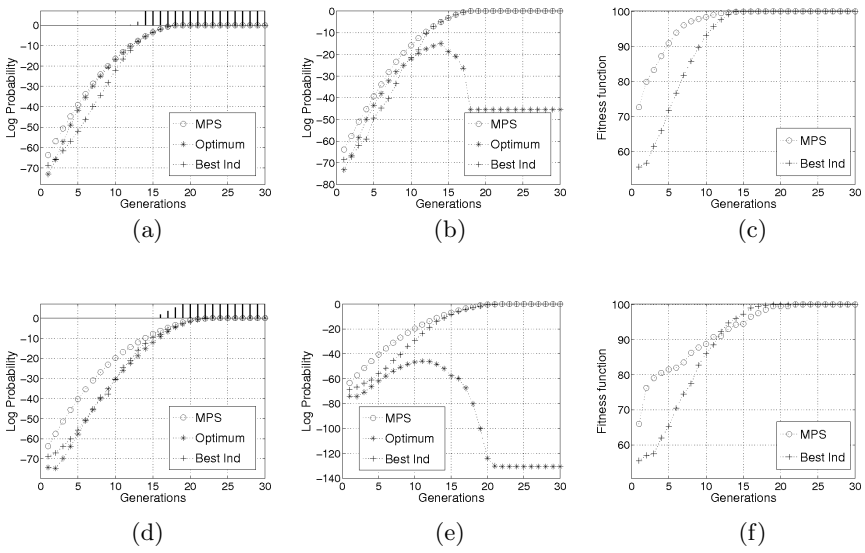


Fig. 5.2. Probability values and function values when EBNA is applied to Trap5 using Algorithm B. We have 49 out of 50 successful runs with population size m and 4 out of 50 with population size $m/2$. (a) Successful runs with population size m. (b) Unsuccessful runs with population size m. (c) Successful runs with population size m. (d) Successful runs with population size $m/2$. (e) Unsuccessful runs with population size $m/2$.(f) Successful runs with population size $m/2$.

In Fig. 5.2(a) we show the probability values for successful runs when the population size m (given by bisection) is used. In this case, EBNA reaches the optimum in 49 out of 50 runs. Theoretically, a convergence of the algorithm to the global optimum implies an increase in its probability value as the generations advance. This fact is reflected in the results. The probability values for the optimum and the most probable solution (MPS) grow simultaneously, and very closely, when the executions are successful.

When a population size of $m/2$ (Fig. 5.2(d)) is used, the results change drastically and only 4 runs reach the optimum. Although the behavior of the probability values in this case is analogous to Fig. 5.2(a), we can observe that the probability curves for the MPS and the optimum with population size $m/2$

are clearly more distant than with size m. This analysis reflects the crucial role of the population size in the aglorithm. Another important observation is that the growth of the probability values is slower for $m/2$. Thus, with this population size, the optimum is reached for the first time, a few generations later than with size m (bars on the top of Fig. 5.2(a) and 5.2(d)). Nevertheless, with both population sizes, the executions starts to reach the optimum when its probability is approximately -10 in logarithmic scale.

The results of the probability values for the executions where the optimum was not reached are shown in Fig. 5.2(b) and 5.2(e). In these figures, we can see the joint growth of the probability values for the MPS and the optimum at the beginning of the run. However, after a certain generation, both values start to diverge and the optimum is no longer obtained. In unsuccessful runs, there are also differences between the runs with population size m and $m/2$. For this last population size, the probability of the optimum reaches lower values both before decreasing and in the last generations. Even at the beginning of the run, the probability of the optimum is further from the highest probability in the distribution with population size $m/2$ than with m.

Concerning the fitness function value, Fig. 5.2(c) shows the results when the population size m, given by bisection, is used in EBNA. The value of the MPS increases at each generation and it is better than the best individual in almost all generations. However, by looking at Fig. 5.2(f), it can be seen that the MPS has a lower growth with population size $m/2$. Moreover, the best individual of the population is better than the MPS after generation 12. Therefore, the analysis of the function values also reflects the impact of the population size in the algorithm. In all the experiments, the curves of function values are similar in successful and unsuccessful runs.

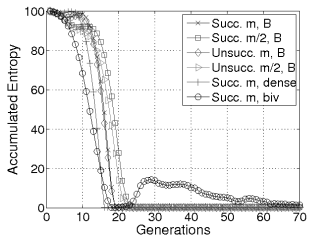
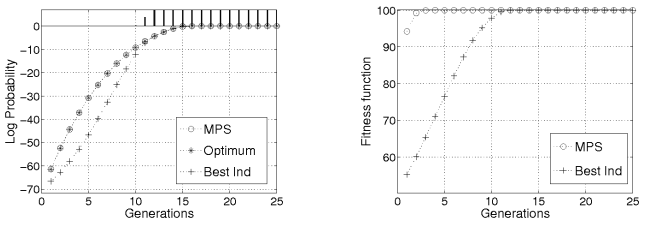


Fig. 5.3. Accumulated entropies of the population when EBNA is applied to Trap5.

Finally, in Fig. 5.3 we present the accumulated entropies of the population during the search. Only the first four curves shown in the legend correspond to EBNA with Algorithm B. These curves show the close relation of this measure with the exponential growth of the probability values. Moreover, we can observe how the population converges to a unique solution since the entropy tends to 0.

5.4.4 Using fixed structures

In this section, we show the behavior of the algorithm when a different amount of information is introduced in the structural model. As previously mentioned, throughout the chapter we only show and comment on those results that provide relevant information, avoiding excessive and redundant information. So, when the structures are fixed, we have observed a low influence of the population size in the results. In these cases, we only show the analysis for the size m.



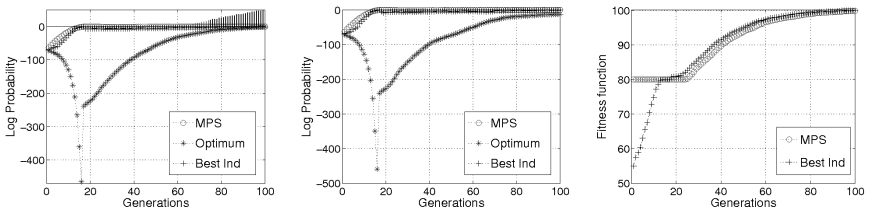
(a) (b)

Fig. 5.4. Successful runs when EBNA is applied to Trap5 using the dense structure with population size m. The optimum is reached for the 50 executions.

Fig. 5.4 shows the probability and fitness values when the dense structure is introduced in EBNA. In this case, we obtain an ideal behavior for an optimization process since the optimum has the highest probability during the whole run and is reached in all executions. Furthermore, in Fig. 5.4(b) we observe that the function values for MPS are close to the optimum from the very beginning of the search. In this case, through a sampling based on inference, the optimum could be reached in the first generations. We should remember that in this case the algorithm is using an exact factorization.

The behavior of the algorithm changes drastically when the bivariate structure (Fig. 5.1(b)) is introduced. Although the EDA shows a good performance because it reaches the optimum in 42 out of 50 runs, the evolution of the probability values (see Fig. 5.5(a)) is particular. The probability of the optimum decreases at the beginning of the run and when the algorithm seems to converge to a local optimum, it suddenly recovers. This fact also occurs for the unsuccessful executions (Fig. 5.5(b)) where the probability of the optimum increases in the last generations. In this case, the optimum has a high probability at the end of the run, which supports the belief that the algorithm would be able to reach the optimum provided that more generations are allowed.

The reason for such an uncommon behavior is the following: In the first part of the run, when the probability of the optimum decreases, the algorithm is deceived by the function and most of the individuals in the population become the local optimum. This local optimum is the assignment of zeros for



(a) (b) (c)

Fig. 5.5. Probability values and function values when EBNA is applied to Trap5 using the bivariate structure with population size m. We have 42 out of 50 successful executions. (a) Successful runs. (b) Unsuccessful runs. (c) Successful runs.

$\boldsymbol{x}$ because it is the suboptimal value that $trap_5$ function gives to each trap partition $\boldsymbol{x}_I$. Fig. 5.3 shows how the curve of entropy for the bivariate model (Succ. m, biv) tends to 0 when the probability of the optimum is minimum in Fig. 5.5(a) and 5.5(b). After this stage, the algorithm recovers and the probability of the optimum begins to increase. The curve of entropy for the bivariate model indicates that different individuals are included in the population just when the algorithm seems to converge to the local optimum. It shows that the algorithm samples better individuals and is reflected in the fitness function values in Fig. 5.5(c). This can be explained through the Laplace correction and the fixed structure of chain subgraphs. This quantitative analysis justifies why it is possible to reach the optimum for this function with a simple bivariate structure.

While several works have analyzed EDA behavior using entropy measures (Ocenasek (2006); Ochoa and Soto (2006); Wright et al. (2004)), the joint analysis of the relationship between the type of model structure, the probability values and the entropy should support a more complete perspective about the EDA dynamics. For instance, the phenomenon we have described in Fig. 5.3, where the entropy initially tends to zero but later recovers, indicates that stopping criteria based on the entropy, e.g. Ocenasek (2006), should take this type of behavior into account to avoid early termination of the EDA.

5.5 EDA behavior solving Gaussian Ising

5.5.1 2D Ising spin glass description

Ising spin glass is an optimization problem which has been used and analyzed in different works related with EDAs (Mühlenbein and Höns (2005); Shakya (2006); Hauschild et al. (2009)). A classic 2D Ising spin glass can be simply formulated. The set of variables $\boldsymbol{x}$ is seen as a set of n spins disposed on a regular 2D grid L with $n = l \times l$ sites and periodic boundaries (see Fig. 5.6). Each node of L corresponds to a spin x_i and each edge (i, j) corresponds to

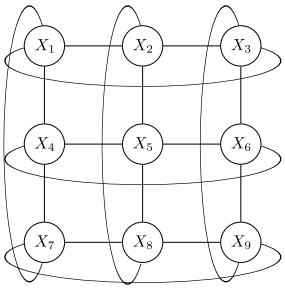


Fig. 5.6. A 3×3 grid structure L showing the interactions between spins for a 2D Ising spin glass with periodic boundaries. Each edge has an associated strength J_{ij}.

a coupling between x_i and x_j. Thus, each spin variable interacts with its four nearest neighbors in the toroidal structure L. Moreover, each edge of L has an associated coupling strength J_{ij} between the related spins.

The target is, given couplings J_{ij}, to find the spin configuration that minimizes the energy of the system computed as,

$$E(\boldsymbol{x}) = -\sum_{(i,j)\in L} x_i J_{ij} x_j - \sum_{i\in L} h_i x_i \tag{5.6}$$

where the sum runs over all coupled spins. In our experiments we take $h_i = 0$ $\forall i \in L$. The states with minimum energy are called *ground states*.

Depending on the range chosen for the couplings J_{ij} we have different versions of the problem. For the *Gaussian Ising* problem, the couplings J_{ij} are real numbers generated following a Gaussian distribution. A specified set J_{ij} of coupling defines a spin glass instance. We generated 100 Gaussian Ising instances using the Spin Glass Ground State server[2]. The minimum energy of the system is also provided by this server.

5.5.2 Structures related to the problem

In order to create a dense structure for this problem, we reproduce the undirected graph L in the model, which represents all the interactions among the variables in the function, and direct the edges without creating cycles to obtain a Bayesian network. Starting from the first spin x_1 we give a westward and southward direction to the edges as can be seen in Fig. 5.7 (a).

We are aware that for this problem, the directions given to the arcs could modify the behavior of EBNA, due to the different independence relations introduced in the Bayesian network. However, as previously mentioned, the information introduced in the structural model is directly obtained from the

[2] http://www.informatik.uni-koeln.de/ls_juenger/index.html

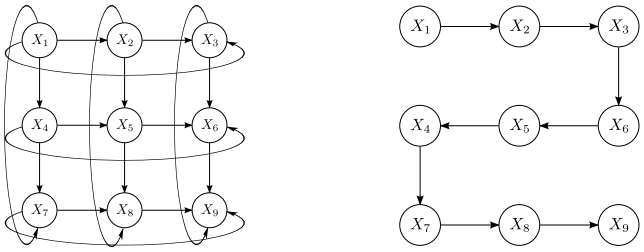


Fig. 5.7. Fixed structural models for 2D Ising spin glass. (a) Dense structure. (b) Bivariate structure.

formulation of the problem. In this sense, the direction of the arcs do not follow a specific criterion.

The second structure is a simple model which connects all variables using a chain. This structure introduces very few interactions related with the problem, as can be seen in Fig. 5.7 (b).

5.5.3 Using structural learning

The evolution of the probability values in different situations is presented in Fig. 5.8. First of all, we note that when the population size m given by bisection is used, EBNA reaches the optimum in 470 runs out of 500, while with population size $m/2$ it decreases to 272. The proportion of successful runs with $m/2$ indicates that, in order to reach the optimum, the population size is less decisive in Gaussian Ising than in Trap5 (4 out of 50 successful runs with $m/2$).

Fig. 5.8(a) and 5.8(c) show the probability values for successful runs with population size m and $m/2$ respectively. We can observe that the probabilistic behavior follows the same patterns as that in Trap5: i) the probability of the optimum increases during the search, being the most probable solution at the end of the run, ii) for the population size given by bisection, the curves for the MPS and the optimum are closer than for $m/2$. Nonetheless, it can be seen that the population size had a larger impact in Trap5 (Fig. 5.2(a) and 5.2(c)). In that problem, the difference between the probability of the optimum and the highest probability in the distribution had a more emphasized change when the population size was varied (it was reflected in the number of successful runs). Moreover, while in Trap5 the probability of the optimum was very close to the highest probability from the first generations with population size m, in Gaussian Ising both probability curves keep a visible distance throughout the generations. Lastly, for Gaussian Ising, the runs do not reach the optimum (bars on the top of the charts in Fig. 5.8(a) and 5.8(c)) until their probability value exceeds approximately the threshold of -20 in logarithmic scale. We note that for Trap5, this threshold was much higher (-10). These particular

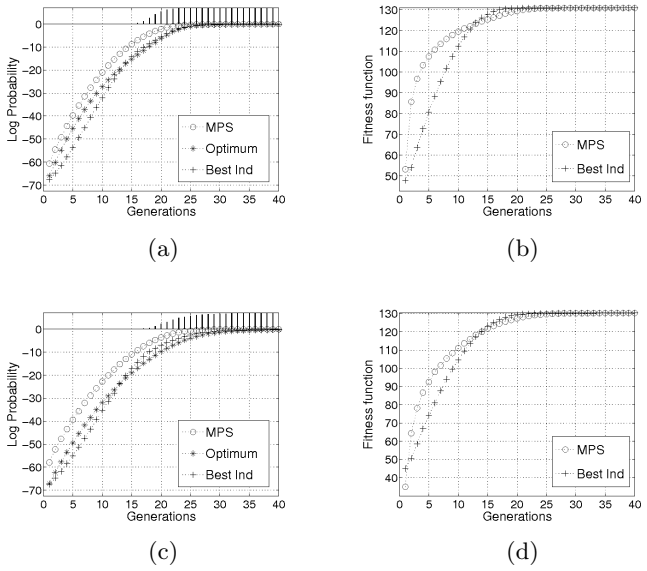


Fig. 5.8. Successful runs when EBNA is applied to Gaussian Ising using Algorithm B. We have 470 out of 500 successful runs with population size m and 272 out of 500 with population size $m/2$. (a) Population size m. (b) Population size m. (c) Population size $m/2$. (d) Population size $m/2$.

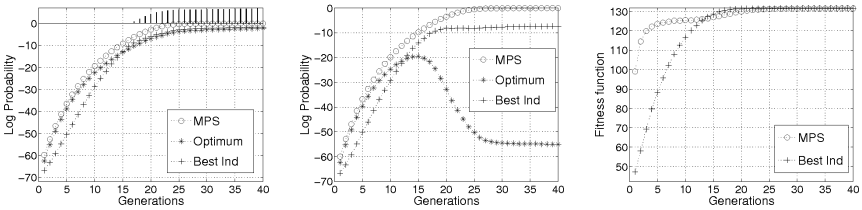
differences in the probability values between both problems could be due to the characteristics of the respective landscapes.

The analysis of the function values shown in Fig. 5.8(b) and 5.8(d) supports the previous discussion. As in Trap5, the difference in function value between the MPS and the best individual was bigger with population size m than with $m/2$. However, in this case, since the population size is less influential, the difference between curves is less marked than in the previous problem.

5.5.4 Using fixed structures

As in the previous problem, in this section we present the results for the population size m given by bisection. First of all, we note that the dense structure does not always reach the optimum as in Trap5. In particular, we achieved 283 out of 500 successful runs. Although the behavior of this structure does not outperform the behavior of EBNA with Algorithm B, its introduction in the algorithm has considerable consequences.

In Fig. 5.9(a) we report results of the analysis of the probability values and function values. In this case, the probability of the optimum is not the highest probability in the distribution during the search as in Trap5 (Fig. 5.4(a)). However, the distance between both probability curves is smaller than with



(a) (b) (c)

Fig. 5.9. Probability values and function values when EBNA is applied to Gaussian Ising using the dense structure with population size m. We have 283 out of 500 successful executions. (a) Successful runs. (b) Unsuccessful runs. (c) Successful runs.

structural learning except for the last generations. This behavior is probably influenced by the criterion for directing the arcs. Depending on the instance, one selected direction could have important effects in the probability of the optimum. An example of this can be seen in Fig. 5.10.

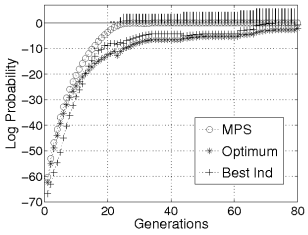
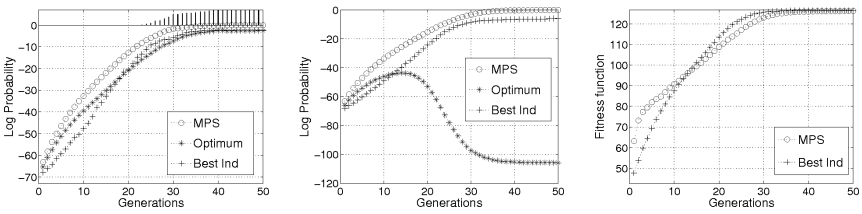


Fig. 5.10. Logarithm of the probabilities in successful runs when EBNA is applied to a particular instance of Gaussian Ising using the dense structure. For this instance, we have 10 out of 50 successful runs.

In Fig. 5.9(c) we report the probability values when EBNA does not reach the optimum. We observe that the probability of the optimum is close to the highest probability in the distribution during the first generations and reaches values up to -20 before decreasing. In general, when the dense structure is introduced in the algorithm, the probability of the optimum in unsuccessful runs has higher values during the search than in previous scenarios.

Regarding the function values, as can be seen in Fig. 5.9(c), the MPS is close to the optimum from the beginning as in the case of Trap5. Moreover, this behavior remains constant in all of the instances analyzed. Again, this fact shows that using structures related with the interactions of the problem presents promising properties that deserve a specific study.

