Departamento de Ciencias de la Computación e Inteligencia Artificial
Konputazio Zientziak eta Adimen Artifiziala Saila
Department of Computer Science and Artificial Intelligence

Universidad del País Vasco
Euskal Herriko Unibertsitatea
University of the Basque Country

Contributions on Theoretical Aspects of Estimation of Distribution Algorithms

by

Cristina González

Co-Supervised by J. A. Lozano and P. Larrañaga

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country in partial fulfillment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, November 2005

Departamento de Ciencias de la Computación e Inteligencia Artificial
Konputazio Zientziak eta Adimen Artifiziala Saila
Department of Computer Science and Artificial Intelligence

Universidad del País Vasco
Euskal Herriko Unibertsitatea
University of the Basque Country

Contributions on Theoretical Aspects of Estimation of Distribution Algorithms

by

Cristina González

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country in partial fulfillment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, November 2005

Acknowledgments

It is time to say thanks, to acknowledge the encouragement and support of people that have helped me in the process that now comes to completion with this dissertation.

First and foremost, I want to thank Jose Antonio Lozano and Pedro Larrañaga, my thesis supervisors. Without their support and patience, this Ph.D. would not have been possible. I am deeply indebted to them for their constant help, and will always appreciate their availability, the time they have devoted to my work, and their encouragement when things weren't going quite right... Thanks so much.

I am also very grateful to my colleagues in the Intelligent Systems Group, as well as to the department of Computer Science and Artificial Intelligence of the University of the Basque Country for their help and kindness. In addition, I owe a debt of gratitude to the department of Applied Mathematics, Statistics and Operation Research of the University of the Basque Country, for making it possible for me to work on this memory near home.

I am grateful to the University of the Basque Country for four years of financial support under a pre-doctoral grant.

Thanks too, to all my friends, for the understanding, kindness and shared times that have made of me a better person.

Thanks many times over to my parents and sisters. They are the people who have taught me the best lesson in life, for I know that the love and support they give me is, and always will be, unconditional.

And loving thanks, finally, to Hugo and Jousé –the people who move my world. To Hugo, for his smile each day, and to Jousé, for sharing my life, dreams, hopes and future.

Agradecimientos

Quisiera aprovechar esta oportunidad para agradecer a todas aquellas personas que, de un modo u otro, me han animado y ayudado durante el proceso que termina con la presente memoria.

En primer lugar y de una manera especial, quisiera dar las gracias a Jose Antonio Lozano y Pedro Larrañaga, mis directores de tesis. Sin su paciencia y su apoyo no hubiera podido llegar al final. Les estoy profundamente agradecida por su constante ayuda, su disponibilidad en todo momento, el tiempo dedicado a mi trabajo, y el ánimo en los momentos menos buenos... De todo corazón, gracias.

Estoy también muy agradecida a mis compañeros del grupo de investigación Intelligent Systems Group, así como a los compañeros del departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad del País Vasco, tanto por la ayuda prestada, como por la relación de amistad compartida. Debo también agradecer al departamento de Matemática Aplicada, Estadística e Investigación Operativa de la Universidad del País Vasco el haberme permitido trabajar en esta memoria cerca de mi casa.

Agradecida estoy a la Universidad del País Vasco por haberme concedido apoyo económico, a través de una beca predoctoral, durante cuatro años.

Gracias también a todos y todas mis amigos y amigas, la amistad, el cariño y el tiempo compartidos han hecho de mí mejor persona.

Mis padres y hermanas me han enseñado la mejor lección de mi vida, sé que el amor y la ayuda que me brindan son y serán siempre incondicionales. Gracias.

Hugo y Jousé son quienes mueven mi mundo. Gracias a Hugo por la sonrisa diaria. Gracias a Jousé por ser mi compañero vital, por compartir sueños, esperanzas y futuro.

A Hugo y Jousé

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

The desire for optimality (perfection) is inherent in humans. The search for extremes inspires mountaineers, scientists, mathematicians, and all the rest of the humanity. The theory of optimization, a prominent research topic for the last decades, refers to the quantitative study of optima and the methods for finding them.

There are optimization problems that are characterized as "hard", in the sense that any optimal solution process for them takes practically infinite time (NP-hard problems). These are often combinatorial optimization problems, like cutting and packing, routing, network design, assignment, scheduling, or time-tabling problems. There are two extremes for solving NP-hard problems. The first extreme uses exact methods such as Branch and Bound, which requires exponential computing time. In practice, for large-sized problems, exact methods are generally unable to produce any results in reasonable time. In the second extreme are non-exact methods, called metaheuristics. Metaheuristics are general frameworks for heuristics in solving hard optimization problems. These algorithms do not guarantee optimality of the solutions found, but in practice, they usually manage to obtain good results within reasonable computing time.

Evolutionary Algorithms (EAs) are metaheuristic techniques that have grown into one of the most successful tools for solving optimization problems. EAs are search methods that take their inspiration from natural selection and survival of the fittest in the biological world. EAs differ from more traditional optimization techniques in that they involve a search from a population of solutions, not from a single point. Each iteration of an EA involves a competitive selection that weeds out poor solutions. The solutions with high fitness are recombined with other solutions by swapping parts of one solution with those of another. Solutions are also mutated by making a small change to a single element of the solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the space for which good solutions have already been seen.

This dissertation is devoted to theoretically analyzing a new class of EAs: Estimation of Distribution Algorithms (EDAs). EDAs have recently been recognized as a new computing paradigm in evolutionary computation. Unlike other EAs, EDAs do not use crossover or mutation. Instead, they explicitly extract global statistical information from the selected solutions and subsequently build a probability model of promising solutions, based on the extracted information. New solutions are sampled from the model thus built. According to the statistical information they exploit, EDAs can be easily classified into three categories: (i) EDAs using only first-order statistics, which employ

probability models in which all the variables are independent and hence only need to estimate the marginal probability of each variable in the selected solutions at each generation; (ii) EDAs which consider second-order statistics, taking into account dependencies between pairs of variables. Here not just the learning of the parameters is necessary but also they need a learning of the dependency structure; (iii) EDAs that require higher-order statistics. They use a conditional dependence network to model the probability distributions.