When we reduce the amount of information in the structural model, the effects in the algorithm are not only seen in the number of successful runs but



(a) (b) (c)

Fig. 5.11. Probability values and function values when EBNA is applied to Gaussian Ising using the bivariate structure with population size m. We have 29 out of 500 successful executions. (a) Successful runs. (b) Unsuccessful runs. (c) Successful runs.

also in the analysis. In this problem, when the bivariate structure of the chain is introduced, we only obtain 29 out of 500 successful runs. Looking at Fig. 5.11 we can see that, both in successful and unsuccessful runs respectively, the probability of the optimum is more distant than in the corresponding previous scenarios from the highest probability. The low accuracy of the information about the problem that the probabilistic models contain is also reflected in the function values shown in Fig. 5.11(c). Although the MPS has a slightly higher function value than the best individual at the beginning of the runs, the MPS is lower than best individual in the remaining generations.

To conclude this part of the chapter, we would like to point out an interesting relationship between the probability and function values. In unsuccessful runs, the probability of the optimum always starts to decrease a few generations after the function values of the best individual reaches the values of the MPS. As the behavior of the function values is similar in successful and unsuccessful runs, this could suggest that the cross between both curves indicates a critical moment in the search.

5.6 EDA behavior solving ±J Ising

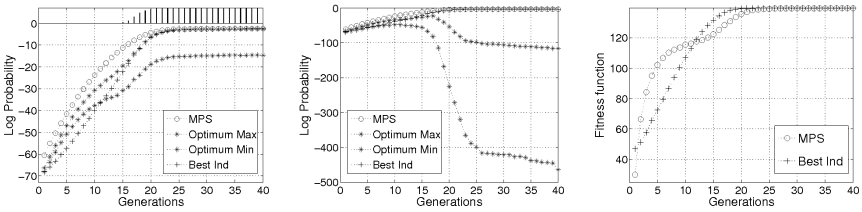
5.6.1 ± J Ising description

As explained in Section 5.5, the main difference between both versions of $2D$ Ising spin glass, is the range of values chosen for the couplings J_{ij}. In the present problem, the couplings J_{ij} are randomly set to either $+1$ or -1 with equal probability. This version, that will be called $\pm J$ *Ising*, could have different spin configurations that reach the ground state (lowest energy) and therefore many optimal solutions may arise. As in the previous case, 100 $\pm J$ Ising instances were generated using the Spin Glass Ground State server[3]. This server also provided the value of the minimum energy of the system. As

[3] http://www.informatik.uni-koeln.de/ls_juenger/index.html

far as the fixed structural models are concerned, we use the same structures as in Gaussian Ising.

5.6.2 Using structural learning



(a) (b) (c)

Fig. 5.12. Probability values and function values when EBNA is applied to $\pm J$ Ising using Algorithm B with population size m. We have 466 out of 500 successful runs with population size m and 226 out of 500 with population size $m/2$. On average, EBNA has reached 126 different optimal solutions at the end of the run with m and 55 with $m/2$. (a) Successful runs. (b) Unsuccessful runs. (c) Successful runs.

The analysis of problems with several optima reveals important changes in the internal behavior of the algorithm. Although the number of successful runs both with population size m (466 out of 500) and $m/2$ (226 out of 500) is very similar to Gaussian Ising, clear differences appear in the probability values. Fig. 5.12(a) shows that the probabilities assigned to the reached optima increase together during the generations. We have observed that the probability of the MPS does not tend to 1 (0 in logarithmic scale) as in unimodal problems. These facts indicate that the probability distribution is shared out among different optimal solutions in the last generations. This is verified by the accumulated entropy of the population (Fig. 5.13) which is greater than 0 at the end of the run. In Fig. 5.14, we illustrate the specific probability values of the MPS. We have seen that the MPS converges to higher probability values with $m/2$ because in this case, the average number of optimal solutions at the end of the run is lower than with m. In the same figure, we can see that this situation is repeated for unsuccessful runs (analysis shown in Fig. 5.12(b)). This indicates that, although the optimum is not found, the algorithm reaches different solutions at the end of the run. The results for the accumulated entropy (see Fig. 5.13) confirm this behavior. An interesting behavior in this problem is that the MPS reaches higher probability values in successful runs than in unsuccessful ones (see Fig. 5.14). This is another issue that would deserve a specific analysis.

In Fig. 5.12(c) we show the function values for the MPS and the best individual. We can see how the best individual clearly exceeds the MPS after

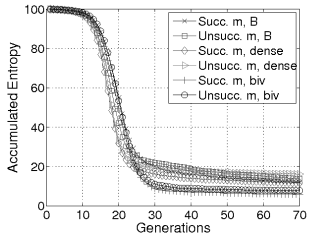


Fig. 5.13. Accumulated entropies of the population when EBNA solves $\pm J$ Ising.

generation 12. This marked fluctuation of the MPS occurs in the same generations when the probabilities of the optima slightly separate from the highest in the distribution in successful runs (Fig. 5.12(a)). Moreover, the probabilities of the optima start to decrease in unsuccessful runs after the cross between the curves of function values. This supports the idea that this phase of the search is critical in order to reach the optimum.

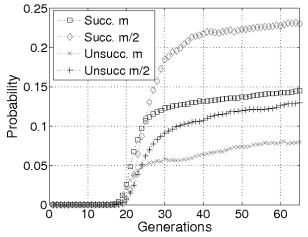
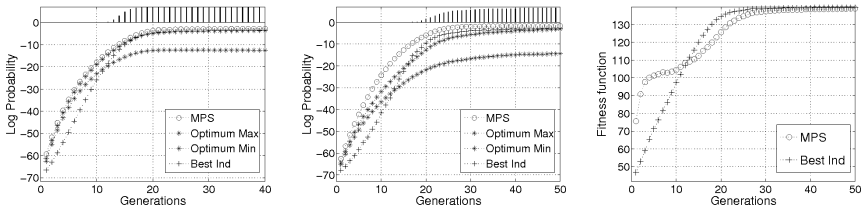


Fig. 5.14. Different curves of probability for the MPS when EBNA is applied to $\pm J$ Ising using Algorithm B. The curves correspond to successful runs with population size m (Succ. m), successful runs with $m/2$ (Succ. $m/2$), unsuccessful runs with m (Unsucc. m) and unsuccessful runs with $m/2$ (Unsucc. $m/2$).

5.6.3 Using fixed structures

In Fig. 5.15(a) we report the analysis of the probability values when the dense structure is introduced and the population size given by bisection is used. In the first generations, we can observe that the probabilities for the optima and the MPS are especially close. We also note that the maximum probability assigned to the set of optima is very close to the MPS during the search. According to the experiments, in this problem, the influence of the selected direction for the arcs is less dramatic than in Gaussian Ising. In fact, EBNA reaches the optimum in 463 out of 500 runs against 283 out of 500 in Gaussian Ising. This indicates that, depending on the problem, the same

amount of information introduced in the structural model can have a different impact both on the probability values and the performance of the algorithm. It could depend on properties of the search space such as multimodality.



(a) (b) (c)

Fig. 5.15. Probability values and function values when EBNA is applied to $\pm J$ Ising using fixed structures with population size m. (a) Probability values for the dense structure. We have 463 out of 500 successful runs and, on average, EBNA has reached 138 different optimal solutions at the end of the runs. (b) Probability values for the bivariate structure. We have 105 out of 500 successful runs and, on average, EBNA has reached 47 different optimal solutions at the end of the runs. (c) Function values for the bivariate structure.

In Fig. 5.15(b) and 5.15(c) we show the results obtained when EBNA uses the bivariate structural model. In this case, we have 105 out of 500 successful executions. This is a clear improvement in the performance of the algorithm with regards to Gaussian Ising in this same scenario. This enhanced performance is reflected in the analysis of the function values (Fig. 5.15(c)). In the first generation the MPS is clearly better than the best individual in the population. Moreover, at the beginning of the run, its difference is even more noticeable than in the case of EBNA using Algorithm B.

5.7 EDA behavior solving Max-SAT

5.7.1 Max-SAT description

The last problem in our analysis is the maximum satisfiability or Max-SAT problem, which has been often used in different works about EDAs (Pelikan and Goldberg (2003); Brownlee et al. (2007)). Put simply, given a set of Boolean variables $\boldsymbol{x}$ and a Boolean expression ϕ, SAT problem asks if there is an assignment of the variables such that the expression ϕ is satisfied. In a Boolean expression we can combine the variables using Boolean connectives such as $\wedge$ (logical and), $\vee$ (logical or) and $\neg$ (negation). An expression in the form x_i or $\neg x_i$ is called a literal.

Every Boolean expression can be rewritten into an equivalent expression in a convenient specialized style. In particular, we use the *conjunctive normal*

form (CNF) $\phi = \bigwedge_{i=1}^{q} C_i$. Each of the q C_js is the disjunction of two or more literals which are called clauses of the expression ϕ. We work with clauses of length $k = 3$. When $k \geq 3$, the SAT problem becomes NP-Complete (Cook (1971)). An example of a CNF expression with 5 Boolean variables and 3 clauses would be, $\phi = (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_4 \vee \neg x_2)$.

The Max-SAT problem has the same structure as SAT, but the result, for an assignment $\boldsymbol{x}$, is the number of satisfied clauses instead of a truth value. In order to solve Max-SAT, the assignment $\boldsymbol{x}$ that maximizes the number of satisfied clauses must be found. Thus, the optimization function can be written as,

$$f_{Max-SAT}(\boldsymbol{x}) = \sum_{i=1}^{q} \phi(C_i) \tag{5.7}$$

where each clause C_i of three literals is evaluated as a Boolean expression that returns 1 if the expression is *true* or 0 if it is $false$. Since C_i is a disjunction, it is satisfied if at least one of its literals is *true*. The variables can overlap arbitrarily in the clauses.

Particularly, we work with the Uniform Random-3-SAT problems obtained from the SATLIB (Hoos and Stutzle (2000)) repository. All the instances used are satisfiable. The presented results comprises 100 instances of 100 variables and 430 clauses. It is important to note that there could be several assignments $\boldsymbol{x}$ that satisfy all clauses and therefore, this problem could have different optimal solutions.

5.7.2 Structures related to the problem

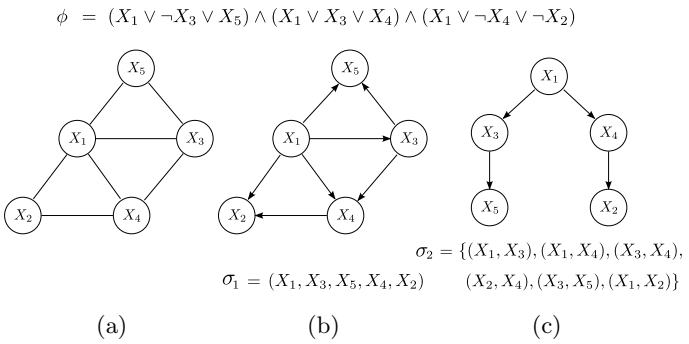


Fig. 5.16. Structures for Max-SAT given a SAT expression ϕ. (a) Related undirected graph, (b) related Bayesian network where σ_1 is the ancestral order and (c) related tree structure where σ_2 is an order to add edges in the tree.

As different Max-SAT instances have different interactions among variables, a particular structure for each instance is needed. In order to create the

dense structure, we join the variables belonging to the same clause C_i with edges. This step is illustrated in the example of Fig. 5.16(a) where a SAT formula is proposed. Now, in order to create a Bayesian network structure, we must direct the edges without creating cycles. In order to do this, we use an ancestral order which tries to minimize the number of parents per variable. Thus, the variables are ordered from the highest to the lowest number of overlaps in the clauses of the SAT instance. This type of structure is illustrated in Fig. 5.16(b) where σ_1 is the defined ancestral order. However, obtaining the MPS for such dense Bayesian network would be unfeasible due to the size of the cliques (up to 70 variables). Therefore, we were forced to reduce the complexity of the structure by deleting some edges. The high density of the interactions between variables in Max-SAT only allows us to work with two parents per variable. Thus, for each variable we select the two parents that correspond with the most frequent interactions with the child obtained from the clauses.

To create the bivariate structure, in this case a tree, we have followed a procedure similar to the Chow-Liu algorithm (Chow and Liu (1968)). In Fig. 5.16(c) we illustrate a possible final result for a particular SAT formula. Firstly, we create an order σ_2 for pairs of variables, related to the number of times that each couple of variables appear together in the SAT clauses, from the highest to the lowest. This is the scoring criterion for the arcs. Starting with an empty structure and following such an order, at each step we add an undirected edge without creating cycles. If there are ties, the selection is random. At the end of the procedure, the root of the tree is the most overlapped variable in the SAT formula taken from the most frequent couple.

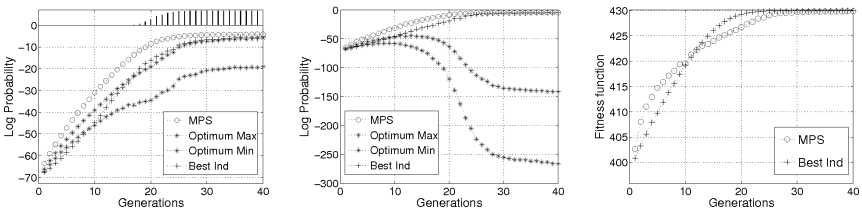
The Max-SAT problem would be a very interesing benchmark to conduct an analysis of the learning limits as it was done in the previous chapter.

5.7.3 Using structural learning

In general, the analysis of EBNA when it is applied to Max-SAT shows similar behavior patterns to $\pm J$ Ising. However, as we previously discussed, each problem provides particular nuances to the analysis. For Max-SAT, we only show the results when EBNA uses the population size given by bisection because this parameter has a lower impact on the algorithm. In this problem, EBNA is only able to obtain bisection sizes for 20 out of 100 instances. In those instances where we do not have a bisection size, we use the maximum population size allowed in the experimentation (2^{14}). In order to analyze unsuccessful runs in instances for which EBNA is not able to reach the optimum, we introduced 350 optimal configurations obtained by a specific solver[4] for Max-SAT.

As we can see in Fig. 5.17(a), the probability assigned to the set of optima is far from the highest in the distribution and this is reflected in the number

[4] http://minisat.se/



(a) (b) (c)

Fig. 5.17. Probability values and function values when EBNA is applied to Max-SAT using Algorithm B with population size m. We have 95 out of 500 successful executions. On average, EBNA has reached 1452 different optimal solutions at the end of the runs. (a) Successful runs. (b) Unsuccessful runs. (c) Successful runs.

of successful runs, in this case, 95 out of 500. Throughout the chapter, we have seen that the more distant the MPS and the optima are, the lower the performance in terms of ratio of successful runs is. In this problem, the curves of probability indicate a lower exponential growth than in the rest of the problems. In fact, the different optimal solutions start to be reached in later generations (bars on the top of Fig. 5.17(a)). We also note that the MPS reaches low probability values at the end of the runs (approximately, 0.05 for Max-SAT against 0.15 for $\pm J$ Ising in the same scenario). This is due to the high number of optima that EBNA is finding. Particularly, on average, we have 1452 different optimal solutions at the end of the runs.

In unsuccessful runs (Fig. 5.17(b)), the probabilities of the optima reach much lower maximum values than in the rest of the problems in the same scenario. Regarding the function values (Fig. 5.17(c)), although the MPS slightly outperforms the values of the best individual at the beginning of the run, this last solution is better than the MPS during a noticeable number of generations. Once again, the probabilities assigned to the optima start to decrease in unsuccessful runs some generations after the curve of the best individual crosses the curve of the MPS. The high number of optimal solutions, the great distance in probability between the MPS and the optima and the low quality of the function values of the MPS, reflect the hardness of this problem for EBNA.

5.7.4 Using fixed structures

In Fig. 5.18 we provide the probability values for successful runs. Although the dense structures only have a maximum of two parents per node, we can see the influence of this type of structure in the analysis. If we compare the dense structure with Algorithm B, we see that not only the maximum probability assigned to the set of optima is closer to the highest probability in the distribution, but also there is a lower difference in probability among the curves shown in the chart. Nevertheless, in all cases the optimum starts to

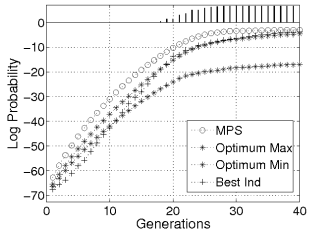


Fig. 5.18. Successful runs when EBNA is applied to Max-SAT using the dense structure with population size m. We have 31 out of 500 successful runs. On average, EBNA has reached 400 different optimal solutions at the end of the run.

be reached when the maximum probability of the optima has the value of 20 approximately. In contrast with structural learning, with fixed structures, EBNA reaches a lower number of optimal solutions at the end of the run, and this fact is also reflected in the final probability values. In this problem, the analysis shows a very similar behavior for both fixed structures.

5.8 Related work

In Mühlenbein et al. (1999), an analysis of the probability assigned by EDAs to the optimum solution is carried out for the Boltzmann EDA (BEDA) and factorized distribution algorithms (FDAs) that use valid and invalid factorizations. The analysis of the probabilities, which was carried out for a toy example, served to illustrate that, under the infinite population assumptions made by BEDA, the use of a valid factorization is a sufficient but not necessary condition for a steady increase, until convergence, of the probability given by BEDA to the optimum. Our work can be seen as an extension of the work presented in Mühlenbein et al. (1999) in the sense that we investigate the probabilities of EDAs that apply structural and parametric learning of a more complex class of models and across a range of different problems. We also provide a method to exactly determine the most probable solution given by the model.

As previously discussed, most of the research done concerning the models learned by EDAs based on Bayesian networks has focused on structural descriptors of the networks, and specifically on the type (i.e. correct or spurious) and number of the network edges (Hauschild and Pelikan (2008); Echegoyen et al. (2008); Hauschild et al. (2009); Lima et al. (2007); Echegoyen et al. (2007); Lima et al. (2008)). The analysis of the Bayesian network edges learned by EDAs has allowed to study the effect of the selection and replacement procedures (Hauschild et al. (2009); Lima et al. (2007)) as well as the learning method (Echegoyen et al. (2008, 2007); Lima et al. (2008)) in the accuracy of the models learned by EDAs and the efficiency of these algorithms. A

more recent work (Lima et al. (2008)) considers the likelihood given to the selected set during the model learning step as another source of information about the behavior of the algorithm. In this case, not only the structure but also the probabilities are taken into consideration when computing the model descriptor. Nevertheless, none of the previously mentioned papers uses the probabilities given by the models to some distinguished solutions (e.g. most probable explanation, known optimum, etc.) as a means to reveal information about EDAs. In no case are there references to the most likely solution that could be sampled from the learned model.

For EDAs that use Markov models (Shakya (2006); Shakya and Santana (2012)), different issues related with the relationship between the fitness function and the probabilistic models learned by EDAs have been investigated. Relevant to the work presented in this chapter, is the use of the models learned by the distribution estimation using Markov network algorithm (DEUM) (Shakya et al. (2005); Shakya and McCall (2007); Brownlee et al. (2012)) as predictors of the fitness function.

In Brownlee et al. (2008), the product moment correlation coefficient between the Markov model learned by DEUM and the fitness function is used to measure the quality of the model as a fitness function predictor. For a given solution, the prediction is the value given by the Markov model to the solution. The quality of the model is measured using the correlation computed from samples of the search space. Furthermore, the prediction accuracy of Markov models with different structural complexity is investigated for different selection strategies and population sizes.

A substantial difference between the work presented in Brownlee et al. (2008) and the results introduced in this chapter is that the analysis of the prediction given by the models is constrained to the solutions taken from the selected population or random samples. The most probable explanation given by the model is not computed. Another difference is that the evolution of the models throughout the generations is not analyzed. By computing the most probable individual given by the model at each generation, we are able to obtain a dynamic view of the quality of the probability model.

5.9 Conclusions

In this work we have analyzed EDAs from a quantitative point of view in order to better understand their internal behavior. Through the recording of probability and function values for a set of distinguished solutions during the search, we have directly studied the probability distributions generated by this type of algorithms. More specifically, the proposed analysis has allowed us to investigate basic open issues raised in Section 5.1, whose study entails a deeper understanding and development of EDAs. Now, we return to these questions, providing the new knowledge that we obtained throughout the study.

- How does the probability assigned to the optimal solution by the probability distributions evolve during the search?

We can distinguish different scenarios depending on the number of optimal solutions of the function to be optimized and the success of the search. In general, in order to reach an optimal solution, its probability must exceed a certain threshold which can vary depending on the intrinsic characteristics of the problem. On the one hand, when EBNA is applied to unimodal problems (Trap5 and Gaussian Ising) and the optimal solution is found, its probability continuously increases until it reaches the value of 1. One exception is function Trap5 and the bivariate structure where the probability of the optimum decreases at the beginning of the run and it increases in the last generations.

On the other hand, when EBNA successfully solves multimodal problems ($\pm J$ Ising and Max-SAT), it is able to reach a subset of the optimal solutions and their probability values also increase during the search. In these problems, the probability is distributed among different solutions at the end of the run (note that the number of generations is limited). Thus, the non-convergence to 1 of the probability values of the MPS or the best individual of the population (both probability curves always rise simultaneously) reflects the multimodality of the function. Moreover, these probability values are lower when the algorithm reaches a higher number of optima. This finding can be used to detect multimodality when an unknown problem needs to be faced.

In unsuccessful runs, the probability of the optimum always has a similar pattern. At the beginning of the run, it increases together with the probability of the MPS and the best individual of the population. However, after a certain generation, before reaching a specific probability threshold, it decreases rapidly.

Both the probability of the MPS and the best individual of the population in logarithmic scale, accurately show how that the algorithm converges as the generations advance. Therefore, by monitoring the probability of the best individual, it would be possible to know the speed of convergence of the algorithm and then, detect a possible premature convergence. According to this, modifications in the replacement technique could be performed in order to regulate the diversity of the population. This information could also be useful in order to distinguish between exploration and exploitation phases. Thus, we could stop the search at the right time (before the probability of the optimum starts to decrease) and take advantage of the information contained in the probabilistic model by using exploitation techniques.

The population size also influences the probability assigned to the optimum during the search and this is reflected in the number of successful runs. When the population size given by bisection is used, the probability values for the optimum tend to be closer to the highest probability in the distribution and, in most of the cases, this is beneficial in order to solve the problem.

- How does the accuracy of the information about the problem contained in the structural model influence the internal behavior of EDAs?

The results support the conclusion obtained in Hauschild et al. (2009) regarding the difficulty of creating adequate probabilistic models by hand even with complete knowledge of the problem structure. However, the quantitative analysis of the models reveals that the use of information about the problem has an important impact in the internal behavior of the algorithm.

In particular, we provide two clear conclusions. Firstly, when we are able to introduce all the interactions between the variables of the problem, the probability of the optimum tends to be closer to the highest probability in the distribution. Secondly, when we introduce this information, the function value for the MPS is very close to the optimum from the beginning of the run. However, despite these favorable properties, these type of models do not always perform satisfactorily. The experimental results indicate that the PLS sampling method (one of the most widely used) does not extract all the valuable information contained in the probabilistic models. For this reason, in order to take advantage of both the high probability assigned to the optimum and the high quality function values of the MPS, the use of a sampling based on exact or even approximate belief propagation techniques (Mendiburu et al. (2007); Lima et al. (2009); Mendiburu et al. (2012)) could be beneficial. Another reason we point out for such non-constant behavior is the direction assigned to the arc, which conditions the order the variables will be sampled by the PLS technique. A possible solution for this issue could be to, according to a given score, look for the best direction for the arcs at each step of the algorithm.

When the information about the problem that the probabilistic model contains is reduced, the probability of the optimum is more distant from the highest in the distribution. Moreover, the function value of the best individual is closer to the MPS in the first generations. These facts justify the poor performance of the algorithm in these cases.

- How does the function value for the most probable solution evolve during the search?

The function value for the MPS always increases during the search until it stabilizes in the last generations. At the beginning of the run, it is usually better than the best individual in the population.

As we previously said, the difference between the function values of the MPS and the best individual increases when a dense structure is used. Another interesting observation is that this difference also reflects the impact that the population size has on a particular problem. Thus, when we increase the population size in order to solve Trap5, both the difference between the function values analyzed and the number of successful runs clearly increases. However, for Max-SAT, increasing the population size hardly influences the curves of function values and the performance of the algorithm. Therefore, by analyzing the MPS and the best individual with different population sizes, we can predict, without additional information about the problem, if increasing

this parameter will be useful in order to obtain better results or if we need to look for other solutions to improve the performance of the algorithm.

By using these function values, we believe that it is also possible to identify different phases of the search. According to the results, in unsuccessful runs, the probability of the optimum starts to decrease shortly after the function value of the best individual outperforms the function value of the MPS. Moreover, the optimum is never reached before this event. It could be used to identify the end of the exploration stage and avoid a premature convergence.

In summary, the difference in function value between the MPS and the best individual could be used, i) to improve the setup of EDA parameters, ii) to measure the quality of the information introduced about the problem in the model, iii) to measure the quality of sampling methods and iv) to detect critical phases in the search. Finally, the analysis carried out in this work has become useful in order to learn about different aspects of the algorithm and propose improvement solutions. We believe that similar approaches to analyze EDAs can be especially useful for other EDA practitioners both in fundamental research and in real problem applications.

6

On the Taxonomy of Optimization Problems under EDAs

6.1 Introduction

In the previous chapter, we analyzed EDAs by looking at the probability distributions generated by this type of algorithms during the search. Constant patterns of behavior were identified which provided us with new knowledge about EDAs. Nevertheless, we also saw that the curves of probability values had different nuances in each optimization problem. In this regard, the behavior of an EDA can be specified by the sequence of probability distributions generated at each generation and, depending on the problem, the algorithm can exhibit a different performance. Based on this description of the EDA, identifying and classifying the different behaviors, and how these behaviors relate with the characteristics of the optimization problems, is a fundamental issue to understand the underlying mechanisms that govern this type of algorithms. In general, understanding the relationship between a search algorithm and the space of problems is a fundamental issue in the optimization field.

In the current chapter, we lay the foundations to elaborate taxonomies of problems under EDAs based on the probability distributions generated at each generation. With the aim of developing this type of taxonomies, we start by considering whether it is possible to group the optimization problems according to the behavior of the EDA. Therefore, we are questioning the existence of groups of problems in which the EDA exhibits a similar performance regarding the optimization purpose. In order to start the study of this topic, we make the following three assumptions: i) we consider EDAs with infinite population, ii) the selection scheme is based on the rank of the solutions and iii) the algorithm is applied in the space of injective functions. These assumptions are needed to address the development of the basic theoretical foundations, nevertheless, the assumptions ii) and iii) could be relaxed to further increase the scope of the study.

In order to elaborate taxonomies of optimization problems under EDAs that satisfy these assumptions, a crucial element is the definition of an equivalence relation between objective functions. The equivalence relation, which

is based on the the sequences of probability distributions generated during the search, partitions the space of functions into equivalence classes. We will see that it is possible to assert that the algorithm has the same behavior for the functions belonging to the same class. In turn, we show that only the probabilistic model is able to create partitions of the space of problems. From the definitions provided, it can be deduced that all the objective functions are equivalent when the probabilistic model is able to exactly recover the underlying distribution of the selected individuals. Therefore, we can say that all the optimization problems are equally difficult from the point of view of this type of EDA. This crucial role of the probabilistic model supports the importance that has been attributed to it in the literature regarding the classification of different EDAs and the development of new techniques and studies.

The partition of the space of problems under EDAs is studied in depth for a simple algorithm whose probabilistic model assumes independence among the variables of the problem. The key point to understand the behavior of an EDA for a given problem is the way in which the objective function is related with the probabilistic model. We show that this relation can be expressed by means of a structure of sets. For this type of univariate EDA, the necessary and sufficient condition to identify equivalent functions is provided. Next, we develop the operators that allow us to describe the functions in the same class and count the number of elements per class. Another fundamental topic that we take into account is the relation between the behavior of the EDA and the properties of the objective function. In particular, we study the connection between the equivalence classes and the local optima of the objective functions belonging to each class. We show that the functions in the same class have the same number of local optima and in the same ranking positions. This fact reveals the intrinsic connection between local optima and any EDA that introduces a univariate probabilistic model. This link has only been shown for specific EDA implementations (González et al. (2001); Zhang (2004)).

Finally, numerical simulations of a univariate EDA that implements tournament selection (Zhang (2004)) are conducted. The algorithm is applied to the injective functions defined over the search space $\{0,1\}^3$. The partition of the space of problems in equivalence classes allows us to carry out a detailed analysis of the different EDA behaviors. Through the numerical analysis, we mainly study the complexity of the problems belonging to each class. Due to the relevant role that the local optima play in EDAs that assume independence among the variables, we present the difficulty of the problems in relation to that number. Moreover, the experiments are useful in order to illustrate the taxonomy of problems which is abstractly presented throughout the chapter.

The rest of the chapter is organized as follows. Section 6.2 formally introduces the problems to solve and discusses some elements of the EDA procedure that are important in this chapter. Section 6.3 presents EDAs with infinite population and provides the specific definitions involved in the framework of the current work. In Section 6.4, the concept of equivalence between functions is presented and discussed. Section 6.5 regards the study of the equivalence

classes under a simple EDA. Firstly, it explains and establishes the sufficient and necessary condition to decide the equivalence between two functions. Secondly, the operators to describe the functions in the same class are deduced. Lastly, the relationship between the equivalence classes and local optima is presented. In Section 6.6, numerical experiments are conducted. Finally, Section 6.7 draws the conclusions obtained during the study.

6.2 Background

In this chapter we consider the following optimization problems as introduced in Chapter 1:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{6.1}$$

where $S = \{0,1\}^n$ is the search space, $\boldsymbol{x} = (x_1, \ldots, x_n) \in S$ is a solution and $f : S \to \mathbb{R}$ is the objective function. For the sake of simplicity, we deal with binary solutions although the fundamental results provided do not depend on the cardinality of the variables or the codification of the solutions. The cardinality $|S|$ of the search space is therefore 2^n. Throughout the chapter, we assume that the function $f(\boldsymbol{x})$ is injective. Thus, for each $z \in f(S)$ there is only one $\boldsymbol{x} = (x_1, \ldots, x_n) \in S$ such that $f(\boldsymbol{x}) = z$. The results could be generalized to non-injective functions because the most basic definitions that we present do not depend on the injectivity of the function.

A crucial fact that we use in the development of the current work is the following. A function $f(\boldsymbol{x})$ naturally induces a permutation σ of the solutions of the search space S. This permutation σ is thought as a 2^n-tuple of the elements in S ordered according to function values $f(S)$. The first solution of σ, namely $\boldsymbol{x}_1$, has the highest function value, $\boldsymbol{x}_2$ has the second highest and so on, with the last solution $\boldsymbol{x}_{2^n}$ being the one with the lowest function value:

$$f(\boldsymbol{x}_1) > f(\boldsymbol{x}_2) > \ldots > f(\boldsymbol{x}_{2^n}).$$

Of course, the first solution $\boldsymbol{x}_1$ of σ corresponds to the solution $\boldsymbol{x}^*$ that solves Problem 6.1. Independently of the specific function values $f(S)$, whenever they provide the same ranking of the solutions $\boldsymbol{x} \in S$, the function is represented by the same permutation σ. In the case of injective functions, we have $2^n!$ permutations σ and the set containing all possible σ is denoted by Π. A σ permutation can also be seen as a ranking of the solutions. From now on, the objective function $f(\boldsymbol{x})$ and the corresponding σ can be used indistinctly as synonyms.

We also define the bijective function $\sigma : S \to \{1, \ldots, 2^n\}$ in order to deal with the permutation σ. Thus, $\sigma(\boldsymbol{x}) = \tau$ gives the position τ of the solution $\boldsymbol{x} \in S$ in the permutation σ. We make certain abuse of notation because σ can represent both a permutation of the solutions in S and the function $\sigma(\boldsymbol{x})$ defined above. Nonetheless, both elements refer the same entity and they will be used without ambiguity as needed.

In Table 6.1, we provide an example of the permutation σ induced by a function $f(\boldsymbol{x})$. In the first column are the original function values, in the second column the corresponding 2^n-tuple σ of the solutions $\boldsymbol{x} \in S$ is disposed and in the third column we indicate the ranking values given by $\sigma(\boldsymbol{x})$.

$f(\boldsymbol{x})$	σ	$\sigma(\boldsymbol{x})$
100	$(1,1,1)$	1
50	$(0,1,0)$	2
45	$(0,0,1)$	3
20	$(1,0,0)$	4
10	$(0,1,1)$	5
3	$(1,0,1)$	6
1	$(1,1,0)$	7
0	$(0,0,0)$	8

Table 6.1. Example of permutation σ induced by an injective function with $n = 3$ variables.

As it is known, EDAs use explicit probability distributions $p(\boldsymbol{x})$ with the aim of finding the optimal solution $\boldsymbol{x}^*$ and solve Problem 6.1. In this chapter, we would like to see the EDA procedure in terms of the probability distributions involved in the search (Santana et al. (2009b)) (see Fig. 6.1). Firstly, D_t denotes the EDA population at generation t and $p(\boldsymbol{x}, t)$ is the underlying probability distribution of this sample. Secondly, $p^s(\boldsymbol{x}, t)$ is the probability distribution of the selected individuals D_t^s. Finally, $p^a(\boldsymbol{x}, t)$ is the distribution given by the probabilistic model chosen to approximate $p^s(\boldsymbol{x}, t)$. Once we have $p^a(\boldsymbol{x}, t)$, the next generation D_{t+1} is constructed by sampling this distribution.

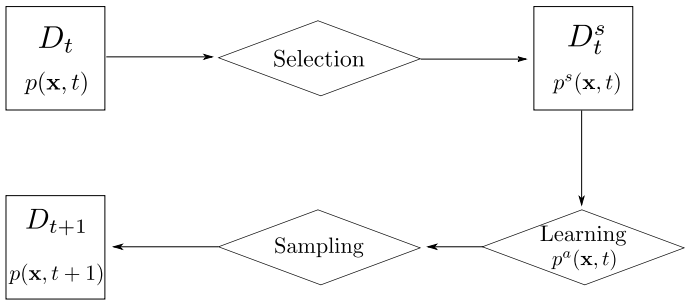


Fig. 6.1. Probability distributions determined by the components of an EDA. D_t, D_t^s, D_{t+1}: population and selected individuals at generation t and population at generation $t+1$; $p(\boldsymbol{x}, t)$, $p^s(\boldsymbol{x}, t)$, $p^a(\boldsymbol{x}, t)$: Probability distributions of the population, the selected set and the probabilistic model approximation at generation t.

In any evolutionary algorithm, selection is one of the fundamental operators and therefore, a wide variety of proposals can be found in the literature. According to Lee and El-Sharkawi (2008), the selection schemes can be classified into two groups: proportionate selection and ordinal-based selection. Proportionate-based procedures select individuals based on their specific function values. Ordinal-based selection procedures select individuals based on their rank within the population. Thus, this class of selection only takes into account qualitative information about the function i.e. it only uses the fact that $f(\boldsymbol{x}) > f(\boldsymbol{y})$ instead of the real value given by the function.

In this work, we assume that the algorithm uses this last type of selection. We must say that assuming ordinal-based selection is not a insurmountable restriction because the definitions that we present could be extended to selections that take into account the specific function values. Common examples of selection schemes based only on the rank of the solutions are truncation, tournament, linear ranking or exponential ranking selection. These schemes are considered in a wide variety of evolutionary algorithms, both in solving real problems and in theoretical studies (Blickle and Thiele (1996); Prügel-Bennett (2000); Zhang (2004)).

6.3 The infinite population EDA model

The application of the EDA scheme to face optimization problems can involve an unapproachable variety of situations and behaviors. With the aim of dealing with all possible EDA behaviors, we use an infinite population model (e.g. Mühlenbein et al. (1999); Zhang and Mühlenbein (2004); Zhang (2004)) in which the algorithm is reduced to its essence and random errors are canceled.

In this type of EDA model, it is assumed that the empirical probability distribution induced by the solutions in D_t and D_t^s (Fig. 6.1) will converge to the underlying probability distributions $p(\boldsymbol{x}, t)$ and $p^s(\boldsymbol{x}, t)$ respectively, as the size of the sample tends to infinity. Therefore, $p(\boldsymbol{x}, t)$ and $p^s(\boldsymbol{x}, t)$ could be thought of as the population and the selected individuals at iteration t in EDAs with infinite population (Zhang and Mühlenbein (2004)). Consequently, we can assume that $p(\boldsymbol{x}, t+1) = p^a(\boldsymbol{x}, t)$.

6.3.1 The algorithm and the population

In Alg. 6.1 we formally describe the infinite population EDA procedure that we consider in the current work. This algorithm will be simply denoted by $\mathcal{A}$.

It is established that the first step in algorithm $\mathcal{A}$ is to create the permutation σ of the solutions $\boldsymbol{x} \in S$. The permutation σ that the objective function provides is a crucial element of the algorithm.

As commented above, an infinite population can be represented by the probability distribution $p(\boldsymbol{x}, t)$ that the algorithm manages at time t. Nevertheless, algorithm $\mathcal{A}$ does not explicitly work with the distributions $p(\boldsymbol{x}, t)$.