Research on EDAs has been focused on proposals of new algorithms and on their applications. However, little attention has been given to their theoretical aspects. Therefore, one of the aims of this thesis is to reduce the gap between the sparse theoretical results and the magnitude of empirical observations.

Here with reference to EDAs we tackle two important issues that measure the performance of any optimization algorithm: convergence and time complexity. On the one hand it is important to know whether or "under what assumptions" it can be guaranteed that the algorithm reaches a (n optimal) solution, which is analyzed with the study of the convergence of the algorithm. Obviously, no unconditional 'yes' can be expected, so the question is mostly reformulated as "under what assumptions can it be guaranteed that the algorithm reaches a (n optimal) solution". Immediately related to this question is the issue of the type of guarantee. In particular, the stochastic nature of EDAs impedes crisp guarantees, turning the question into a probabilistic one. Technically speaking, almost sure convergence, convergence in probability or convergence in mean are some options. From a purely theoretical point of view, guaranteeing, for instance, convergence with probability one is satisfactory. Practically, however, the speed of convergence is just as important. The number of expected search steps needed to reach a (n optimal) solution – known as computation time – is an important implementation independent measure of an algorithm's efficiency. A related concept studied here is the relationship between computation time and problem size, which is called time complexity is studied.

Basically there are two mathematical tools used in this dissertation to analyze EDAs: Markov chain models and discrete dynamical systems, which are two commonly-used approaches for studying EAs.

Since the population of an EDA only depends on the state of previous population in a probabilistic manner, it is clear that Markov chains are appropriate to model and analyze EDAs, enabling the study of convergence in the sense of probability. Such an approach exactly models the behavior of an EDA, but due to the combinatorial explosion of the populations, the transition probability matrix is very difficult to obtain and analyze, nor is it easy to come up with a simple derivation of detailed answers to particular questions (like the expected time of visiting the optimum for the first time).

The dynamical systems approach often assumes that the size of population is infinite. As a result the iterative process of an EA is represented by deterministic dynamical systems and therefore the mathematical analysis becomes easier. However, an upper bound of the error between the actual EA and its model is not easily estimated.

1.1 Overview of the Dissertation

This dissertation is divided into four parts and is comprised of nine chapters.

Part I consists of three chapters. The present chapter is devoted to generally introducing and explaining the motivation behind the dissertation. Chapter 2 introduces the necessary notation and mathematical tools used in the thesis for the modeling and study of EDAs. In Chapter 3 the reasons for the birth of EDAs are explained and a review is made of the EDAs proposed for the solution of

combinatorial optimization problems and optimization in continuous domains, ordered according to the complexity of the interrelations that they are able to express.

Part II consists of four chapters and presents the convergence results obtained for some EDAs. In Chapter 4 Markov chains are used to give a general convergence theorem for EDAs. The most common discrete EDAs are analyzed using this theorem, resulting in convergence and non-convergence algorithms. For those algorithms that do not converge, some conditions have been imposed on the parameters of their probability distributions to guarantee convergence. The aim of Chapter 5 is to offer a mathematical analysis of convergence properties of the PBIL algorithm. This analysis is carried out under two analytic frameworks: Markov chains and discrete dynamical systems. Although the Markov chain used here is neither irreducible nor aperiodic – which implies a very complex analysis of the chain – a strong dependence of convergence behavior of PBIL on parameter α and on initial probability distributions is proven. Moreover the dynamical systems framework enables us to carry out a more general analysis. Chapter 6 is devoted to the UMDA algorithm. Under the Markov chains framework an empirical analysis of the convergence behavior of the algorithm is described. This chapter also shows how UMDA with infinite population and proportionate selection is modeled using discrete dynamical systems. This part finalizes with Chapter 7, which presents the only analysis made for a continuous EDA in this dissertation: the theoretical study of the behavior of the Univariate Marginal Distribution Algorithm for continuous domains (UMDA_c) in dimension n. To this end the application of this algorithm to the minimization of two kinds of functions is mathematically modeled. First n-dimensional linear functions are used to model the algorithm when far from the optimum. Next quadratic function is used to analyze the algorithm when near the optimum.

Part III consists of one chapter, which deals with the analysis of time complexity of EDAs. First, it offers a result concerning worst-case first hitting time for EDAs based on Markov chains and discusses how this result influences the calculus of bounds of average-case hitting times for EDAs. Next a study based on empirical results of the average first hitting time of EDAs is presented, the algorithms are applied to one example of linear, pseudo-modular, and unimax functions. The average first hitting time is analyzed and compared. Part III also addresses recent issues in EDAs: (i) the relationship between the complexity of the probabilistic model used by the algorithm and its efficiency, and (ii) the matching between this model and the relationship among the variables of the objective function.

Finally, Part IV, which consists of Chapter 9, lists the main contributions of this dissertation and points out some lines of further research.

Chapter 2

Basic Mathematical Tools

This chapter introduces the necessary notation and mathematical tools used in this dissertation for the modeling and study of EDAs. The idea is not to make an exhaustive review of each topic, but to summarize the definitions, basic concepts and theorems used in our research. However, readers who want to go more deeply into a subject are encouraged to consult the references contained in each Section.

In Section 2.1 of this chapter, some preliminary concepts related to probability theory are introduced. Then order statistics are briefly presented in Section 2.2. Section 2.3 gives necessary information about Markov chains for the reading of this dissertation, and finally, in Section 2.4, dynamical systems are introduced.

2.1 Probability Theory

The mathematical modeling of EDAs involves dealing with uncertainty, and probability theory provides us with a suitable formal framework for doing so. Therefore this section presents the basic set of definitions and theorems of probability theory that will be used throughout the dissertation. The reader may be interested in consulting Billingsley (1986) and DeGroot (1987) for further information about probability theory.