1 $\sigma \leftarrow$ permutation of the solutions $\boldsymbol{x} \in S$ given by $f(\boldsymbol{x})$
2 $\mathbf{p}_{t=0} \leftarrow$ generate the initial population
3 **do**
4 Compute the probability of selection through the selection operator ϕ as $\mathbf{p}_t^s = \phi(\mathbf{p}_t)$
5 Compute $\mathbf{p}_t^a = \mathcal{M}(\mathbf{p}_t^s, \sigma)$ to approximate $\mathbf{p}_t^s$.
6 $\mathbf{p}_{t+1} \leftarrow \mathbf{p}_t^a$
7 $t \leftarrow t + 1$
8 **until** Convergence

Alg. 6.1: EDA with infinite population, $\mathcal{A}$.

Instead, at each generation, this algorithm manages a probability vector $\mathbf{p} = (p_1, p_2, \ldots, p_{|S|})$ where each probability value p_i in $\mathbf{p}$ corresponds to the probability of the solution $\boldsymbol{x}_i$, which is in the position i of the permutation σ. It is always interpreted that the first value p_1 of the vector $\mathbf{p}$ is the probability of the optimal solution, p_2 is the probability of the second best solution, and so on, with the last probability p_{2^n} corresponding to the solution with the worst function value. Accordingly, we have the probability vectors $\mathbf{p}^s$ and $\mathbf{p}^a$ representing the selected population and the approximation respectively. To be absolutely precise, the probability vectors at time t should be represented as $\mathbf{p}_t = (p_1^t, p_2^t, \ldots, p_{|S|}^t)$. Note that, unlike the probability distribution $p(\boldsymbol{x})$, a vector $\mathbf{p}$ implies no specific assignment of probabilities to $\boldsymbol{x}$ configurations. In order to obtain the probability distribution $p(\boldsymbol{x}, t)$ that algorithm $\mathcal{A}$ manages at time t, we link the vector $\mathbf{p}_t$ and the permutation σ through the pair $(\mathbf{p}, \sigma)$. The pair $(\mathbf{p}, \sigma)$ induces a probability function $p(\boldsymbol{x})$ such that $p(\sigma^{-1}(i)) = p_i$. Therefore, we have that $p_1 = p(\sigma^{-1}(1))$ is the probability of the optimum, $p_2 = p(\sigma^{-1}(2))$ is the probability of the second best solution and so on.

The arrangement of the probability vector $\mathbf{p}$ and its relationship with the permutation σ is illustrated in Table 6.2. In this example, we consider two different permutations σ_1 and σ_2. We assume that algorithm $\mathcal{A}$ was applied to both functions and that it manages the same probability vector $\mathbf{p}_t$ at time t. We can see that, for example, p_1 is associated to the solution $(1, 1, 1)$ according to σ_1. However, according to σ_2, p_1 is associated to the solution $(0, 0, 0)$. Therefore, p_1 is the probability of the optimum in both cases, independently of the specific configuration of this solution. In the third column of Table 6.2, we can see that the same vector induces different probability distributions depending on σ. Note that, since we have $2^n!$ different permutations σ, a vector $\mathbf{p}$ can correspond to $2^n!$ different probability distributions.

The space of the possible probability vectors that the algorithm can generate is defined by the following simplex:

σ_1	$\mathbf{p}_t$	$p(\boldsymbol{x},t)$	$\mathbf{p}_t^s$
(1,1,1)	p_1	$p(\sigma^{-1}(1)) = p(1,1,1)$	p_1^s
(0,1,0)	p_2	$p(\sigma^{-1}(2)) = p(0,1,0)$	p_2^s
(0,0,1)	p_3	$p(\sigma^{-1}(3)) = p(0,0,1)$	p_3^s
(1,0,0)	p_4	$p(\sigma^{-1}(4)) = p(1,0,0)$	p_4^s
(0,1,1)	p_5	$p(\sigma^{-1}(5)) = p(0,1,1)$	p_5^s
(1,0,1)	p_6	$p(\sigma^{-1}(6)) = p(1,0,1)$	p_6^s
(1,1,0)	p_7	$p(\sigma^{-1}(7)) = p(1,1,0)$	p_7^s
(0,0,0)	p_8	$p(\sigma^{-1}(8)) = p(0,0,0)$	p_8^s

σ_2	$\mathbf{p}_t$	$p(\boldsymbol{x},t)$	$\mathbf{p}_t^s$
(0,0,0)	p_1	$p(\sigma^{-1}(1)) = p(0,0,0)$	p_1^s
(1,0,1)	p_2	$p(\sigma^{-1}(2)) = p(1,0,1)$	p_2^s
(1,1,0)	p_3	$p(\sigma^{-1}(3)) = p(1,1,0)$	p_3^s
(0,1,1)	p_4	$p(\sigma^{-1}(4)) = p(0,1,1)$	p_4^s
(1,0,0)	p_5	$p(\sigma^{-1}(5)) = p(1,0,0)$	p_5^s
(0,1,0)	p_6	$p(\sigma^{-1}(6)) = p(0,1,0)$	p_6^s
(0,0,1)	p_7	$p(\sigma^{-1}(7)) = p(0,0,1)$	p_7^s
(1,1,1)	p_8	$p(\sigma^{-1}(8)) = p(1,1,1)$	p_8^s

Table 6.2. Arrangement of the probability vectors and their relationship with the probability distributions. Example with $n = 3$ variables.

$$\Omega_{|S|} = \{(p_1, p_2, \ldots, p_{|S|}) : \sum_{i=1}^{|S|} p_i = 1, p_i \geq 0\}.$$

In addition, we establish that any initial vector $\mathbf{p}_0 = (p_1, \ldots, p_{2^n})$ satisfies that $p_i < 1$ for all $i \in \{1, \ldots, 2^n\}$. From any given $\mathbf{p}_0$, the application of algorithm $\mathcal{A}$ to a function f induces a deterministic sequence of probability vectors:

$$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots. \tag{6.2}$$

We use this sequence to describe the behavior of the algorithm. Note that if we took into account the probability distributions $p(\boldsymbol{x}, t)$ to describe the EDA behavior, the algorithm could always generate different sequences of distributions for each possible σ. The use of probability vectors $\mathbf{p}$ provides a higher level of abstraction, which is essential in order to group EDA behaviors.

6.3.2 The selection scheme ϕ

As previously indicated, we assume selection schemes based on the rank of the solutions within the population. Since in algorithm $\mathcal{A}$ any probability value p_i is interpreted as the probability of the solution $\sigma^{-1}(i)$ with position i in the permutation σ, the selection can be simply defined by a function $\phi : \Omega_{|S|} \rightarrow$

$\Omega_{|S|}$. This function modifies the probability values of the individuals according to the rank in which they are.

In order to create the partition of the space of problems, we do not need to consider any specific implementation of the selection operator ϕ. However, it will be essential for ϕ to satisfy three basic properties or axioms.

- Property 1 (Neutrality). The selection operator is independent of the configuration $\boldsymbol{x}$ of a solution. This operator only takes into account the fitness value $f(\boldsymbol{x})$ of the solution. Although this property is implicit in the definition of ϕ, we believe that it is worth a brief discussion. Thus, since we assume ordinal-based selection scheme, given $\mathbf{p}$, we always obtain the same the probability vector $\mathbf{p}^s = \phi(\mathbf{p})$ after selection, independently of σ. This fact is illustrated in the last column of Table 6.2 where we indicate that the probability vector after selection is the same in both functions.
- Property 2 (No degeneration). The selection operator can not assign extreme probabilities 1 or 0 to solutions whose probabilities are in the interval $(0, 1)$. More formally, the vector $\mathbf{p}^s = (p_1^s, \ldots, p_{2^n}^s)$, computed as $\mathbf{p}_t^s = \phi(\mathbf{p}_t)$ at generation t, satisfies that if $0 < p_i < 1$ then $0 < p_i^s < 1$ for all $i \in \{1, \ldots, n\}$ in every generation $t = 1, 2, 3, \ldots$. In addition, if $p_i = 0$ then $p_i^s = 0$. The convergence of the algorithm can only take place as a result of the evolutionary process when t tends to infinity.

The conditions mentioned above are needed in order to guarantee that the taxonomies of problems are valid for any implementation of ϕ. Therefore, the selection ϕ will play no role in the partition of the space of optimization problems.

In the context of EDAs, some examples of selection schemes formulated and studied under infinite population can be found in Mühlenbein et al. (1999), Zhang and Mühlenbein (2004) and Zhang (2004).

6.3.3 The approximation step $\mathcal{M}$

In algorithm $\mathcal{A}$, the approximation step deals with the probability distribution $p^s(\boldsymbol{x})$ of the selected individuals. Therefore, the probability vector $\mathbf{p}^s$ has to be related with the corresponding solutions $\boldsymbol{x}$ by means of the pair $(\mathbf{p}^s, \sigma)$. The approximation step is defined as a function $\mathcal{M} : \Omega_{|S|} \times \Pi \to \Omega_{|S|}$ and it is computed as $\mathbf{p}^a = \mathcal{M}(\mathbf{p}^s, \sigma)$ in the algorithm.

We assume that the function $\mathcal{M}$ approximates the distribution $p^s(\boldsymbol{x})$ by using a Bayesian network $(\boldsymbol{s}, \boldsymbol{\theta_s})$. According to the factorization expressed by the structure $\boldsymbol{s}$, the corresponding set of parameters $\boldsymbol{\theta_s}$ are calculated from $p^s(\boldsymbol{x})$ to obtain the probability distribution $p^a(\boldsymbol{x})$. This approximate distribution $p^a(\boldsymbol{x})$ will be in the space of probability distributions allowed by the structure $\boldsymbol{s}$. For the sake of simplicity, we assume that any Bayesian network $(\boldsymbol{s}, \boldsymbol{\theta_s})$ managed by algorithm $\mathcal{A}$ is implicit in the function $\mathcal{M}$.

Note that the approximation $\mathcal{M}$ is the only operator in $\mathcal{A}$ that takes into account the permutation σ. Therefore, $\mathcal{M}$ can translate the difference between functions to different behaviors of the algorithm.

6.4 Equivalent functions and equivalence classes

In this section we discuss the concept of equivalence between functions and provide the formal definitions. An EDA $\mathcal{A}$ induces deterministic sequences of probability vectors. An iteration of this algorithm is computed by a function $\mathcal{G} : \Omega_{|S|} \times \Pi \to \Omega_{|S|}$ as $\mathbf{p}_{t+1} = \mathcal{G}(\mathbf{p}_t, \sigma)$. The function $\mathcal{G}$ is a composition of the selection ϕ and the factorization function $\mathcal{M}$ (steps 4 and 5 in Alg. 6.1) such that $\mathcal{G} = \mathcal{M} \circ \phi$. Thus, the sequence of probability vectors induced by $\mathcal{A}$ can be expressed as the iterations of the function $\mathcal{G}$:

$$\mathbf{p}_0, \mathcal{G}(\mathbf{p}_0, \sigma), \mathcal{G}^2(\mathbf{p}_0, \sigma), \mathcal{G}^3(\mathbf{p}_0, \sigma), \ldots$$

where $\mathcal{G}^t(\mathbf{p}_0, \sigma)$ is the vector at iteration t.

The properties of functions similar to $\mathcal{G}^t$ have usually been studied by means of discrete dynamical systems. In González et al. (2001) and Zhang (2004), important insights and results about the convergence of some EDAs were provided using this approach. In the current chapter, the definition of equivalence does not consider specific implementations for $\mathcal{M}$ or ϕ and hence, we do not have a formulation of $\mathcal{G}$ to study the dynamics of the algorithms. Although the iterations of $\mathcal{G}$ are modeled as a dynamical system, we will take a more general perspective to describe the behavior of EDAs.

The definition of equivalence between objective functions under EDAs can be expressed as follows.

Definition 6.1 *Let σ_1 and σ_2 be the permutations induced by the objective functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ respectively. Let $\mathcal{A}$ be an EDA with any given ϕ and $\mathcal{M}$. We say that σ_1 and σ_2 are equivalent under $\mathcal{A}$, and by extension f_1 and f_2, if for any $\mathbf{p}_0 \in \Omega_{|S|}$, $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$ for all $t = 1, 2, 3, \ldots$.*

In a less formal way, we say that two functions are equivalent under $\mathcal{A}$ if the corresponding sequences of probability vectors induced by the algorithm are equal from any initial point. The equivalence between functions means that we have the same EDA behavior if we focus on the rank of the solutions instead of their specific $\boldsymbol{x}$ configurations. Therefore, if two functions are equivalent, we can say that the algorithm will have the same performance in terms of solving Problem 6.1. In Table 6.3, a very simple example with $n = 2$ variables illustrates the equality of vector sequences. Departing from the uniform distribution, algorithm $\mathcal{A}$ exactly induces the same sequence of vectors $\mathbf{p}_t$. If both sequences are equal from any initial $\mathbf{p}_0$ then σ_1 and σ_2 will be equivalent. Note that the sequences of probability distributions $p(\boldsymbol{x}, t)$ are different.

σ_1	$\mathbf{p}_0$	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	...	$\mathbf{p}_\infty$
(1,1)	0.25	0.4375	0.6836	0.8999	...	1
(0,1)	0.25	0.3125	0.2539	0.0962		0
(0,0)	0.25	0.1875	0.0586	0.0039		0
(1,0)	0.25	0.0625	0.0039	0.0000	...	0

σ_2	$\mathbf{p}_0$	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	...	$\mathbf{p}_\infty$
(1,0)	0.25	0.4375	0.6836	0.8999	...	1
(1,1)	0.25	0.3125	0.2539	0.0962		0
(0,1)	0.25	0.1875	0.0586	0.0039		0
(0,0)	0.25	0.0625	0.0039	0.0000	...	0

Table 6.3. Example with $n = 2$ variables of two equal sequences when $\mathcal{A}$ is applied to σ_1 and σ_2.

Definition 6.1 provides an equivalence relation because it is a reflexive, symmetric and transitive relation between functions. Given this equivalence relation, for each σ, we have the equivalence class of σ, denoted by $[\sigma]$. The equivalence relation partitions the space of functions into equivalence classes under an algorithm $\mathcal{A}$. The sequences of probability vectors generated by the algorithm uniquely identify the functions in a class.

The equivalence between functions is defined under a given algorithm $\mathcal{A}$ which implements certain ϕ and $\mathcal{M}$. However, as we discussed in the previous section, both operators do not play the same role. If two functions are equivalent under $\mathcal{A}$, then both functions will be equivalent for any given ϕ implemented in $\mathcal{A}$ satisfying Properties 1 and 2. However, two functions that are equivalent for $\mathcal{M}$ could not be so for $\mathcal{M}'$, which implements a different probabilistic model. We can conclude that only the factorization used to approximate $p^s(\boldsymbol{x})$ can create different partitions of the space of problems. This partition is independent of the implementation of ϕ.

By taking into account all the aforementioned definitions of algorithm, function and equivalence, we can deduce the following result assuming that ϕ satisfies the properties of neutrality and no degeneration.

Theorem 6.1 *Let $\mathcal{A}$ be an EDA whose implementation of $\mathcal{M}$ satisfies $p^a(\boldsymbol{x}, t) = p^s(\boldsymbol{x}, t)$ for all $t = 1, 2, 3, \ldots$. All the objective functions are in the same equivalence class under $\mathcal{A}$.*

Proof. The proof of the theorem is straightforward. Two functions σ_1 and σ_2 are equivalent if $\mathcal{A}$ generates the same sequence of probability vectors departing from any initial $\mathbf{p}_0$. Due to the neutrality of ϕ, we obtain the same vector $\mathbf{p}_0^s = \phi(\mathbf{p}_0)$ after selection for σ_1 and σ_2. Next, if $\mathcal{M}$ satisfies $p^a(\boldsymbol{x}) = p^s(\boldsymbol{x})$ then $\mathcal{M}(\mathbf{p}_0^s, \sigma_1) = \mathcal{M}(\mathbf{p}_0^s, \sigma_2)$ and therefore we have the same vector $\mathbf{p}_0^a$ for both functions. Since $\mathbf{p}_{t+1} = \mathbf{p}_t^a$, the algorithm obtains the same probability

vector $\mathbf{p}_1$ and consequently it will obtain the same vectors $\mathbf{p}_t$ at any time $t = 2, 3 \ldots$. $\square$

Therefore, if it is assumed that $p(\boldsymbol{x}, t+1) = p^s(\boldsymbol{x}, t)$, the algorithm has the same behavior for all the injective functions and hence, the same properties of convergence to the optimum (Mühlenbein et al. (1999); Zhang and Mühlenbein (2004)). As commented above, only the probabilistic model is able to create partitions of the space of problems. And moreover, if this model is able to exactly represent the distribution of the selected individuals, all the functions are in a single class. These results support the usual classification of EDAs which is carried out according to the probabilistic model implemented.

6.4.1 Descriptors of the behavior of EDAs

Once the space of problems is divided into equivalence classes, we could study the behavior that the algorithm exhibits in each class. In this regard, we discuss the role of two descriptors of the behavior of the EDA. On the one hand, the most basic descriptors are the sequences of probability vectors. By definition, we know that the algorithm induces the same set of sequences for any function in the same class and different sets of sequences for functions in different classes. Therefore, the set of sequences associated to any function of a class can be used to unequivocally represent the behavior of the corresponding class.

On the other hand, the behavior of an EDA can be described by using the basins of attraction of the solutions in S. Basin of attraction is a term used in dynamical systems that we adopt in a simplified manner. Roughly speaking, the basin of attraction of a point $\boldsymbol{x}$ in a dynamical system is the set of initial points that converge to $\boldsymbol{x}$. More formally, by using the function $\mathcal{G}$, we say that the basin of attraction of a solution $\boldsymbol{x} \in S$, is the set $\mathcal{Z} \subseteq \Omega_{|S|}$ such that $\forall \mathbf{p} \in \mathcal{Z}$, $lim_{t\to\infty}\mathcal{G}^t(\mathbf{p}, \sigma) = \mathbf{p}_{\boldsymbol{x}}$, where $\mathbf{p}_{\boldsymbol{x}}$ assigns 1 to the corresponding ranking position of this solution i.e. $p_{\sigma(\boldsymbol{x})} = 1$. According to this definition, the basins of attraction related to each solution generate a partition of $\Omega_{|S|}$ that can be expressed by the sets $\mathcal{Z}_1, \ldots, \mathcal{Z}_{|S|}$.

The sequences of probability vectors express the complete process of convergence, whereas the basins of attraction represent only the final convergence of the algorithm. From Definition 6.1, we can deduce that if functions σ_1 and σ_2 are equivalent, the algorithm generates the same basins of attraction $\mathcal{Z}_1, \ldots, \mathcal{Z}_{|S|}$ for both functions. Therefore, all the functions belonging to the same class have the same basins of attraction. However, two different classes could have the same basins of attraction although they always have different sequences of probability vectors. Informally speaking, we might say that, for functions in different classes, the algorithm can reach the same places although it uses different roads.

6.5 Case study: Univariate EDA

The concepts and definitions about the taxonomy of problems that were abstractly discussed in the previous sections will be studied in-depth in a simple EDA that assumes independence among the variables of the problem. Different algorithms such as population-based incremental learning (Baluja (1994)), compact genetic algorithm (Harik et al. (1998)) or univariate marginal distribution algorithm (Mühlenbein (1998)) introduce this type of model.

We consider that the function $\mathcal{M}$ approximates the distribution $p^s(\boldsymbol{x})$ as follows. The structure $\boldsymbol{s}$ implemented in $\mathcal{M}$ provides a factorization of the joint probability distribution which assumes that the variables of the problem are independent. Then, considering the set of parameters $\boldsymbol{\theta_s}$, the function $\mathcal{M}$ can be expressed in the following form (Larrañaga and Lozano (2002)),

$$p^a(\boldsymbol{x}|\boldsymbol{\theta_s}) = \prod_{i=1}^{n} p^s(x_i|\boldsymbol{\theta}_i) \tag{6.3}$$

where $\boldsymbol{\theta_s} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. The local parameters $\boldsymbol{\theta}_i = (\theta_i^0, \theta_i^1)$ specify the marginal probability distributions $p^s(x_i)$ where $\theta_i^{x_i}$ is the probability value $p^s(X_i = x_i)$. In EDAs with finite population, each parameter $\theta_i^{x_i}$ can be estimated from the selected individuals by using different approaches such as relative frequencies or more sophisticated update rules. Nevertheless, in the model of infinite population, the marginal probabilities can be exactly calculated as,

$$p^s(x_i) = \sum_{\boldsymbol{x}\setminus x_i} p^s(\boldsymbol{x}). \tag{6.4}$$

From now on, we will obviate the explicit reference to the parameters $\boldsymbol{\theta_s}$ in Equation 6.3. Specifically, the function $\mathcal{M}$ is implemented in algorithm $\mathcal{A}$ as,

$$p^a(\boldsymbol{x}) = \prod_{i=1}^{n} \sum_{\boldsymbol{x}\setminus x_i} p^s(\boldsymbol{x}). \tag{6.5}$$

6.5.1 Equivalence condition

The key to understand the relation between a given optimization problem and an EDA is the way in which the permutation σ is related to the probabilistic model $(\boldsymbol{s}, \boldsymbol{\theta_s})$ used to compute the approximation step $\mathcal{M}$. More specifically, we take into account how the ranking positions $\tau = \sigma(\boldsymbol{x})$ of the solutions are organized in the different marginal distributions that form the factorization. These positions τ are grouped by using different sets as described below. The calculation of each marginal probability $p^s(x_i)$ can be put into relation with the ranking of the solutions which are involved in it. Thus, by using the inverse function $\sigma^{-1}(\tau)$ we can rewrite Equation 6.4 as,

$$p^s(x_i) = \sum_{\boldsymbol{x} \setminus x_i} p^s(\boldsymbol{x}) = \sum_{\tau \in Q_i^{x_i}} p^s(\sigma^{-1}(\tau)) \tag{6.6}$$

where $Q_i^{x_i} = \{\tau : \sigma^{-1}(\tau) = (y_1, \ldots, y_n) \wedge y_i = x_i\}$ is the set of ranking positions corresponding to points $\boldsymbol{x} \in S$, whose probabilities have been used to calculate the marginal distribution. Note that the cardinality of any $Q_i^{x_i}$ is always 2^{n-1}. For each marginal probability, we have the set of ranking positions associated to $p(X_i = 0)$, denoted by Q_i^0, and the set associated to $p(X_i = 1)$, denoted by Q_i^1. Note that the sets $Q_i^{x_i}$ are intrinsically related to the parameters $\theta_i^{x_i}$. Then, we associate the set $O_i = \{Q_i^0, Q_i^1\}$ to the marginal distribution $p(x_i)$. Note that for all i, $Q_i^0 \cup Q_i^1 = \{1, \ldots, 2^n\}$ and $Q_i^0 \cap Q_i^1 = \emptyset$. In addition, all the sets $Q_i^{x_i}$ involved in the factorization have to be different. Finally, we define the set $G_\sigma = \{O_1, O_2, \ldots, O_n\}$ which includes all the information needed to link the function and the factorization. The ranking positions belonging to each subset depend on the function σ to which the algorithm is applied. In summary, the relationship between the probabilistic model and the function σ is expressed by the structure of sets represented in Fig. 6.2 (a). An illustrative example of the definitions mentioned above is presented in Fig. 6.2 (b). We can see that, by using Equation 6.6, the marginal probability $p(X_1 = 0)$ is calculated through the solutions with the rankings $\{2, 3, 5, 8\}$. In turn, $p(X_1 = 1)$ is associated to the set of rankings $\{1, 4, 6, 7\}$. Thus, we have the set $O_1 = \{\{2, 3, 5, 8\}, \{1, 4, 6, 7\}\}$ related to the marginal distribution $p(x_1)$. In this example, we also have the sets $O_2 = \{\{3, 4, 6, 8\}, \{1, 2, 5, 7\}\}$ associated to $p(x_2)$ and $O_3 = \{\{2, 4, 7, 8\}, \{1, 3, 5, 6\}\}$ for $p(x_3)$. Finally, we can create the corresponding set $G_\sigma = \{O_1, O_2, O_3\}$ associated to the factorization of the probability distribution and the function σ being optimized.

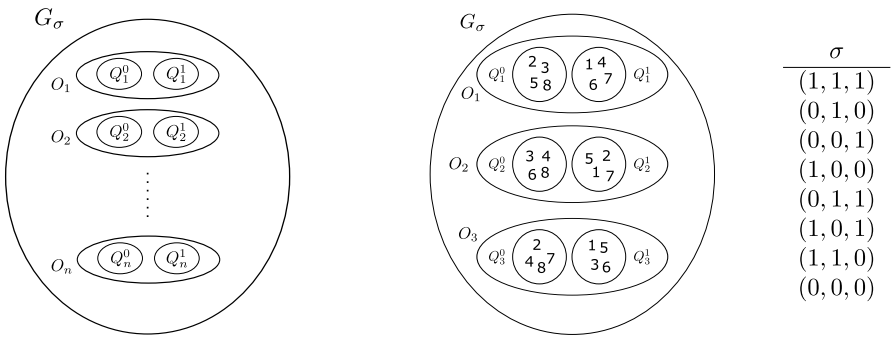


Fig. 6.2. (a) Set-based representation of the relation between the function and the probabilistic model. (b) Example of set G_σ and the corresponding function σ for $n = 3$ variables.

In general, we can prove the following necessary and sufficient condition of equivalence between objective functions when algorithm $\mathcal{A}$ implements the function $\mathcal{M}$ according to Equation 6.3. By using this condition, we will carry out the partition of the space of functions into equivalence classes.

Theorem 6.2 *Let $\mathcal{A}$ be an EDA that implements $\mathcal{M}$ as $p^a(\boldsymbol{x}) = \prod_{i=1}^{n} p^s(x_i)$. Two functions σ_1 and σ_2 are equivalent under $\mathcal{A}$ if and only if the corresponding sets G_{σ_1} and G_{σ_2} are equal.*

Proof. Two functions σ_1 and σ_2 are equivalent if $\mathcal{A}$ generates the same sequence of probability vectors departing from any initial $\mathbf{p}_0$. Therefore, we need to prove that $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$ in every generation t for all initial $\mathbf{p}_0$ if and only if $G_{\sigma_1} = G_{\sigma_2}$. Firstly, we show that if $G_{\sigma_1} = G_{\sigma_2}$, then $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$. In order to do that and taking into account that ϕ is independent of σ, it is enough to prove that $\mathcal{M}(\mathbf{p}^s, \sigma_1) = \mathcal{M}(\mathbf{p}^s, \sigma_2)$ for any given $\mathbf{p}^s$. $G_{\sigma_1} = G_{\sigma_2}$ if and only if for any $O_i^{\sigma_1} \in G_{\sigma_1}$, there exists $O_j^{\sigma_2} \in G_{\sigma_2}$ such that $O_i^{\sigma_1} = O_j^{\sigma_2}$ and vice versa. In turn, $O_i^{\sigma_1} = O_j^{\sigma_2}$ if an only if for any $Q_i^{x_i} \in O_i^{\sigma_1}$ there exists $Q_j^{x_j} \in O_j^{\sigma_2}$ such that $Q_i^{x_i} = Q_j^{x_j}$ and vice versa. Therefore, according to Equation 6.6, if $G_{\sigma_1} = G_{\sigma_2}$, then we are calculating the same set of probability values for both functions and we will obtain the same probability vector $\mathbf{p}^a$ in both cases.

Secondly, we prove that if $\mathcal{G}^t(\mathbf{p}_0, \sigma_1) = \mathcal{G}^t(\mathbf{p}_0, \sigma_2)$ in every generation t for all initial $\mathbf{p}_0$, then $G_{\sigma_1} = G_{\sigma_2}$. To prove this part of the theorem, it suffices to consider a specific set of initial probability vectors containing values greater than 0 only in the desired positions. Thus, let $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{2^n})$ be any initial vector such that $p_i > 0$ if $\sigma_1^{-1}(i) = (1, x_2, \ldots, x_n)$, otherwise $p_i = 0$. These initial vectors have probability values greater than 0 only in the positions associated to solutions that begin with 1 in σ_1. Due to Property 2 of ϕ, we obtain a vector $\mathbf{p}_0^s$ after selection with non-zero probabilities in the same positions as in $\mathbf{p}_0$. Then, we have that $p^s(X_1 = 1, t = 0) = 1$ for σ_1 after selection. Since we know that $\mathcal{G}^1(\mathbf{p}_0, \sigma_1) = \mathcal{G}^1(\mathbf{p}_0, \sigma_2)$, it necessarily implies that there exists $j \in \{1, \ldots, n\}$ such that $p^s(X_j = x_j, t = 0) = 1$ for σ_2 and therefore $O_1^{\sigma_1} = O_j^{\sigma_2}$. Otherwise, we would have that $p_i > 0$ for all i in $\mathcal{G}^1(\mathbf{p}_0, \sigma_2)$ and hence, the sequence would be different. By the same argument, any initial vector $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{2^n})$ satisfying $p_i > 0$ if $\sigma_1^{-1}(i) = (x_1, 1, \ldots, x_n)$ implies that there exists $k \neq j, k \in \{1, \ldots n\}$ such that $O_2^{\sigma_1} = O_k^{\sigma_2}$. The same process is repeated for the remaining indices until n, where we consider any initial point $\mathbf{p}_0 = (p_1, p_2, \ldots, p_{2^n})$ such that $p_i > 0$ if $\sigma_1^{-1}(i) = (x_1, x_2, \ldots, 1)$. Since we have already matched $n-1$ sets $O^{\sigma_1} \in G_{\sigma_1}$ with the corresponding $n-1$ sets $O^{\sigma_2} \in G_{\sigma_2}$, there remains a last index variable l such that $O_n^{\sigma_1} = O_l^{\sigma_2}$. Therefore, if the algorithm generates the same sequences of probability vectors from every initial point for σ_1 and σ_2, then $G_{\sigma_1} = G_{\sigma_2}$. □

6.5.2 Characterization of equivalence classes

Before addressing in detail the description of the functions belonging to a class, we show in Table 6.4 an example of three equivalent functions σ, σ' and σ'' . We have applied two different operations to obtain these equivalent functions. Firstly, we have generated the function σ' by negating the values x_1 for all $\boldsymbol{x} = (x_1, x_2, x_3)$ belonging to σ. Secondly, the function σ'' has been obtained by swapping the values x_2 and x_3 for all $\boldsymbol{x} = (x_1, x_2, x_3)$ belonging to σ'. This operation can be seen as a swapping of the columns x_2 and x_3. In Fig. 6.3, we present the corresponding sets G_σ, $G_{\sigma'}$ and $G_{\sigma''}$ related to the functions of Table 6.4. Since these sets are equal, we know by Theorem 6.2 that the functions are equivalent.

Rank	σ (x_1, x_2, x_3)	σ' $(\neg x_1, x_2, x_3)$	σ'' $(\neg x_1, x_3, x_2)$
1	(1,1,1)	(0,1,1)	(0,1,1)
2	(0,1,0)	(1,1,0)	(1,0,1)
3	(0,0,1)	(1,0,1)	(1,1,0)
4	(1,0,0)	(0,0,0)	(0,0,0)
5	(0,1,1)	(1,1,1)	(1,1,1)
6	(1,0,1)	(0,0,1)	(0,1,0)
7	(1,1,0)	(0,1,0)	(0,0,1)
8	(0,0,0)	(1,0,0)	(1,0,0)

Table 6.4. Equivalent functions σ, σ' and σ''.

In general, taking into account that σ is defined as a 2^n-tuple of the solutions $\boldsymbol{x} = (x_1, \ldots, x_n) \in S$, the following two operations permit the description of the functions in the class $[\sigma]$:

- Operator M1 (Negation). Given a permutation σ and an index $i \in \{1, \ldots, n\}$, the operator M1 returns a function σ' such that for all ranking positions $\tau \in \{1, \ldots, 2^n\}$ verifies $\sigma^{-1}(\tau) = (x_1, \ldots, x_n)$, $\sigma'^{-1}(\tau) = (y_1, \ldots, y_n)$ and $y_i = \neg x_i$. Through this operation we change zeros with ones, and vice versa, in the corresponding variable x_i. This operator can be successively applied until all combinations of variable negations are obtained. Thus, including σ in the count, we can generate 2^n different functions by means of M1.
- Operator M2 (Swapping). Given a permutation σ and two indexes $i, j \in \{1, \ldots, n\}$, the operator M2 returns a function σ' such that for all ranking positions $\tau \in \{1, \ldots, 2^n\}$ verifies $\sigma^{-1}(\tau) = (x_1, \ldots, x_n)$, $\sigma'^{-1}(\tau) = (y_1, \ldots, y_n)$, $y_i = x_j$ and $y_j = x_i$. If we consider the elements of σ disposed in columns, then we can say that the permutation of the columns associated to the variables X_i and X_j produces an equivalent function.

From this interpretation, it is easy to see that $n!$ different functions can be generated by applying M2.

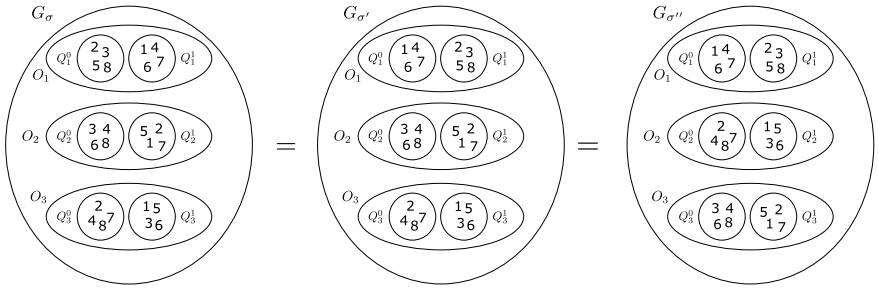


Fig. 6.3. Example with $n = 3$ variables of three equal sets G_σ, G'_σ and G''_σ associated to three equivalent functions σ, σ' and σ'' presented in Table 6.4.

According to Theorem 6.2, we can assert that, given a function σ, the aforementioned operators return equivalent functions σ' because $G_\sigma = G_{\sigma'}$. In fact, these two operators are a consequence of this equality of sets. We have previously shown how the set G_σ is created from σ according to the probabilistic model used by the algorithm. Inversely, we can also read the permutation σ from the set G_σ. Thus, a set $Q_i^{x_i}$ is telling us the positions τ of the permutation σ in which $X_i = x_i$ (check Fig. 6.2). If these positions are read from every set $Q_i^{x_i}$, then we will obtain σ. Note that if we modify the ranking positions belonging to $Q_i^{x_i}$, we are modifying the positions in which $X_i = x_i$ and therefore we will read a different permutation σ.

Specifically, departing from a given G_σ, we can move the ranking positions that this set contains in two different ways in order to read different functions σ' such that $G_\sigma = G_{\sigma'}$. The first type of movement is as follows. Given any $O_i = \{Q_i^0, Q_i^1\} \in G_\sigma$, the ranking positions $\tau_0 \in Q_i^0$ and $\tau_1 \in Q_i^1$ are swapped together between both subsets to create an equal set $G_{\sigma'}$ in which $\tau_1 \in Q_i^0$ and $\tau_0 \in Q_i^1$. From $G_{\sigma'}$, we can read a new function σ' equivalent to σ. In Fig. 6.3, this type of movement is applied to the set $O_1 \in G_\sigma$ obtaining an equal set $G_{\sigma'}$. From this movement, we deduce that the negation operator M1 produces equivalent functions.

In the second type of movement, given any pair $O_i = \{Q_i^0, Q_i^1\}, O_j = \{Q_j^0, Q_j^1\} \in G_\sigma$, we can exchange the ranking positions belonging to both sets O_i and O_j as follows. Let $\tau_i \in Q_i^{x_i}$ and $\tau_j \in Q_j^{x_j}$, we swap the elements between the sets $Q_i^{x_i}$ and $Q_j^{x_j}$ to create the set $G_{\sigma'}$ in which $\tau_j \in Q_i^{x_i}$ and $\tau_i \in Q_j^{x_j}$. All the ranking positions τ_i belonging to $O_i = \{Q_i^0, Q_i^1\}$ have to be moved at the same time, otherwise, the associated marginal probability will make no sense. In Fig. 6.3, this type of movement is applied to the sets O_2

and O_3 in $G_{\sigma'}$ producing an equal set $G_{\sigma''}$. From this movement we deduce that the swapping operator M2 produces equivalent functions.

Based on the above-mentioned movements that the equality of sets allows, it can be stated that, given σ, the operators M1 and M2 allow to describe all the functions in the class $[\sigma]$. There are no more operations that produce equivalent functions according to Theorem 6.2. Therefore, by combining all possible functions that can be generated by using M1 and M2, we can conclude that the number of equivalent functions σ per class is $n!2^n$. Therefore, the number of classes is

$$\frac{(2^n-1)!}{n!}.$$

This is the number of different behaviors that an univariate EDA can show in solving Problem 6.1.

6.5.3 Equivalence classes and local optima

The impact that different problem characteristics have in the performance of search algorithms is a fundamental topic in the optimization field. Properties such as the number of local optima, the additive decomposition of the function (Mühlenbein and Mahnig (1999b); Gao and Culberson (2005)) or many other difficulty measures (Naudts and Kallel (2000)) have been proposed and studied in order to advance the performance of evolutionary algorithms according to those descriptors of the problem. Particularly, in the field of EDAs, the relation between the local optima of the function and specific EDA implementations has been theoretically shown in González et al. (2001) and Zhang (2004). In this section, we connect this property of the problems with the equivalence classes developed along the chapter.