2.1.1 Basic Definitions

In order to use mathematics to describe an experimental phenomenon it is necessary to build a mathematical model to describe the phenomenon. One of the first steps in this modeling process is to define the set of all possible outcomes.

Definition 2.1. *The exhaustive set of mutually exclusive possible outcomes of a random experiment is called the* **sample space** *of the experiment. The sample space will be denoted by Ω, and an arbitrary element of Ω will be denoted by ω. Ω is referred to as a* **discrete** *sample space when there is a countable number (finite or infinite) of distinct outcomes; otherwise Ω is referred to as a* **continuous** *sample space.*

Definition 2.2. *The subsets of the sample space Ω of a given random experiment are referred to as* **events**.

Thus, an event is a collection of outcomes of a random experiment. The sample space Ω of the random experiment in question is itself the event that, by definition, always occurs. At the other extreme, the empty set $\emptyset$ is the event that, by definition, never occurs.

Like most other mathematical theories, probability theory does not necessarily apply to absolutely all the collections of subsets of Ω, i.e. it is usual to restrict attention to collections of admissible subsets of Ω that are closed under a certain set of operations. Specifically we usually impose the condition that any subset of Ω that can be constructed from a countable number of subsets of Ω using a certain set of operations should itself be admissible. This leads to the following definition which plays an important role in probability theory.

Definition 2.3. *Let Ω be the sample space of a random experiment. A class $\mathcal{F}$ of subsets of Ω, i.e. events, is called* $\boldsymbol{\sigma}$ **− algebra** *if it contains Ω itself and is closed under the formation of complements and countable unions, that is:*

(i) $\Omega \in \mathcal{F}$.

(ii) $A \in \mathcal{F} \Rightarrow A^c \in \mathcal{F}$.

(iii) $\bigsqcup_{n=1}^{\infty}(A_n) \in \mathcal{F}$ *for each sequence* $(A_n)_{n\in\mathbb{N}}$ *with* $A_n \in \mathcal{F}$ *for all* $n \in \mathbb{N}$,

where $A^c = \Omega \setminus A$ denotes the complement of set A.

In any random experiment with sample space Ω, we assume that the collection of interesting or admissible events $\mathcal{F}$ forms a σ-algebra. $\mathcal{F}$ is usually known as the event space of the random experiment.

Definition 2.4. *Let $\mathcal{F}$ be a σ-algebra relative to Ω and $\mathcal{E} \subset \mathcal{F}$ with $\mathcal{E} \neq \emptyset$. If $\mathcal{F} \subset \mathcal{F}'$ for each σ-algebra relative to Ω with $\mathcal{E} \subset \mathcal{F}'$ then $\mathcal{F}$ is called the* $\boldsymbol{\sigma}$ **− algebra generated by** $\boldsymbol{\mathcal{E}}$ *or a* **minimal** $\boldsymbol{\sigma}$ **− algebra** *and is denoted by $\sigma(\mathcal{E})$.*

The following σ-algebra(s) will be of special interest: Let $\Omega = \mathbb{R}(\mathbb{R}^n)$. The σ-algebra $\mathcal{B}(\mathcal{B}^n)$ generated by the sets of the form $[a, b) = \{x \in \mathbb{R} : a \leq x \leq b; a, b \in \mathbb{R}\}(\{x \in \mathbb{R}^n : a \leq x \leq b; a, b \in \mathbb{R}^n\})$ is called the (**n-dimensional**) **Borel σ-algebra**. An element $B \in \mathbf{B}(\mathbf{B}^n)$ is termed a **Borel set**.

Each time a random experiment is performed, exactly one of the possible events will occur. Usually it is not known which of the possible events will occur. The investigator may feel that each event has "as good a chance" of occurring as any other, or he may feel that some events have a "better" chance of occurring than others. In order to translate these "feelings" into precise mathematical terms, he can define a probability measure, which will measure how likely an event is to occur when the random experiment is run.

Definition 2.5. *Let Ω and $\mathcal{F}$ be the sample space and the event space, respectively of a random experiment. A real-valued function on $\mathcal{F}$, $P : \mathcal{F} \to \mathbb{R}$ is called a* **probability measure** *if it satisfies the following conditions:*

(i) $P(A) \geq 0$ *for all* $A \in \mathcal{F}$.

(ii) $P(\Omega) = 1$.

(iii) Given a sequence of pairwise disjoint events $(A_n)_{n\in\mathbb{N}}$, $P(\bigsqcup_{n=1}^{\infty}(A_n)) = \sum_{n=1}^{\infty} P(A_n)$.

Definition 2.6. *Let Ω and $\mathcal{F}$ be the sample space and the event space respectively, of a random experiment. Let P be a probability measure on $\mathcal{F}$. The triplet $(\Omega, \mathcal{F}, \mathsf{P})$ is called a* **probability space**.

The probability space $(\Omega, \mathcal{F}, \mathsf{P})$ contains the information necessary to study the probabilistic properties of the random experiment. However, the analysis might be complicated by the fact that the outcome of the experiment may not be a number. In order to alleviate this problem the points in Ω are often mapped into real numbers. In this way a number can be used to represent an outcome of the random experiment. A unidimensional random variable is a function that associates a numerical value with every outcome of a random experiment. We denote, as usual, the unidimensional random variables by uppercase letter or letters and their values by the same letter or letters in lowercase.

Definition 2.7. *Let $(\Omega, \mathcal{F}, \mathsf{P})$ be an arbitrary probability space and $\mathcal{B}$ the Borel σ-algebra. A real-valued function on Ω, $X : \Omega \to \mathbb{R}$ such that:*

$$\forall B \in \mathcal{B} \quad X^{-1}(B) = \{X \in B\} \in \mathcal{F}$$

is called a **unidimensional random variable**.

Clearly, a random variable is neither "random" nor a "variable". The variable is "random" in the sense that its value may vary from trial to trial as the random experiment is repeated.

Definition 2.8. *Let X be a unidimensional random variable. X is said to be a* **unidimensional discrete random variable** *if there is a countable number (finite or infinite) of distinct values that X can take, otherwise X is said to be a* **unidimensional continuous random variable**.

Definition 2.9. *Let X be a unidimensional discrete random variable. $p(x) = \mathsf{P}(X = x) = \mathsf{P}(\{\omega \in \Omega | X(\omega) = x\})$ is called a* **probability mass function for X** *if it satisfies the following conditions:*

(i) $p(x) \geq 0$ *for all* $x \in \mathbb{R}$.

(ii) $\sum_x p(x) = 1$.

It is common to use the term probability distribution as synonym of probability mass function. Although strictly speaking the latter term is more accurate, in this dissertation we will use the former. Note also that $p(x)$ denotes the probability that $X = x$ as well as a probability distribution for X. Whether $p(x)$ refers to a probability or a probability distribution should be clear from the context.

Definition 2.10. *Let X be a unidimensional continuous random variable. $f(x)$ is called a* **probability density function for X** *if it satisfies the following conditions:*

(i) $f(x) \geq 0$ *for all* $x \in \mathbb{R}$.

(ii) $\int_{\mathbb{R}} f(x)dx = 1$.

(iii) $\mathsf{P}(\alpha \leq X \leq \beta) = \mathsf{P}(\{\omega \in \Omega | \alpha \leq X(\omega) \leq \beta\}) = \int_\alpha^\beta f(x)dx$ *for all α and β such that $\alpha \leq \beta$.*

Definition 2.11. *Let $(\Omega, \mathcal{F}, \mathsf{P})$ be an arbitrary probability space and $\mathcal{B}^n$ the n-dimensional Borel σ-algebra. A real-valued function on Ω, $\mathbf{X} = (X_1, \ldots, X_n) : \Omega \to \mathbb{R}^n$ such that:*

$$\forall B \in \mathcal{B}^n \quad \mathbf{X}^{-1}(B) = \{\mathbf{X} \in B\} \in \mathcal{F}$$

is called an n**-dimensional random variable**.

Note that we use a letter or letters in boldface to designate a multidimensional random variable and the same boldface lowercase letter or letters to denote an assignment of a value to the multidimensional random variable.

Definition 2.12. *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be an* n*-dimensional random variable. If there is a countable number (finite or infinite) of distinct values that* $\mathbf{X}$ *can take, i.e., every* X_i *is a unidimensional discrete random variable, then* $\mathbf{X}$ *is said to be an* n**-dimensional discrete random variable**. *On the other hand, if every* X_i *is a unidimensional continuous random variable, then* $\mathbf{X}$ *is said to be an* n**-dimensional continuous random variable**.

Since unidimensional random variables are a particular case of multidimensional random variables, in the remainder of this section we will discuss only the latter.

Definition 2.13. *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be an* n*-dimensional discrete random variable.* $p(\mathbf{x}) = p(x_1, \ldots, x_n) = P(X_1 = x_1, \ldots, X_n = x_n) = P(\mathbf{X} = \mathbf{x}) = P(\{\omega \in \Omega | \mathbf{X}(\omega) = \mathbf{x}\})$ *is called a* **joint probability mass function for** $\mathbf{X}$ *if it satisfies the following conditions:*

(i) $p(\mathbf{x}) \geq 0$ *for all values* $\mathbf{x}$ *of* $\mathbf{X}$.

(ii) $\sum_{\mathbf{x}} p(\mathbf{x}) = 1$.

It is common to use the term joint probability distribution as a synonym of joint probability mass function. Although strictly speaking the latter term is more accurate, in this dissertation we will use the former. Note also that $p(\mathbf{x})$ denotes the probability that $\mathbf{X} = \mathbf{x}$ as well as a probability distribution for $\mathbf{X}$. Whether $p(\mathbf{x})$ refers to a probability or a joint probability distribution should be clear from the context.

Definition 2.14. *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be an* n*-dimensional continuous random variable.* $f(\mathbf{x}) = f(x_1, \ldots, x_n)$ *is called a* **joint probability density function for** $\mathbf{X}$ *if it satisfies the following conditions:*

(i) $f(\mathbf{x}) \geq 0$ *for all* $\mathbf{x} \in I\!R^n$.

(ii) $\int_{I\!R} \cdots \int_{I\!R} f(x_1, \ldots, x_n) dx_1 \cdots dx_n = 1$.

(iii) $P(\boldsymbol{\alpha} < \mathbf{X} \leq \boldsymbol{\beta}) = P(\alpha_1 < X_1 \leq \beta_1, \ldots, \alpha_n < X_n \leq \beta_n)$
$= P(\{\omega \in \Omega | \alpha_i < X_i(\omega) \leq \beta_i \text{ for all } i\})$
$= \int_{\alpha_1}^{\beta_1} \cdots \int_{\alpha_n}^{\beta_n} f(x_1, \ldots, x_n) dx_1 \cdots dx_n$ *for all* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ *and* $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ *such that* $\alpha_i \leq \beta_i$ *for all* i.

Definition 2.15. *Let* $p(\mathbf{x})$ *be a joint probability mass function for an* n*-dimensional discrete random variable* $\mathbf{X}$. *Then:*

$$F(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x}) = \sum_{x_1' \leq x_1} \cdots \sum_{x_n' \leq x_n} p(x_1', \ldots, x_n')$$

is called the **cumulative distribution function** *for the* n*-dimensional discrete random variable* $\mathbf{X}$.

Definition 2.16. *Let* $f(\mathbf{x})$ *be a joint probability density function for an* n*-dimensional continuous random variable* $\mathbf{X}$. *Then:*

$$F(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x}) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} f(x_1', \ldots, x_n') dx_1' \cdots dx_n'$$

is called the **cumulative distribution function** *for the n-dimensional continuous random variable* $\mathbf{X}$.