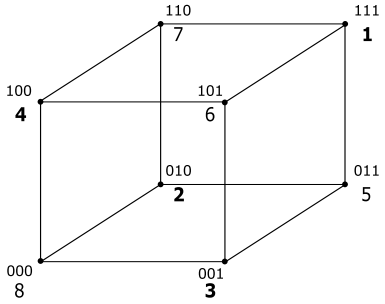
We consider the neighborhood system in S induced by the Hamming distance. Thus, the distance $H(\boldsymbol{x}, \boldsymbol{y})$ between two solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by,

$$H(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

The neighbors of a solution $\boldsymbol{x}$ are those solutions $\boldsymbol{y} \in S$ such that $H(\boldsymbol{x}, \boldsymbol{y}) = 1$. In terms of the permutation σ, a solution $\boldsymbol{x}$ is called a local optima if $\sigma(\boldsymbol{x}) < \sigma(\boldsymbol{y})$ for any $\boldsymbol{y} \in S$ such that $H(\boldsymbol{x}, \boldsymbol{y}) = 1$. Fig. 6.4 illustrates how the neighborhood system and the local optima can be seen for the given σ. Each vertex of the cube represents a solution $\boldsymbol{x} \in S$. Below each solution, we have the corresponding position of each $\boldsymbol{x}$ in σ. It can be checked that the first four solutions in the permutation are local optima because they are better solutions than their neighbors. These solutions are marked in bold.

The relation between the equivalence classes and the Hamming neighborhood system in S is established by the following theorem.

Theorem 6.3 *All the functions in the same equivalence class $[\sigma]$ have the same number of local optima and in the same ranking positions.*



σ
$(1,1,1)$
$(0,1,0)$
$(0,0,1)$
$(1,0,0)$
$(0,1,1)$
$(1,0,1)$
$(1,1,0)$
$(0,0,0)$

Fig. 6.4. Example of Hamming neighborhood system with $n = 3$ over the function σ. Numbers in bold represent local optima.

Proof. A local optimum can be defined according to the position that it holds in the permutation σ and the Hamming distance to the preceding solutions. Then, a solution $\sigma^{-1}(i) = \boldsymbol{x}$ is a local optima for σ if $H(\sigma^{-1}(i), \sigma^{-1}(j)) \geq 2$ for all $j < i$. Given a function σ, it is easy to see that by applying the operators M1 and M2 of negation and swapping, any function σ' that we obtain verifies $H(\sigma^{-1}(i), \sigma^{-1}(j)) = H(\sigma'^{-1}(i), \sigma'^{-1}(j))$ for all $i, j \in \{1, \ldots, 2^n\}$. Therefore, if there is a local optima in the position i of σ then there is a local optima in the same position i of σ' and vice versa. Since the operators M1 and M2 allow to describe all the functions of a class, these functions always have the same number of local optima and in the same ranking positions. □

Theorem 6.3 agrees with the results presented by González et al. (2001) and Zhang (2004). According to these works, all the local optima are attractive fixed points (Scheinerman (1996)) of the dynamical systems that were used to model the corresponding univariate EDA implementations. Therefore, since we say that the algorithm has the same behavior for all the functions belonging to a class, all those functions should necessarily have the same number of local optima and in the same ranking positions to support González et al. (2001) and Zhang (2004). Nevertheless, Theorem 6.3 could also provide a more general perspective because it implies that the relation between univariate EDAs and the local optima of the function is an intrinsic property of the probabilistic model and independent of the implementation of the selection scheme.

6.6 Numerical Experiments in $S = \{0, 1\}^3$

In this section, we use the previously elaborated partition of classes of the functions to carry out a more detailed analysis of the different behaviors the univariate EDA can have in the injective functions in $\{0, 1\}^3$. We conduct numerical simulations of an EDA with infinite population which imple-

ments tournament selection and computes the approximation step according to Equation 6.5. The local dynamical behavior of this algorithm was theoretically studied in Zhang (2004).

Particularly, we will concentrate on the complexity of the classes for the algorithm. The complexity is measured by two descriptors: i) the size of the basin of attraction of the global optimum, ii) the generated sequence of probability vectors. We will consider that the smaller the size of the basin of attraction in a class, the more difficult the problems in that class are. Moreover, when two classes have the same basin size, then the more time the algorithm takes to converge, the more difficult the function is.

In order to add more information to this analysis, and taking into account the relevant role that the local optima play in the EDA that assumes independence (see Theorem 6.3), we will put the previous complexity results in relation to this problem characteristic. We will see that the complexity for the EDA in terms of the size of the basin of attraction is closely related with the number of local optima and their positions in the function ranking. To the best of our knowledge, this is the first time that such an analysis is done in the literature.

6.6.1 Experimental design

We take into account all the possible σ that can be constructed over the search space $S = \{0,1\}^3$. Therefore, we consider $2^3! = 40320$ functions. By creating the set G_σ for each function and applying Theorem 6.2 for each pair of sets, we group the functions by equivalence classes. We have $3!2^3 = 48$ functions per class and hence 840 classes. We only need to consider one function per class because all the functions in the same class behave equivalently. The selection of the function which represents the class is arbitrary.

To carry out the EDA simulations, we need to specify four elements: the initial points, the selection mechanism, the approximation step (Equation 6.5), and the stopping condition. We create 10000 initial probability vectors which try to be representative of the simplex Ω_8. These initial points have been randomly generated by sampling a Dirichlet distribution with all the parameters equal to 1. In addition, we also take into account the uniform distribution as an initial point. Then, for each function, we launch 10001 EDA runs one from each initial probability vector previously generated. All these runs try to represent the EDA behavior for the corresponding optimization problem.

We use two-tournament selection according to Zhang (2004) to implement the selection ϕ. This selection takes uniformly at random two solutions of the population and then chooses the individual with best objective function value. This procedure should be repeated until the selected set is completed. Since we deal with injective functions, two solutions can not have the same function value. In the infinite population EDA model, the probability vector $\mathbf{p}^s$ after tournament selection can be computed from the vector $\mathbf{p}$ as follows:

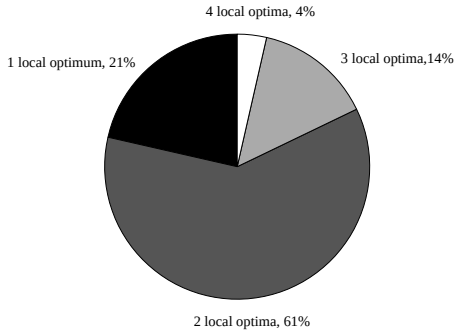


Fig. 6.5. Proportion of classes with different number of local optima.

$$p_i^s = p_i^2 + 2p_i \sum_{j=i+1}^{n} p_j. \tag{6.7}$$

Tournament selection obeys the properties that we imposed to ϕ in Section 6.3.2. Therefore, the partition induced from the equivalence condition of Theorem 6.2 is valid.

The stopping condition of the algorithm is a maximum of 40 iterations. This number of generations provides a satisfactory trade-off between accuracy in the numerical results and computational cost. The algorithm converges to 1 in 93% of all the runs conducted. In the rest of the runs, the highest probability value after 40 generations is always greater than 0.9998. In these cases, it is assumed that the algorithm has converged to the corresponding solution. The numerical precision that we have used is double-precision floating point requiring 64 bits per stored value.

In the numerical analysis, the size of the basins of attraction are stored in a vector $\mathbf{b} = (b_1, \ldots, b_8)$ where each b_i is the number of initial points that have converged to the solution with rank i.

6.6.2 Results

First of all, in Fig. 6.5, we show the proportion of functions with different number of local optima in order to provide a general perspective of this problem characteristic. By considering classes instead of specific functions, we have 180 classes with just one local optimum (the global optimum), 510 classes with two local optima, 120 classes with three local optima and finally 30 classes with four local optima.

6.6.2.1 Analysis based on basins of attraction

In order to provide a first general picture of the equivalence classes, we represent in Fig. 6.6 the basins of attraction of the optimum by means of different

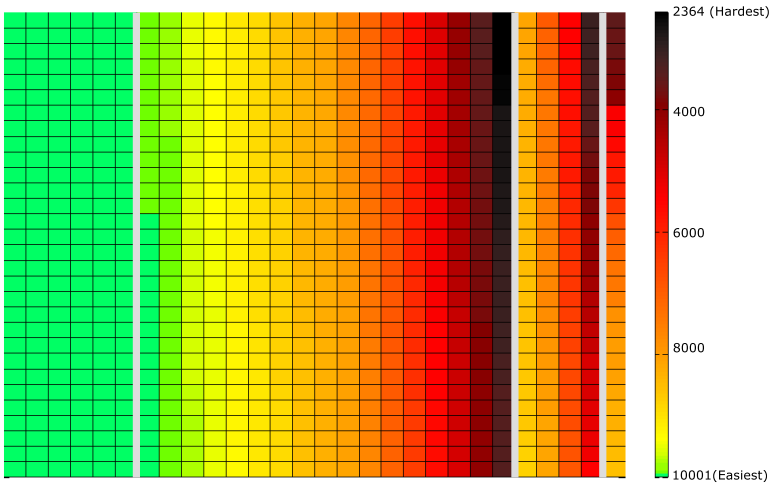


Fig. 6.6. Size of the basin of attraction of the optimal solution for each class.

colors. The sizes of these basins of attraction are interpreted in terms of problem difficulty. The color bar on the right of this picture indicates the relation between the colors and the size of the basin. At the bottom of the spectrum, the green color is assigned to the largest basins of attraction which indicate easy problems. At the top, the dark colors represent low basins and hence, they reveal the hardest problems. In addition, the classes have been grouped by the number of local optima. Thus, the picture has been divided into four parts separated by vertical gray lines. From left to right, we have the classes with one, two, three and four local optima respectively. In each of these parts, the classes are ordered according to the size of the basins of attraction.

Note that we have assigned the green color only to the classes in which the size of the basins is 10001 or it is a very close number to that. We start to use yellow colors when approximately 150 initial points do not converge to the optimum. We try to highlight all the small variations between classes because they could imply dramatic differences in EDAs with finite populations and problems with greater dimension. According to Fig. 6.6, we could say that the green classes are easy. These green classes cover all the problems with one local optima and a small number of problems with two local optima. It can be observed that a higher number of local optima does not necessarily imply more difficult problems. In fact, the darkest colors are in the area corresponding to classes with two local optima. It is the zone in which we can see the widest range of colors.

As discussed in Section 6.4, different classes can have the same basins of attraction. Thus, Fig. 6.6 shows how the classes can be grouped according to these basins of attraction. This fact could be related to the existence of a second level of grouping among classes. Nevertheless, it deserves a specific study.

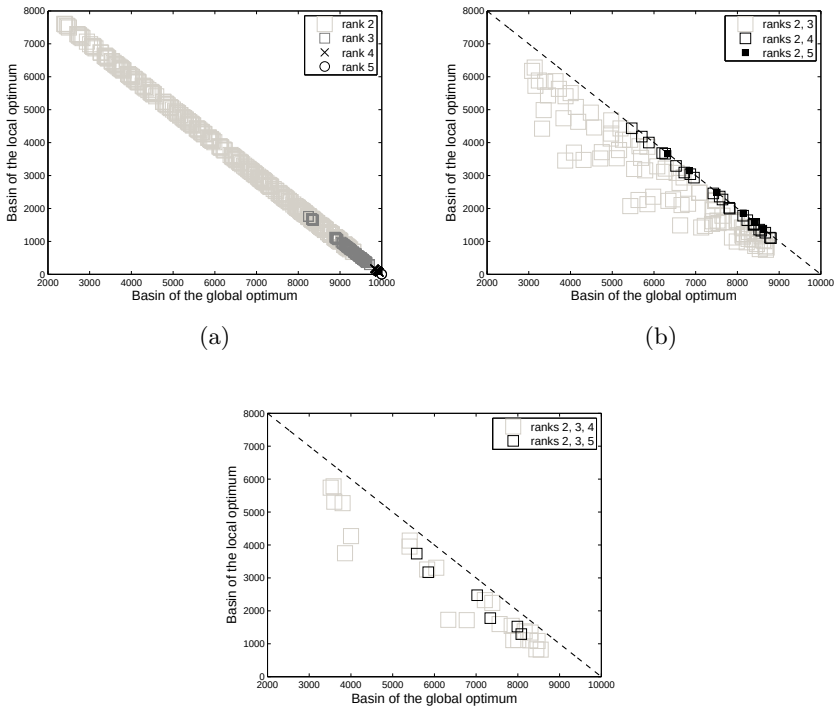


Fig. 6.7. Basins of attraction of the optimal solution (x-axis) and the best local optimum (y-axis). (a) Classes with one local optimum besides the global optimum. As indicated in the legend, the local optimum can hold the ranks 2, 3, 4 or 5 in a permutation σ. (b) Classes with two local optima besides the global optimum. The two local optima can be at the following ranking positions: $(2, 3)$, $(2, 4)$ and $(2, 5)$. (c) Classes with three local optima besides the global optimum. The three local optima can be at the following ranking positions: $(2, 3, 4)$ and $(2, 3, 5)$. The dashed line ($y = 10001 - x$) serves as reference to indicate the basins of attraction of the rest of local optima.

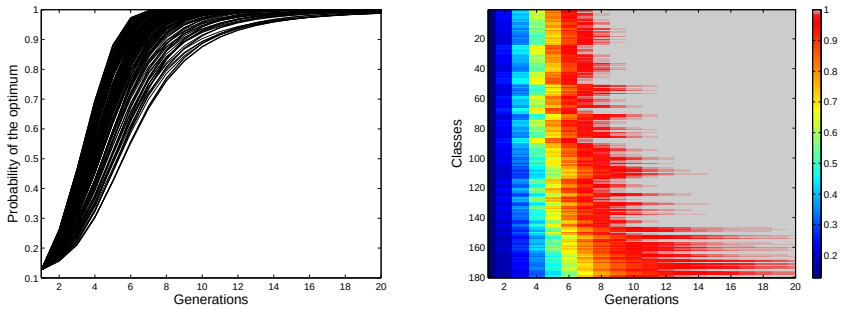
Through Fig. 6.7, we analyze the basins of attraction of both the global optimum and the rest of local optima. We consider all the ranking positions at which the local optima can be allocated. In this case, when we refer to

local optima, the global optimum is not included. The different scenarios are indicated in the legends and expressed by means of different markers. Note that in these plots only the basins of the global optimum and the best local optimum are explicitly indicated. The dashed line $y = 10001 - x$ is used as reference to illustrate the proportion of initial points that have converged to the rest of local optima.

In Fig. 6.7(a), we show the classes with one local optimum. This solution can be in different positions of the permutation σ as indicated in the figure. Note that the difficulty of the problem strongly depends on the ranking position of the local optimum. When this solution changes from the second position to the third position of the ranking, the difficulty of the problem decreases dramatically. In fact, when the local optimum has rank 5, the complexity of the problems is very similar to the complexity of a problem without local optima. Fig. 6.7(b) and 6.7(c) show the classes with two and three local optima respectively. We can see the frequency in which the algorithm can reach the different local optima depending on the class. Analogously to the previous situation, as the local optima have a lower rank, their basins of attraction clearly decrease. Particularly, when the rank of the worst local optimum changes from 3 to 4 in Fig. 6.7(b), the difficulty of the problems dramatically decreases. In Fig. 6.7(c) the changes are more subtle. Nevertheless, we can observe that the classes with a local optimum in the fifth position never reach the highest complexities of the classes with a local optimum in the fourth position.

6.6.2.2 Analysis based on sequences

We know that the sequences of probability vectors generated by the algorithm uniquely identify the functions in a class. In this section, we use this fact in order to distinguish the different EDA behaviors for the classes with only one local optimum (the global optimum). According to the basins of attraction (Fig. 6.6), all these classes have the same complexity for the algorithm. However, we can observe in Fig. 6.8 different convergence behaviors. In Fig. 6.8(a), we show the curves that the probability of the optimum depicts throughout the generations. Alternatively, Fig. 6.8(b) represents the same probabilities of the optimum by means of colors. The specific probability values are shown in the color bar on the right. Particularly in Fig. 6.8(b), we can clearly see how several classes on the bottom of the chart have a slower convergence. This slower convergence to the optimum can also be interpreted as a consequence of facing harder problems. From this point of view, we could say that not all the functions with one local optima have the same complexity. This fact could have implications in EDAs with finite samples.



(a) (b)

Fig. 6.8. Probability of the optimum along the generations for the classes with only one local optimum. (a) Curves of the probability of the optimum. (b) Probability of the optimum represented by means of colors for each class. The specific probability values are shown in the color bar.

6.7 Conclusions

This chapter can be divided into three main parts. Firstly, we have laid the foundations to create taxonomies of problems under EDAs by providing the needed definitions regarding the optimization problems, the algorithm and the equivalence relation. From these definitions, it has been deduced that all the problems are in the same class when the probabilistic model does not impose restrictions to approximate the distribution of the selected individuals. Secondly, we have studied the taxonomy of problems that arises under an univariate EDA. To express the relation between the probabilistic model and the function, we have defined the sets G_σ. Based on these sets, we are able to provide a necessary and sufficient condition to decide the equivalence between functions and to partition the space of problems. Through the operators of negation and swapping, it is possible to describe all the functions in a class and count its members. By taking into account the aforementioned elements, we reveal an intrinsic connection between the univariate probabilistic model and the neighborhood system induced by Hamming distance. In the third and last part, we have conducted numerical simulations of an univariate EDA which implements tournament selection. We can extract the following main conclusions from the experiments. i) A higher number of local optima does not necessarily imply more difficult problems. In this regard, the difficulty of the problem strongly depends on the ranking position of the local optima. ii) We have observed how the classes can be grouped according to the basins of attraction showing a second level of grouping. iii) We have shown that not all the functions without local optima have the same complexity.

In general terms, this chapter introduces a framework which allows to formally study the relationship between EDAs and the space of optimization

problems. The results that we have presented can be generalized and extended in many directions. Specifically, once the partition of the space of problems has been created, we consider the following questions particularly important: how to describe and identify the classes of easy and hard problems for EDAs? (Chen et al. (2010)), which are the problem descriptors that allow to identify the class to which that problem belongs to?, how to study the convergence of the algorithm (Zhang (2004); Zhang and Mühlenbein (2004)) for the problems in a class?, which is the minimum complexity that should be introduced in the probabilistic model in order to converge to the optimum? (Echegoyen et al. (2012)), how to extrapolate these results to algorithms that use finite samples?. This type of issues could be translated to other evolutionary algorithms. We believe that working in the direction given by these questions is important to reach an in-depth understanding of the underlying mechanisms that govern evolutionary algorithms.

7

Conclusion

7.1 Introduction

The following section gathers the general conclusions obtained throughout the different contributions. More specific conclusions were presented at the end of the corresponding chapters. Section 7.3 discusses some outstanding open research trends and poses possible lines for future work. Finally, Section 7.4 presents the publications that result from work in this dissertation.

7.2 Conclusions

In essence, this thesis has been devoted to increase our comprehension about EDAs. Although we have worked in discrete domains and all the probabilistic models that we have considered can be expressed by means of Bayesian networks, many of the concepts and ideas that have been discussed can be extrapolated to other kinds of EDAs or even to other evolutionary algorithms.