Definition 2.17. *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be an n-dimensional discrete random variable. Then*

$$p_{X_i}(x_i) = \sum_{x'_1} \cdots \sum_{x'_n} P(X_1 = x'_1, \ldots, X_i = x_i, \ldots X_n = x'_n) \text{ , for } i = 1, \ldots, n$$

are said to be the **univariate marginal probability mass functions**.

Definition 2.18. *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be an n-dimensional continuous random variable. Then*

$$f_{X_i}(x_i) = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} f(x_1, \ldots, x_n) dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n \text{ , for } i = 1, \ldots, n$$

are said to be the **univariate marginal probability density functions**.

Definition 2.19. *Let* X *be a discrete random variable having a probability mass function* $p(x)$. *Then*

$$E[X] = \sum_{x \in \Omega} xp(x)$$

is called the **expectation of the discrete random variable** X.

In cases where more than one probability mass function is being used, we write $E_p[X]$ *to distinguish among them.*

If $g(\cdot)$ *is a real-valued measurable function, and* $\sum_{x \in \Omega} |g(x)|p(x) < \infty$, *then*

$$E[g(X)] = \sum_{x \in \Omega} g(x)p(x)$$

is called the **expectation of the discrete random variable** $g(X)$.

Definition 2.20. *Let* X *be a continuous random variable having a probability density function* $f(x)$. *Then*

$$E[X] = \int_{-\infty}^{\infty} xf(x)dx$$

is called the **expectation of the continuous random variable** X.

If $g(\cdot)$ *is a real-valued measurable function, and* $\int_{-\infty}^{\infty} |g(x)|f(x)dx < \infty$, *then*

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx$$

is called the **expectation of the continuous random variable** $g(X)$**.**

Definition 2.21. *Let* $\mathbf{X} = (X_1, \ldots, X_n)$ *be a random vector, then the* **expectation vector of X** *is given by:*

$$E[\mathbf{X}] = (E[X_1], \ldots, E[X_n]).$$

2.1.1.1 Modes of Stochastic Convergence

Since random sequences are defined on probability spaces, the main difference between all modes of stochastic convergence and the convergence concept of classical analysis relies on the fact that the definition of the stochastic version must take into account the existence of a probability measure.

These various modes of stochastic convergence are distinguished by the way in which the probability measure enters the definition. Here, only the most popular concepts will be considered.

Definition 2.22. *Let X be a random variable and (X_n) a sequence of random variables defined on a probability space $(\Omega, \mathcal{F}, P)$. Then (X_n) is said*

(a) **to converge completely to** X, *denoted as $X_n \xrightarrow{c} X$, if for any $\epsilon > 0$*

$$\lim_{n\to\infty} \sum_{i=1}^{n} P\{|X_i - X| > \epsilon\} < \infty$$

(b) **to converge almost surely to** X, *denoted as $X_n \xrightarrow{a.s.} X$, if*

$$P\{\lim_{n\to\infty} |X_n - X| = 0\} = 1$$

(c) **to converge in probability to** X, *denoted as $X_n \xrightarrow{P} X$, if for any $\epsilon > 0$*

$$\lim_{n\to\infty} P\{\, |X_n - X| < \epsilon \,\} = 1$$

(d) **to converge in mean to** X, *denoted as $X_n \xrightarrow{m} X$, if*

$$\lim_{n\to\infty} E[\, |X_n - X| \,] = 0.$$

There are two chains of implications that relate the above concepts:

Theorem 2.1. *Let X be a random variable and (X_n) a sequence of random variables defined on a probability space $(\Omega, \mathcal{F}, P)$. Then*

(i) $X_n \xrightarrow{c} X \Rightarrow X_n \xrightarrow{a.s.} X \Rightarrow X_n \xrightarrow{P} X$.

(ii) $X_n \xrightarrow{m} X \Rightarrow X_n \xrightarrow{P} X$.

The converse is not true in general.

2.2 Order Statistics

The subject of order statistics deals with the properties and applications of ordered random variables and functions involving them. This theory represents a substantial "building block" in analyzing selection procedures and convergence rates of Evolutionary Algorithms in continuous spaces. Here we only introduce briefly a few concepts that will help to understand the analysis of UMDA in continuous spaces carried out in this dissertation. An excellent summary of theory on order statistics can be found in David (1970).

Definition 2.23. *Let $X_1, \ldots, X_n$ be random variables. If they are rearranged in ascending order of magnitude, written as:*

$$X_{1:n} \leq X_{2:n} \leq \ldots \leq X_{n:n}$$

then $X_{i:n}$ $(i = 1, \ldots, n)$ is called the $\mathbf{i}^{th}$ **order statistic**.

Ordered random variables $X_{i:n}$ are necessarily dependent whereas unordered random variables X_i may be statistically independent or dependent. Moreover, variables can be identically distributed (all the variables have the same probability distribution function) or not. Here, only the independent identical distributed case will be presented. We also consider that X_i variables are continuous.

Theorem 2.2. *Let $X_1, \ldots, X_n$ be independent identically distributed continuous random variables with probability density function $f(x)$ and cumulative density function $F(x)$. Then the probability density function of the i^{th} order statistic $X_{i:n}$ is:*

$$f_{i:n}(x) = \frac{1}{B(i, n-i+1)} f(x) F^{i-1}(x)[1-F(x)]^{n-i}$$

where $B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt$, $a > 0$, $b > 0$ denotes the complete Beta function.

Theorem 2.3. *Let $X_1, \ldots, X_n$ be a collection of n independent identically distributed continuous random variables with density $f(x)$ and cumulative distribution function $F(x)$ for all $i = 1, \ldots, n$. Then the joint density of the i^{th} and j^{th} order statistic , $X_{i:n}$ and $X_{j:n}$, for $1 \leq i < j \leq n$ is:*

$$f_{i,j:n}(x_i, x_j) = \frac{n!}{(i-1)!(j-i-1)!(n-j)!} F(x_i)^{i-1} \left(F(x_j) - F(x_i)\right)^{j-i-1} \left(1 - F(x_j)\right)^{n-j} f(x_i) f(x_j)$$

with $-\infty < x_i < x_j < \infty$.

In many cases the expectations $\mathsf{E}[X_{i:n}]$ of order statistics are of special interest, but unfortunately analytic expressions for $\mathsf{E}[X_{i:n}]$ are rarely obtainable (because the calculations are often complex) except in a few particular cases.

2.3 Markov Chains

As we will see later, one of the most powerful mathematical tools for modeling EDAs is a particular case of stochastic processes: Markov chains. Markov chains constitute a widely studied topic on probability theory and this section is not intended as an introduction or review of them. Here only the fundamental concepts of Markov chains that are basic for later purposes are presented. For further information see Isaacson and Madsen (1985), which deals entirely with the topic, or Ross (1997) and Taylor and Karlin (1994) which are good introductions to stochastic processes.

The term stochastic process has been reserved for the study of families of random variables and the relationships between those variables.

Definition 2.24. *A* **stochastic process** *is a family of random variables defined on some sample space Ω. If there are countably many members of the family, the process will be denoted by $X_0, X_1, X_2, \ldots$. In this case the process is called a* **discrete-time** *process. If there are uncountably many members of the family, the process will be denoted by $\{X_t | t \geq 0\}$ and it is called a* **continuous-time** *process.*

Definition 2.25. *The set of distinct values assumed by a stochastic process is called the* **state space** *and denoted by E. If the state space of a stochastic process is countable, or finite, the process will be called a* **chain**.

Definition 2.26. *A stochastic process $\{X_k\}$, $k = 0, 1, 2 \ldots$ with state space $E = \{0, 1, 2 \ldots\}$ is said to satisfy the* **Markov property** *if for every n and all states $i_0, \ldots, i_{n-1}, i, j$ it is true that:*

$$P\{X_{n+1} = j | X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i\} = P\{X_{n+1} = j | X_n = i\}$$

Placing restrictions on the process to be considered, we can define a discrete-time Markov chain (in the remainder of this section we only deal with this kind of stochastic process).

Definition 2.27. *A* **discrete-time Markov chain** *is a discrete-time process with a countable or finite state space whose (time) index set is* $T = 0, 1, 2, \ldots$ *and which satisfies the Markov property.*

It is frequently convenient to label the state space of the Markov chain by the nonnegative integers $\{0, 1, 2, \ldots\}$, which we will do unless otherwise explicitly stated, and it is customary to speak of X_n as being in state i if $X_n = i$.

Definition 2.28. *The probability of* X_{n+1} *being in state* j *given that* X_n *is in state* i *is called the* **one-step transition probability** *and is denoted by* $P_{ij}^{n,n+1}$. *That is*

$$P_{ij}^{n,n+1} = \mathsf{P}\{X_{n+1} = j | X_n = i\}.$$

The notation emphasizes that in general the transition probabilities are a function not only of the initial and final states, but also of the time of transition as well. We will limit our discussion to cases where in which the one-step transition probabilities are independent of the time variable n. These chains are known as **stationary** or **homogeneous** Markov chains. Then $P_{ij}^{n,n+1} = P_{ij}$ is independent of n, and P_{ij} is the conditional probability that the state value undergoes a transition from i to j in one trial. It is customary to arrange these numbers P_{ij} in a matrix, in the infinite square array

$$\mathbf{P} = \begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} & \cdots \\ P_{10} & P_{11} & P_{12} & P_{13} & \cdots \\ P_{20} & P_{21} & P_{22} & P_{23} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots \\ P_{i0} & P_{i1} & P_{i2} & P_{i3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots \end{pmatrix}$$

and to refer to $\mathbf{P} = (P_{ij})$ as the **transition probability matrix** of the Markov chain. If the number of states is finite, then $\mathbf{P}$ is a finite square matrix whose number of rows is equal to the number of states. Note that every transition matrix has the following properties:

(i) all the entries are nonnegative,

(ii) the sum of the entries in each row is one.

This matrix contains all the relevant information regarding the movement of the process among the states in E. In fact the study of the Markov chain can be reduced to a study of the corresponding transition probability matrix $\mathbf{P}$. The chain is completely defined once its initial state X_0 (or more generally the probability distribution of X_0) is specified.

2.3.1 Limiting Probabilities

Analysis of a Markov chain mainly involves calculating the probabilities of the possible realizations of the process. Crucial to these calculations are the n-step transition probability matrices $\mathbf{P}^{(n)} = (P_{ij}^{(n)})$. Here $P_{ij}^{(n)}$ denotes the probability that the process will go from state i to state j in n transitions. Formally:

$$P_{ij}^{(n)} = \mathsf{P}\{X_{m+n} = j | X_m = i\}.$$

The n-step transition matrix may be obtained by multiplying the matrix $\mathbf{P}$ by itself n times:

$$\mathbf{P}^{(n)} = \mathbf{P}^n.$$

Let (X_t) be a finite Markov chain with probability matrix $\mathbf{P}$. For chains with many states the problem of calculating $\mathbf{P}^n$ becomes quite tedious, especially for large n. In some cases the n-step transition matrix converges to a fixed vector $\boldsymbol{\pi}$, which is independent of the initial state X_0 of the chain. The limit vector $\boldsymbol{\pi}$ is called the "long run distribution". Below we present the terminology necessary to give the theorem that establishes when this "long run distribution" exists.

Definition 2.29. *A subset, C, of the state space E is called* **closed** *if $P_{ij} = 0$ for all $i \in C$ and $j \notin C$. If a closed set consists of a single state it is called an* **absorbing** *state.*

Definition 2.30. *A Markov chain is called* **irreducible** *if there exists no nonempty closed set other than E itself. If E has proper closed subsets, it is called* **reducible**.