All the contributions of the dissertation have been mainly developed by paying attention to three basic elements. Firstly, of course, the optimization problems, which have been described through different characteristics such as the structure, the multimodality or the local optima. Secondly, the learning step of the EDA, which determines the probabilistic models used by the algorithm, and thirdly, the population. It is well known that EDAs and other evolutionary algorithms have different parameters and can be mixed with additional techniques with the aim of improving their performance. However, the aforementioned three elements, together with the selection step, are the essence of this type of algorithms and hence, they are the corner stone to understand their behavior in a broad sense, regardless of specific implementations.

The main topics and conclusions of this thesis are summarized in the following paragraphs.

The relationship between the structure that the interactions of the problem variables provide and the structural models learned by the algorithm has been a recurrent issue throughout the dissertation. In this regard, we have seen that the structures that the algorithm learns during the search provide valuable information about the interdependences among the variables of the problem. This fact has been observed in other related works and it is considered as a distinctive feature of EDAs compared with other types of evolutionary algorithms. However, it has also been noticed that introducing a learning method that obtains the best Bayesian networks at each generation does not necessarily improve the performance of the algorithm. Nevertheless, with enough population size, this type of algorithm is able to obtain structures that provide much more information about the problem than the approximate learning.

We have also seen that the topology of the problem structure influences both the difficulty of the problem and the models learned by the algorithm. By considering a range of problems with the same number of explicit interactions between variables, we have shown that the problems with a regular grid-like structure seem to be easier for EDAs. However, rewiring a few number of interactions is enough to increase the complexity of the problem and due to this, the algorithm needs to learn more dependences related to the problem structure in order to reach the optimum. The structure of this type of problems has been related to networks that emerge in natural systems, called small-world networks. These results bring to mind the hypothesis of Kauffman (1993) in which it is said that complex systems in nature achieve a state which optimizes the complexity of the tasks the system can perform and simultaneously optimizes evolvablity. Roughly speaking, evolvability can be understood as the ability of a population of organisms to evolve through natural selection.

The ability of EDAs to capture the structure of the problem has been shown and analyzed. However, where the limits of the algorithm are in order to do that is an open issue. We have seen that, in the worst-case scenario, EDAs encounter important limitations as the number of interactions among the variables of the problem increases. In fact, after a certain degree of interaction, the performance of the EDA drops drastically even using unrestricted Bayesian networks. Although learning structural models clearly improves the behavior of the algorithm, the relationship between these structures and the exact factorizations associated to the functions suggests different sources for the limits of the EDA performance. On the one hand, the results suggest limits due to the use of approximate learning techniques. On the other hand, even by considering an ideal or exact learning method, the limits could be due to the computational cost of managing the increasing structural complexity that seem to be needed to solve the problems. Improving our knowledge about the limits of EDAs and other search algorithms, form the basis to start to conceive a framework in which, based on that knowledge, we could select the most appropriate algorithm depending on the problem at hand.

When the algorithm is studied from the perspective of the probability of the optimum and the most probable solution, novel insights can be provided. The main elements of the algorithm that we have considered, which are the structural model and the population size, clearly influence the probability of the optimum and the most probable solution. Moreover, the patterns of behavior are constant in every optimization problem analyzed. For instance, using an adequate population size or an accurate structural model increases the probability of the optimum during the search in relation to the most probable solution, even in runs where the optimum is not reached. In addition, the function values of the most probable solution also reflect the influence of the population size and the structural model accuracy. The properties of the problem at hand, such as the multimodality, or even the difficulty that it entails for the algorithm, are reflected in this type of analysis. The experimental framework designed is not only useful to better understand EDAs but also to devise new improvements of the algorithm.

Finally, after studying EDAs through different approaches, we have conducted a formal attempt to describe the relationship that emerges between EDAs and the space of optimization problems. We have laid the foundations to elaborate taxonomies of problems under EDAs. Thus, given an EDA, the space of problems can be partitioned in equivalence classes representing the different behaviors that the algorithm can exhibit. The classes can be connected with characteristics of the problems belonging to them. Through this taxonomy, we have shown an underlying connection between the neighborhood system induced by the Hamming distance and univariate EDAs. Moreover, we have analyzed numerically the difficulty of the problems belonging to each class and the relation of that difficulty with the number of local optima. For instance, we have shown that a higher number of local optima does not necessarily imply more difficult problems or that not all the functions without local optima necessarily have the same complexity. In general, this contribution opens new research lines since new and more general results can be deduced from the definitions and concepts presented. In addition, exploring the relation that emerges between a search algorithm and the space of optimization problems can entail a better understanding of the limits of effective application of that algorithm. Therefore, the development of the equivalence classes could be thought from the perspective of the learning limits and both roads could be connected.

7.3 Future work

The contributions of this dissertation have left different lines of work open. Some of these lines are related with ongoing research trends whereas other topics can constitute new subjects of research. We believe that the following points are the most interesting for further study.

- As previously commented, the relationship between the structure of the problem and the structural models used by EDAs is a recurrent issue throughout the thesis. In this regard, different adjectives such as benign, malign, strong or deceptive have been used to describe the interactions among the variables of the problem and then, study their effect both in EDAs and other evolutionary algorithms. Although some attempts to formalize this type of concepts have been presented (Kallel et al. (2000); Ochoa and Soto (2006); Santana et al. (2005)), we clearly need to conduct more research in order to understand and specify all the aforementioned terms in the context of optimization by means of EDAs.
- The concepts discussed in the previous point are closely related to the difficulty of the problems. We have studied how the topology of the problem structure determines the difficulty of the problems for EDAs. However, the role that the function values of the sub-functions play in the complexity of the problem is a question that remains open. Note that considering these function values also implies taking into account the concepts mentioned above regarding the properties of the interactions among the variables of the problem. In general, we believe that there is a lack of formalism and understanding when dealing with the interactions among the problem variables, their relation with the structural model and their impact in the difficulty of the problem. Continuing research in this direction is a fundamental task to better understand the relationship between the structure of the problem and the probabilistic models that the algorithm needs to reach the optimum.
- Regarding the limits of effectiveness in EDAs, a more in-depth study should be carried out in order to increase the soundness of the conclusions. Thus, more accurate learning techniques, more sophisticated EDAs aided by niching or local searches, or even other approaches such as mixtures of evolutionary algorithms, should be tested under the same worst-case scenario. Then, analyzing the levels of problem difficulty that this type of algorithms successfully reaches, would be useful to better understand both the learning limits of EDAs and the limits of other search techniques. To complement the results obtained by using functions based on deceptive sub-functions, similar experiments could be conducted with other classes of functions such as Max-SAT or Ising. The role of the population in the limits of effectiveness of the algorithm was also discussed. We argue that a given population size can only contain useful information to solve problems to a certain degree of interaction among their variables. However, studies related with the information that the populations contain about the problem have hardly been considered. We believe that the formalization and study of this notion would be worthwhile.
- The taxonomy of problems presented in the previous chapter opens new research lines. First of all, some generalization such as the introduction of non-injective functions and general Bayesian networks could be developed. In addition, providing the needed definitions to deal with any type

of selection scheme could also be considered. Other important extensions are related to the connection between the characteristics of the problems and the equivalence classes to which they belong. We have shown the connection of the classes with the neighborhood system induced by the Hamming distance for univariate EDAs. This connection can be studied for more complex probabilistic models. For example, preliminary results indicate that, if we add an arc to the univariate model, then it is possible to include functions with one and two local optima in the same class. This implies that some functions with two local optima can entail the same difficulty as functions with one local optimum (the global optimum). This agrees with Zhang (2004), where it is said that using higher order statistics could improve the chance of finding the global optimum. Moreover, we hypothesize that it is possible to discover new links with other problem characteristics or descriptors. For instance, we have very preliminary results regarding the additive decomposition of the functions and its relationship with the equivalence classes. In turn, the classes could also be tagged in terms of the difficulty of the problems they contain. In an ideal scenario, the information available about the problem at hand could be used to identify the class to which it belongs to and then try to advance, for example, whether for a given factorization the algorithm will reach the optimum. In fact, knowing if a determined factorization will converge to the optimum for a given function is one of the most important issues in EDAs.

7.4 Contributions

The work carried out in this dissertation has produced the following publications and submissions:

A. Refereed journals

- C. Echegoyen, A. Mendiburu, R. Santana and J. A. Lozano. On the Taxonomy of Optimization Problems under Estimation of Distribution Algorithms. *Evolutionary Computation*. Submitted, 2012.
- C. Echegoyen, A. Mendiburu, R. Santana and J. A. Lozano. Towards Understanding EDAs Based on Bayesian Networks Through a Quantitative Analysis. *IEEE Transactions on Evolutionary Computation*. Vol. 16, No. 2, Pp. 173-189, 2012.
- R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armañanzas, S. Shakya. MATEDA-2.0: Estimation of Distribution Algorithms in MATLAB. *Journal of Statistical Software*. Vol. 35, No. 7, Pp. 1-30, 2010.

B. Book chapters

- C. Echegoyen, A. Mendiburu, R. Santana and J.A. Lozano. Analyzing the k Most Probable Solutions in EDAs Based on Bayesian Networks. *Exploitation of linkage learning in evolutionary algorithms*. Evolutionary Learning and Optimization. Springer. Y.-P. Chen editor. Pp. 163-189, 2010.
- C. Echegoyen, R. Santana, J.A. Lozano and P. Larrañaga. The Impact of Exact Probabilistic Learning Algorithms in EDAs Based on Bayesian Networks. *Linkage in Evolutionary Algorithms*. Studies in Computational Intelligence Series. Springer. Y.-P. Chen and M.-H. Lim editors. Pp. 109-139, 2008

C. Conference communications

- C. Echegoyen, A. Mendiburu, R. Santana and J. A. Lozano. Clases de Equivalencia en Algoritmos de Estimación de Distribuciones. *VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB-2012)*, Albacete, Spain. 2012. (Best student paper award. Best methodological paper nomination).
- C. Echegoyen, Q. Zhang, A. Mendiburu, R. Santana, J.A. Lozano. On the limits of effectiveness in estimation of distribution algorithms. In *Proceedings of the 2011 Congress on Evolutionary Computation (CEC-2011)*, New Orleans, USA. IEEE Press. Pp. 1573-1580, 2011. (Best student paper award).
- C. Echegoyen, A. Mendiburu, R. Santana and J.A. Lozano. Estimation of Bayesian networks algorithms in a class of complex networks. In *Proceedings of the 2010 Congress on Evolutionary Computation (CEC-2010)*, Barcelone, Spain. IEEE Press. Pp. 2154-2164, 2010.
- C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano. Analyzing the probability of the optimum in EDAs based on Bayesian networks. In *Proceedings of the 2009 Congress on Evolutionary Computation (CEC-2009)*, Trondheim, Norway. Pp. 1652-1659, 2009. (Best student paper award).
- C. Echegoyen, A. Mendiburu, R. Santana, J.A. Lozano. Estudio de la probabilidad del óptimo en EDAs basados en redes Bayesianas. *VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB-2009)*, Málaga, Spain. Pp. 31-38, 2009.
- C. Echegoyen, R. Santana, J.A. Lozano and P. Larrañaga. Exact Bayesian network learning in estimation of distribution algorithms. In *Proceedings of the 2007 Congress on Evolutionary Computation (CEC-2007)*, Singapore. IEEE Press. Pp. 1051-1058, 2007.
- C. Echegoyen, R. Santana, J.A. Lozano and P. Larrañaga. Aprendizaje exacto de redes Bayesianas en algoritmos de estimación de distribuciones. *Jornadas de Algoritmos Evolutivos y Metaheurísticas (JAEM I)*. Zaragoza, Spain. Pp. 277-284, 2007.

D. Technical reports

- C. Echegoyen, Q. Zhang, A. Mendiburu, R. Santana, J.A. Lozano. Analyzing limits of effectiveness in different implementations of estimation of

distribution algorithms. Technical Report EHU-KZAA-TR-2-2011. University of the Basque Country, Department of Computer Science and Artificial Intelligence, January 2011.

- C. Echegoyen, A. Mendiburu, R. Santana, J.A. Lozano. A quantitative analysis of estimation of distribution algorithms based on Bayesian networks. Technical Report EHU-KZAA-TR-2-2009. University of the Basque Country, Department of Computer Science and Artificial Intelligence, Septiembre 2009.
- R. Santana, C. Echegoyen, A. Mendiburu, C. Bielza, J. A. Lozano, P. Larrañaga, R. Armañanzas and S. Shakya. MATEDA: A suite of EDA programs in Matlab. Technical Report EHU-KZAA-IK-2/09. University of the Basque Country, Department of Computer Science and Artificial Intelligence, February 2009.

References

Akaike, H. (1974). A new look at the statistical identification model. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97.

Albert, R., Jeong, H., and Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Nature*, 406:378–382.

Ali, W. and Topchy, A. P. (2004). Memetic optimization of video chain designs. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pages 869–882, Seattle, WA, USA. Springer.

Armañanzas, R. (2009). *Consensus policies to solve bioinformatic problems through Bayesian network classifiers and estimation of distribution algorithms*. PhD thesis, The University of the Basque Country (UPV/EHU), Spain.

Armañanzas, R., Inza, I., Santana, R., Saeys, Y., Flores, J. L., Lozano, J. A., Van de Peer, Y., Blanco, R., Robles, V., Bielza, C., and Larrañaga, P. (2008). A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6):1–12.

Armañanzas, R., Saeys, Y., Inza, I., Garca-Torres, M., Bielza, C., van de Peer, Y., and Larrañaga, P. (2011). Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):760–774.

Arnborg, S., Corneil, D. G., and Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA.

Baluja, S. (2006). Incorporating a priori knowledge in probabilistic-model based optimization. In Pelikan, M., Sastry, K., and Cantú-Paz, E., editors,

Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Studies in Computational Intelligence, pages 205–222. Springer-Verlag.

Baluja, S. and 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*, pages 30–38. Morgan Kaufmann.

Barahona, F. (1982). On the computational complexity of Ising spin glass model. *Journal of Physics A: Mathematical and General*, 15(10).

Bengoetxea, E. (2003). *Inexact Graph Matching Using Estimation of Distribution Algorithms*. PhD thesis, Ecole Nationale Supérieure des Télécommunications.

Bengoetxea, E. and Larrañaga, P. (2010). EDA-PSO: a hybrid paradigm combining estimation of distribution algorithms and particle swarm optimization. In *Proceedings of the 7th International Conference on Swarm Intelligence*, pages 416–423, Berlin, Heidelberg. Springer-Verlag.

Bengoetxea, E., Miquélez, T., Larrañaga, P., and Lozano, J. A. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter Experimental results in function optimization with EDAs in continuous domain, pages 177–190. Kluwer Academic Publishers, Boston/Dordrecht/London.

Blanco, R., Inza, I., and Larrañaga, P. (2003). Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18(2):205–220.

Blanco, R., Larrañaga, P., Inza, I., and Sierra, B. (2001). Selection of highly accurate genes for cancer classification by estimation of distribution algorithms. In *Proceedings of the Workshop Bayesian Models in Medicine held within (AIME-2001)*, pages 29–34.

Blickle, T. and Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394.

Bosman, P. A. (2003). *Design and Application of Iterated Density-Estimation Evolutionary Algorithms*. PhD thesis, Universiteit Utrecht, Utrecht, The Netherlands.

Bosman, P. A. (2010). The Anticipated Mean Shift And Cluster Registration In Mixture-Based EDAs For Multi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2010)*, pages 351 – 358. ACM Press.

Bosman, P. A. and Grahl, J. (2007). Matching inductive search bias and problem structure in continuous estimation of distribution algorithms. *European Journal of Operational Research*, 185:1246–1264.

Brownlee, A., McCall, J., and Brown, D. (2007). Solving the MAXSAT problem using a multivariate EDA based on Markov networks. In *Proceedings of the 2007 conference on Genetic and Evolutionary Computation (GECCO-2007)*, pages 2423–2428, London, England. ACM.

Brownlee, A., McCall, J., Shakya, S., and Zhang, Q. (2009). Structure learning and optimisation in a Markov-network based estimation of distribution algorithm. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation (CEC-2009)*, pages 447–454, Piscataway, NJ, USA. IEEE Press.

Brownlee, A., McCall, J., and Shakya, S. K. (2012). *Markov Networks in Evolutionary Computation*, chapter The Markov Network Fitness Model, pages 125–140. Springer. In press.

Brownlee, A., McCall, J., Zhang, Q., and 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)*, pages 2621–2628, Hong Kong. IEEE Press.

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60.

Castillo, E., Gutierrez, J. M., and Hadi, A. S. (1997). *Expert Systems and Probabilistic Network Models*. Springer-Verlag.

Ceberio, J., Irurozki, E., Mendiburu, A., and Lozano, J. A. (2011). A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*. In press.

Chen, B., Li, L., and Hu, J. (2009). A novel EDAs based method for HP model protein folding. In *Proceedings of the 2009 Congress on Evolutionary Computation (CEC-2009)*, pages 309–315, Norway. IEEE Press.

Chen, S.-C. and Yu, T.-L. (2009). Difficulty of linkage learning in estimation of distribution algorithms. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO-2009)*, pages 397–404, New York, NY, USA. ACM Press.

Chen, T., Tang, K., Chen, G., and Yao, X. (2010). Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):1–22.

Chickering, D. M., Geiger, D., and Heckerman, D. (1995). Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128.

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

Coffin, D. J. and Smith, R. E. (2007). The limitations of distribution sampling for linkage learning. In *Proceedings of the 2007 Congress on Evolutionary Computation (CEC-2007)*, pages 364–369, Singapore. IEEE Press.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using belief networks. *Artificial Intelligence*, 42:393–405.

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

Cotta, C. (2003). Protein structure prediction using evolutionary algorithms hybridized with backtracking. In Mira, J. and Alvarez, J. R., editors, *Artificial Neural Nets Problem Solving Methods*, volume 2687 of *Lecture Notes in Computer Science*, pages 321–328. Springer Verlag.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic networks and expert systems*. Statistics for Engineering and Information Science. Springer-Verlag, New York.

Cuesta-Infante, A., Santana, R., Bielza, C., naga, P. L., and Hidalgo, J. I. (2010). Bivariate empirical and n-variate archimedean copulas in estimation of distribution algorithms. In *Proceedings of the 2010 Congress on Evolutionary Computation (CEC-2010)*, Barcelone, Spain. IEEE Press.

Cutello, V., Nicosia, G., Pavone, M., and Timmis, J. (2007). An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation*, 11(1):101–117.

Darwin, C. (1859). *The Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*.

Dawid, A. P. (1979). Conditional independence in statistical theory. *Journal of the Royal Statistical Society Series B*, 41:1–31.

Dawid, A. P. (1980). Conditional independence for statistical operations. *Annals of Statistics*, 8(3):598–617.

Dawid, A. P. (1997). Conditional independence.

De Bonet, J. S., Isbell, C. L., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9, pages 424–430. The MIT Press, Cambridge.

de Campos, L. M., Fernández-Luna, J. M., Gámez, J. A., and Puerta, J. M. (2002). Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311.

Deb, K. and Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10:385–408.

Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85.

Dill, K. A. (1985). Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509.