Definition 2.31. *Two states, i and j, are said to* **intercommunicate** *if for some $n \geq 0$, $P_{ij}^{(n)} > 0$, and for some $m \geq 0$, $P_{ji}^{(m)} > 0$.*

This definition says that it is possible for the chain to go from i to j in n steps, and it is possible to go from j to i in m steps. The integers n and m need not be the same.

Theorem 2.4. *A Markov chain is irreducible if and only if all pairs of states intercommunicate.*

The period of state i is concerned with the times at which the chain might return to state i.

Definition 2.32. *Let i be a state of a Markov chain; i is said to have* **period** *d if d is the greatest common divisor of those values of n for which $P_{ii}^{(n)} > 0$.*

Definition 2.33. *Let f_{ij}^n denote the probability that the* **first visit** *to state j from state i occurs at time n. That is*

$$f_{ij}^n = P\{X_n = j,\ X_v \neq j,\ v = 1, 2, \ldots, n-1 | X_0 = i\}.$$

If $i = j$ we refer to f_{ii}^n as the probability that the first return to state i occurs at time n. By definition we say $f_{ij}^0 = f_{ii}^0 = 0$.

Definition 2.34. *For fixed states i and j, let $f_{ij}^* = \sum_{n=1}^{\infty} f_{ij}^n$. The symbol f_{ij}^* represents the probability of* **ever visiting** *state j from state i. If $i = j$, we let $f_{ii}^* = \sum_{n=1}^{\infty} f_{ii}^n$ denote the probability of* **ultimately returning** *to state i.*

Definition 2.35. *A state i is said to be* **recurrent** *if $f_{ii}^* = 1$. If $f_{ii}^* < 1$, then i is called* **transient**.

Definition 2.36. *If $f_{ii}^* = 1$, define the* **expected return time** *to state i as $\mu_i = \sum_{n=1}^{\infty} n f_{ii}^n$.*

Definition 2.37. *A recurrent state with an infinite expected return time is called* **null recurrent**. *If the expected return time is finite, the state is called* **positive recurrent**.

We have introduced all the necessary definitions for the following important theorem.

Theorem 2.5. *For an irreducible, positive recurrent aperiodic Markov chain $\lim_{n\to\infty} P_{ij}^{(n)}$ exists and is independent of i. Furthermore, letting*

$$\pi_j = \lim_{n\to\infty} P_{ij}^{(n)}, \quad j \geq 0$$

then π_j is the only nonnegative solution of

$$\begin{aligned} \pi_j &= \sum_{i=0}^{\infty} \pi_i P_{ij} \quad j \geq 0 \\ \sum_{j=0}^{\infty} \pi_j &= 1. \end{aligned}$$

2.3.2 Absorption Times and Absorption Probabilities

Consider a Markov chain whose states are labeled $0, 1, \ldots, N$. States $0, 1, \ldots, r-1$ are absorbing, and here $P_{ii} = 1$ for $0 \leq i \leq r-1$, while states $r, \ldots, N$ are transient in that $P_{ij}^{(n)} \to 0$ as $n \to \infty$ for $r \leq i, j \leq N$. Therefore the transition matrix associated with the Markov chain is:

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{Q} & \mathbf{R} \end{bmatrix},$$

where $\mathbf{I}_r$ is the identity matrix of dimension r. Matrix $\mathbf{Q}$ has dimension $(N-r+1) \times r$ and its entries represent the probability of going from a transient state to an absorbing state. Finally we describe matrix $\mathbf{R}$, which is a matrix of dimension $(N-r+1) \times (N-r+1)$ and contains the probabilities of going from one transient state to another transient state. The set of transient states is denoted by R and $H = E - R$ denotes the set of absorbing states.

Definition 2.38. *Let $0, \ldots, r-1$ be the set of absorbing states of a Markov chain. Then the* **absorption** *time starting from the i^{th} transient state or the* **first hitting time to the set of absorbing states** *starting from the i^{th} transient state is given by the random variable*

$$\tau_i = \min\{n; n \geq 0, 0 \leq X_n \leq r-1 | X_0 = i\}.$$

We are interested in the expectation of the random variable τ_i. Since it always holds that $\tau_i = 0$ for any $i \in H$, we only need to concentrate on the states outside set H.

Definition 2.39. *The* **expected absorption time** *starting from the i^{th} transient state or the* **expected first hitting time to the set of absorbing states** *starting from the i^{th} transient state is*

$$m_i = \mathsf{E}[\tau_i; \tau_i < \infty] \ .$$

The vector whose components represent the expected first hitting time depending on the initial transient state i, is $\mathbf{m} = [m_i]_{i \in R}$. It is proven (see Theorem 3.2 in Iosifescu (1980)) that this vector can be calculated as follows:

$$\mathbf{m} = (\mathbf{I}_{|R|} - \mathbf{R})^{-1}\mathbf{1} \ ,$$

where $\mathbf{1}$ denotes the $|R|$-dimensional vector $(1, \ldots, 1)^t$.

If we denote $\mathbf{W} = (\mathbf{I}_{|R|} - \mathbf{R})^{-1}$, based on the above comments we can state that the expected absorption time m_i starting from the i^{th} state is given by the expression:

$$m_i = \sum_j w_{ij}. \tag{2.1}$$

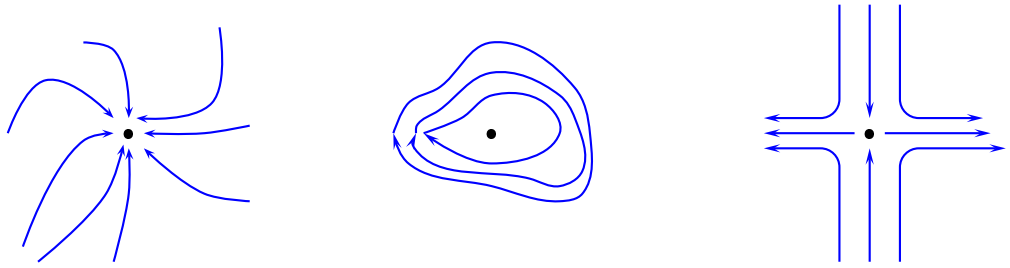


Figure 2.1. Fixed points with three different types of stability. The fixed point on the left is stable. The fixed point in the center is marginally stable. The fixed point on the right is unstable.