Dong, W. and Yao, X. (2008). NichingEDA: Utilizing the diversity inside a population of EDAs for continuous optimization. In *Proceedings of the 2008 Congress on Evolutionary Computation (CEC-2008)*, pages 1260–1267, Hong Kong. IEEE Press.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT press.

Dorogovtsev, S., Goltsev, A., and Mendes, J. (2008). Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275–1335.

Echegoyen, C., Lozano, J. A., Santana, R., and Larrañaga, P. (2007). Exact Bayesian network learning in estimation of distribution algorithms. In *Proceedings of the 2007 Congress on Evolutionary Computation (CEC-2007)*, pages 1051–1058. IEEE Press.

Echegoyen, C., Mendiburu, A., Santana, R., and Lozano, J. A. (2009). Analyzing the probability of the optimum in EDAs based on Bayesian networks. In *Proceedings of the 2009 Congress on Evolutionary Computation (CEC-2009)*, pages 1652–1659, Trondheim, Norway. IEEE Press.

Echegoyen, C., Mendiburu, A., Santana, R., and Lozano, J. A. (2012). Towards Understanding EDAs Based on Bayesian Networks Through a Quantitative Analysis. *IEEE Transactions on Evolutionary Computation*, 16(2):173–189.

Echegoyen, C., Santana, R., Lozano, J. A., and Larrañaga, P. (2008). *Linkage in Evolutionary Computation*, chapter The Impact of Exact Probabilistic Learning Algorithms in EDAs Based on Bayesian Networks, pages 109–139. Springer Berlin / Heidelberg.

Echegoyen, C., Zhang, Q., Mendiburu, A., Santana, R., and Lozano, J. A. (2011a). Analyzing limits of effectiveness in different implementations of estimation of distribution algorithms. Technical Report EHU-KZAA-TR-2/2011, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

Echegoyen, C., Zhang, Q., Mendiburu, A., Santana, R., and Lozano, J. A. (2011b). On the limits of effectiveness in estimation of distribution algorithms. In *Proceedings of the 2011 Congress on Evolutionary Computation (CEC-2011)*, pages 1573 –1580, New Orleans, USA. IEEE Press.

Eiben, E. A. and Smith, J. E. (2003). *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer.

Erdös, P. and Rényi, A. (1959). On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297.

Etxeberria, R. and Larrañaga, P. (1999). Global optimization using Bayesian networks. In Ochoa, A., Soto, M. R., and Santana, R., editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 151–173, Havana, Cuba.

Etxeberria, R., Larrañaga, P., and Pikaza, J. (1997). Analysis of the behaviour of the genetic algorithms when searching Bayesian networks from data. *Pattern Recognition Letters*, 18(11-13):1269–1273.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163.

Friedman, N., Linial, M., and Nachman, I. (2000). Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620.

Friedman, N. and Yakhini, Z. (1996). On the sample complexity of learning Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 274–282. Morgan Kaufmann.

Gámez, J. A. (2004). *Advances in Bayesian Networks*, chapter Abductive inference in Bayesian networks: A review, pages 101–120. Springer.

Gámez, J. A., Mateo, J. L., and Puerta, J. M. (2007). EDNA: Estimation of dependency networks algorithm. In Mira, J. and Álvarez, J. R., editors, *Bio-inspired Modeling of Cognitive Tasks, Second International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC-2007)*, volume 4527 of *Lecture Notes in Computer Science*, pages 427–436. Springer Verlag.

Gao, Y. and Culberson, J. C. (2005). Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143.

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

González, C., Lozano, J. A., and Larrañaga, P. (1999). The convergence behavior of PBIL algorithm: a preliminar approach. Technical Report KZZA-IK-99-03, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

González, C., Lozano, J. A., and Larrañaga, P. (2001). Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479.

González, C., Lozano, J. A., and Larrañaga, P. (2002). Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning*, 31(4):313–340.

Handa, H. (2005). Estimation of distribution algorithms with mutation. In *Evolutionary Computation in Combinatorial Optimization*, volume 3448 of *Lecture Notes in Computer Science*, pages 112–121. Springer Berlin / Heidelberg.

Harik, G., Lobo, F. G., and Golberg, D. E. (1998). The compact genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation (CEC-98)*, pages 523–528.

Harik, G., Lobo, F. G., and Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297.

Hauschild, M. and Pelikan, M. (2008). Enhancing efficiency of hierarchical BOA via distance-based model restrictions. MEDAL Report No. 2008007, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL).

Hauschild, M., Pelikan, M., Lima, C., and Sastry, K. (2007). Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, volume I, pages 523–530. ACM Press.

Hauschild, M., Pelikan, M., Sastry, K., and Goldberg, D. E. (2012). Using previous models to bias structural learning in the hierarchical BOA. *Evolutionary Computation*, 20(1):135–160.

Hauschild, M., Pelikan, M., Sastry, K., and Lima, C. (2009). Analyzing Probabilistic Models in Hierarchical BOA. *IEEE Transactions on Evolutionary Computation*, 13(6):1199–1217.

Heckerman, D. (1995). A tutorial on learning with Bayesian networks. Technical report, Microsoft Advanced Technology Division, Microsoft Corporation, Seattle, Washington.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In Lemmer, J. F. and Kanal, L. N., editors, *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence*, pages 149–164. Elsevier.

Herskovits, E. and Cooper, G. (1990). Kutató: An entropy-driven system for construction of probabilistic expert systems from database. Technical report, Knowledge Systems Laboratory, Medical Computer Science, Stanford University, California.

Höhfeld, M. and Rudolph, G. (1997). Towards a theory of population-based incremental learning. In *Proceedings of the 4th International Conference on Evolutionary Computation*, pages 1–5. IEEE Press.

Hoos, H. and Stutzle, T. (2000). SATLIB: An online resource for research on SAT. In van Maaren I. P. Gent, H. and Walsh, T., editors, *SAT2000*, pages 283–292. IOS Press.

Huang, C. and Darwich, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263.

Ising, E. (1925). The theory of ferromagnetism. *Zeitschrift fuer Physik*, 31:253–258.

Jensen, F. V. and Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer Verlag.

Kallel, L., Naudts, B., and Reeves, R. (2000). Properties of fitness functions and search landscapes. In Kallel, L., Naudts, B., and Rogers, A., editors, *Theoretical Aspects of Evolutionary Computing*, pages 177–208. Springer Verlag.

Kauffman, S. (1993). *Origins of Order*. Oxford University Press.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE Press.

Khor, S. (2009). Effect of degree distribution on evolutionary search. In *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference (GECCO-2009)*, pages 1857–1858, New York, NY, USA. ACM.

Kjaerulff, U. B. (1990). Triangulation of graphs - algorithms giving small total state space. Technical Report R-90-09, Aalborg University.

Koivisto, M. and Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

Lam, W. and Bacchus, F. (1994). Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293.

Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (2000a). Combinatorial optimization by learning and simulation of Bayesian networks. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 343–352, San Francisco, CA. Morgan Kaufmann Publishers.

Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (2000b). Optimization in continuous domains by learning and simulation of Gaussian networks. In Wu, A. S., editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pages 201–204.

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

Lauritzen, S. L. (1996). *Graphical Models*. Oxford:Clarendon Press.

Lee, K. Y. and El-Sharkawi, M. A., editors (2008). *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. John Wiley & Sons.

Lima, C. F., Lobo, F. G., and Pelikan, M. (2008). From mating pool distributions to model overfitting. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008)*, pages 431–438. IEEE Press.

Lima, C. F., Lobo, F. G., Pelikan, M., and Goldberg, D. E. (2011). Model accuracy in the Bayesian optimization algorithm. *Soft Computing*, 15:1351–1371.

Lima, C. F., Pelikan, M., Goldberg, D. E., Lobo, F. G., Sastry, K., and 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)*, pages 1083–1090. IEEE Press.

Lima, C. F., Pelikan, M., Lobo, F. G., and Goldberg, D. E. (2009). Loopy substructural local search for the bayesian optimization algorithm. In Stützle, T., Birattari, M., and Hoos, H. H., editors, *SLS*, volume 5752 of *Lecture Notes in Computer Science*, pages 61–75. Springer.

Malone, B., Yuan, C., Hansen, E. A., and Bridges, S. (2012). Improving the scalability of optimal Bayesian network learning with external-memory frontier breadth-first branch and bound search. *arXiv.org*, arXiv:1202.3744v1 [cs.AI].

Meek, C. (1995). Strong completeness and faithfulness in bayesian networks. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 411–418. Morgan Kaufmann Publishers.

Mendiburu, A., Santana, R., and Lozano, J. A. (2007). Introducing belief propagation in estimation of distribution algorithms: A parallel framework.

Technical Report EHU-KAT-IK-11/07, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

Mendiburu, A., Santana, R., and Lozano, J. A. (2012). *Markov Networks in Evolutionary Computation*, chapter Fast fitness improvements in Estimation of Distribution Algorithms using belief propagation, pages 141–155. Springer. In press.

Mühlenbein, H. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346.

Mühlenbein, H. (2012). *Markov Networks in Evolutionary Computation*, chapter Convergence of Estimation of Distribution Algorithms, pages 91–108. Springer. In press.

Mühlenbein, H. and Höns, R. (2005). The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27.

Mühlenbein, H. and Höns, R. (2006). The factorized distributions and the minimum relative entropy principle. In Pelikan, M., Sastry, K., and Cantú-Paz, E., editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 11–38. Springer-Verlag.

Mühlenbein, H. and Mahnig, T. (1999a). Convergence theory and applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology*, 7(1):19–32.

Mühlenbein, H. and Mahnig, T. (1999b). FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376.

Mühlenbein, H. and Mahnig, T. (2001). Evolutionary computation and beyond. In Uesaka, Y., Kanerva, P., and Asoh, H., editors, *Foundations of Real-World Intelligence*, pages 123–188. CSLI Publications, Stanford, California.

Mühlenbein, H., Mahnig, T., and Ochoa, A. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247.

Mühlenbein, H. and Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature (PPSN IV)*, volume 1141 of *Lectures Notes in Computer Science*, pages 178–187, Berlin. Springer Verlag.

Murphy, K. (2001). The Bayes Net Toolbox for Matlab. *Computer science and Statistics: Proceedings of Interface*, 33:1024–1034.

Naudts, B. and Kallel, L. (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15.

Neapolitan, R. E. (1990). *Probabilistic reasoning in expert systems - theory and algorithms*. Wiley.

Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, NJ.

Ocenasek, J. (2006). Entropy-based convergence measurement in discrete estimation of distribution algorithms. In Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E., editors, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, pages 39–50. Springer-Verlag.

Ochoa, A. and Soto, M. R. (2006). Linking entropy to estimation of distribution algorithms. In Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E., editors, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, pages 1–38. Springer-Verlag.

Okabe, T., Jin, Y., Sendhoff, B., and Olhofer, M. (2004). Voronoi-based estimation of distribution algorithm for multi-objective optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation CEC-2004*, pages 1594–1601, Portland, Oregon. IEEE Press.

Ortigosa-Hernandez, J., Rodriguez, J., Alzate, L., Lucania, M., Inza, I., and Lozano, J. A. (2012). Approaching sentiment analysis by using semi-supervised learning of multi-dimensional classifiers. *Neurocomputing*, Special Issue in Data Mining Applicacions and Case Studies(Accepted for publication).

Parviainen, P. and Koivisto, M. (2009). Exact structure discovery in Bayesian networks with less space. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-2009)*, pages 436–443, Arlington, Virginia, United States. AUAI Press.

Paul, T. and Iba, H. (2003). Linear and combinatorial optimizations by estimation of distribution algorithms. In *Proceedings of the 9th MPS Symposium on Evolutionary Computation*, pages 99–106.

Payne, J. and Eppstein, M. (2006). Emergent mating topologies in spatially structured genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*, pages 207–214.

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

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

Pelikan, M. and Goldberg, D. E. (2003). Hierarchical BOA solves Ising Spin Glasses and MAXSAT. In *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 1271–1282, Chicago, Illinois USA. Springer.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, volume I, pages 525–532, Orlando, FL. Morgan Kaufmann Publishers, San Francisco, CA.

Pelikan, M., Martin, K., Goldberg, D. E., Butz, M. V., and Hauschild, M. (2009). Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In *Proceedings of the*

11th Annual Genetic and Evolutionary Computation Conference (GECCO-2009), pages 851–858, New York, NY, USA. ACM.

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

Peña, J. M., Lozano, J. A., and Larrañaga, P. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter Benefits of data clustering in multimodal function optimization via EDAs, pages 99–124. Kluwer Academic Publishers, Boston/Dordrecht/London.

Peña, J. M., Lozano, J. A., and Larrañaga, P. (2004). Unsupervised learning of Bayesian networks via estimation of distribution algorithms: An application to gene expression data clustering. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(1):63–82.

Prügel-Bennett, A. (2000). Finite population effects for ranking and tournament selection. *Complex Systems*, 12(2):183–205.

Robles, V., Peña, J. M., Pérez, M. S., and Herves, V. (2006). GA-EDA: A new hybrid cooperative search evolutionary algorithm. In Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E., editors, *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*, pages 187–220. Springer Verlag.

Romero, T., Larrañaga, P., and Sierra, B. (2004). Learning Bayesian networks in the space of orderings with estimation of distribution algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(4):607–625.

Saeys, Y., Degroeve, S., Aeyels, D., Rouzé, P., and Van de Peer, Y. (2004). Feature selection for splice site prediction: A new method using EDA-based feature ranking. *BMC Bioinformatics*, 5:64–75.

Santafe, G., Lozano, J. A., and Larrañaga, P. (2006). Bayesian model averaging of naive Bayes for clustering. *IEEE transactions on systems man and cybernetics Part B*, 36(5):1149–1161.

Santana, R. (2002). An analysis of the performance of the mixture of trees factorized distribution algorithm when priors and adaptive learning are used. Technical Report ICIMAF 2002-180, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba.

Santana, R. (2005). Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97.

Santana, R., Bielza, C., Larrañaga, P., Lozano, J. A., Echegoyen, C., Mendiburu, A., Armañanzas, R., and Shakya, S. (2010). MATEDA-2.0: Estimation of distribution algorithms in MATLAB. *Journal of Statistical Software*, 35(7):1–30.

Santana, R., Bielza, C., Lozano, J. A., and Larrañaga, P. (2009a). Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *Proceedings of the 11th Annual Genetic and Evolutionary*

Computation Conference (GECCO-2009), pages 445–452, New York, NY, USA. ACM.

Santana, R., de León, E. P., and Ochoa, A. (1999). The edge incident model. In Ochoa, A., Soto, M. R., and Santana, R., editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 352–359, Havana, Cuba.

Santana, R., Larrañaga, P., and Lozano, J. A. (2005). Interactions and dependencies in estimation of distribution algorithms. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC-2005)*, pages 1418–1425, Edinburgh, U.K. IEEE Press.

Santana, R., Larrañaga, P., and Lozano, J. A. (2008a). Protein folding in simplified models with estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 12(4):418–438.

Santana, R., Larrañaga, P., and Lozano, J. A. (2009b). Research topics on discrete estimation of distribution algorithms. *Memetic Computing*, 1(1):35–54.

Santana, R., Mendiburu, A., and Lozano, J. A. (2008b). An empirical analysis of loopy belief propagation in three topologies: Grids, small-world networks and random graphs. In Jaeger, M. and Nielsen, T. D., editors, *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM-2008)*, pages 249–256.

Santana, R., Ochoa, A., and Soto, M. R. (2001). The mixture of trees factorized distribution algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 543–550, San Francisco, CA. Morgan Kaufmann Publishers.

Scheinerman, E. R. (1996). *Invitation to Dynamical Systems*. Prentice-Hall.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 7(2):461–464.

Shakya, S. (2006). *DEUM: A framework for an Estimation of Distribution Algorithm based on Markov Random Fields*. PhD thesis, The Robert Gordon University. School of Computing, Aberdeen, UK.

Shakya, S. and 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., and Brown, D. (2005). Using a Markov network model in a univariate EDA: An empirical cost-benefit analysis. In Beyer, H.-G. and O'Reilly, U.-M., editors, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 727–734, Washington, D.C., USA. ACM Press.

Shakya, S. and Santana, R., editors (2012). *Markov Networks in Evolutionary Computation*. Springer. In Press.

Shapiro, J. L. (2005). Drift and scaling in estimation of distribution algorithms. *Evolutionary Computation*, 13(1):99–123.

Shimony, S. E. (1994). Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410.

Silander, T. and 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)*, pages 445–452. Morgan Kaufmann Publishers.

Simionescu, P. A., Beale, D., and Dozier, G. V. (2007). Teeth-number synthesis of a multispeed planetary transmission using an estimation of distribution algorithm. *Journal of Mechanical Design*, 128(1):108–115.

Soto, M. R., González-Fernández, Y., and Ochoa, A. (2012). Vine estimation of distribution algorithm. In *Proceedings of the VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB-2012)*, Albacete, Spain.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

Thierens, D. and Bosman, P. A. (2001). Multi-objective optimization with iterated density estimation evolutionary algorithms using mixture models. In *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001), Cuba*, pages 129–136, Havana, Cuba.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.

Whitacre, J. M., Sarker, R. A., and Pham, Q. T. (2008). The self-organization of interaction networks for nature-inspired optimization. *IEEE Transactions on Evolutionary Computation*, 12(2):220–230.

Wright, A., Poli, R., Stephens, C., Langdon, W. B., and Pulavarty, S. (2004). An estimation of distribution algorithm based on maximum entropy. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pages 343–354, Seattle, WA, USA. Springer.

Wu, H. and Shapiro, J. L. (2006). Does overfitting affect performance in estimation of distribution algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO-2006)*, pages 433–434, New York, USA. ACM.

Yannakakis, M. (1981). Computing the minimum fill-in is NP-complete. *SIAM Journal on Discrete Mathematics*, 2(1):77–79.

Yu, T.-L., Santarelli, S., and Goldberg, D. E. (2006). Military antenna design using a simple genetic algorithm and hboa. In Pelikan, M., Sastry, K., and Cantú-Paz, E., editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 275–290. Springer-Verlag.

Yuan, B., Orlowska, M. E., and Sadiq, S. W. (2007). Finding the optimal path in 3D spaces using EDAs - the wireless sensor networks scenario. In *Proceedings of the Adaptive and Natural Computing Algorithms, 8th International Conference (ICANNGA-2007)*, pages 536–545, Warsaw, Poland. Springer Verlag.

Zhang, Q. (2004). On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 8(1):80–93.

Zhang, Q. and Mühlenbein, H. (2004). On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):127–136.

Zhang, Q., Sun, J., and Tsang, E. (2005). Evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 9(2):192–200.

Zhang, Q., Sun, J., and Tsang, E. (2007). Combinations of estimation of distribution algorithms and other techniques. *International Journal of Automation and Computing*, 4(3):273–280.

Zhang, Q., Sun, J., Tsang, E., and Ford, J. A. (2003). Hybrid estimation of distribution algorithm for global optimization. *Engineering Computations*, 21(1):91–107.

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