Definition 2.40. *Let* $0, \ldots, r-1$ *be the set of absorbing states of a Markov chain. Then the* **absorption probability** *to an absorbing state* j, $0 \leq j \leq r-1$ *starting from the* i^{th} *state is*

$$u_{ij} = \lim_{n \to \infty} P\{\tau_i \leq n, X_{\tau_i} = j | X_0 = i\}.$$

The absorption probabilities, u_{ij}, to an absorbing state j starting from the i^{th} state, are given by the elements of the matrix $\mathbf{U} = (u_{ij})$:

$$\mathbf{U} = \mathbf{WQ}. \tag{2.2}$$

2.4 Discrete Dynamical Systems

Finally we present the last mathematical tool used in this dissertation for the theoretical study of EDAs: discrete dynamical systems. A good reference for beginners in dynamical systems is Sheinerman (1996).

Definition 2.41. *Let* $f : \mathbb{R}^n \to \mathbb{R}^n$ *be a function. An equation of the form:*

$$\mathbf{x}(k+1) = f(\mathbf{x}(k))$$

is a **discrete dynamical system**.

A discrete dynamical system is a recurrence relation, with the index k playing the role of a discrete "time". The vector $\mathbf{x}$ is the state of the dynamical system, and the function f tells us how the system moves. In special circumstances, however, the system does not move. The system can be fixed in a special state; we call these states fixed points.

Definition 2.42. *A* **fixed point** *of a dynamical system is a state vector* $\widetilde{\mathbf{x}}$ *with the property that if the system is ever in the state* $\widetilde{\mathbf{x}}$*, it will remain in that state for all time.*

To find fixed points of the discrete dynamical system $\mathbf{x}(k+1) = f(\mathbf{x}(k))$ we need to find the solutions $\widetilde{\mathbf{x}}$ of the equation $\mathbf{x} = f(\mathbf{x})$.

Not all fixed points are the same. Let us describe three types of fixed points a system may possess.

Definition 2.43. *Let* $\mathbf{x}(k+1) = f(\mathbf{x}(k))$ *be a discrete dynamical system. A fixed point* $\widetilde{\mathbf{x}}$ *of the discrete dynamical system is called* **stable** *provided that:*

(i) For every positive number ϵ *there exists* $\delta > 0$ *such that if* $|\widetilde{\mathbf{x}} - \mathbf{x}(0)| < \delta$*, then* $|\widetilde{\mathbf{x}} - \mathbf{x}(k)| < \epsilon$ *for all* $k \geq 0$*.*

(ii) There exists $\delta > 0$ *so that for any* $\mathbf{x}(0)$ *such that* $|\widetilde{\mathbf{x}} - \mathbf{x}(0)| < \delta$*, then for every* $\epsilon > 0$ *there is* $K > 0$ *so that if* $k \geq K$*, then* $|\widetilde{\mathbf{x}} - \mathbf{x}(k)| < \epsilon$*.*

Part (i) of definition 2.43 means that for all starting values $\mathbf{x}(0)$ near $\widetilde{\mathbf{x}}$, the system stays near $\widetilde{\mathbf{x}}$, while part (ii) says that for all starting values $\mathbf{x}(0)$ near $\widetilde{\mathbf{x}}$, $\mathbf{x}(k) \to \widetilde{\mathbf{x}}$ as $k \to \infty$, i.e. the system converges to $\widetilde{\mathbf{x}}$.

Definition 2.44. *Let* $\mathbf{x}(k+1) = f(\mathbf{x}(k))$ *be a discrete dynamical system. A fixed point* $\widetilde{\mathbf{x}}$ *of the discrete dynamical system is called* **marginally stable or neutral** *provided that:*

(i) For every positive number ϵ *there exists* $\delta > 0$ *such that if* $|\widetilde{\mathbf{x}} - \mathbf{x}(0)| < \delta$*, then* $|\widetilde{\mathbf{x}} - \mathbf{x}(k)| < \epsilon$ *for all* $k \geq 0$*.*

(ii) There exists $\delta > 0$ *so that for any* $\mathbf{x}(0)$ *such that* $|\widetilde{\mathbf{x}} - \mathbf{x}(0)| < \delta$*, then there exists* $\epsilon > 0$ *and* $K > 0$ *so that if* $k \geq K$*, then* $|\widetilde{\mathbf{x}} - \mathbf{x}(k)| > \epsilon$*.*

In other words, a fixed point $\widetilde{\mathbf{x}}$ is marginally stable if (i) for all starting values $\mathbf{x}(0)$ near $\widetilde{\mathbf{x}}$, the system stays near $\widetilde{\mathbf{x}}$, but (ii) for all starting values $\mathbf{x}(0)$ near $\widetilde{\mathbf{x}}$, the system does not converge to $\widetilde{\mathbf{x}}$.

Definition 2.45. *Let* $\mathbf{x}(k+1) = f(\mathbf{x}(k))$ *be a discrete dynamical system. A fixed point* $\widetilde{\mathbf{x}}$ *of the discrete dynamical system is called* **unstable** *if it is neither stable nor marginally stable.*

A fixed point can also be said to be unstable if there are starting values $\mathbf{x}_0$ very near $\widetilde{\mathbf{x}}$ so that the system moves far away from $\widetilde{\mathbf{x}}$.

Figure 2.1 illustrates each of these possibilities. The fixed point on the left of the figure is stable; all trajectories which begin near $\widetilde{\mathbf{x}}$ remain near, and converge to $\widetilde{\mathbf{x}}$. The fixed point in the center of the figure is marginally stable (neutral); trajectories which begin near $\widetilde{\mathbf{x}}$ stay nearby but never converge to $\widetilde{\mathbf{x}}$. Finally, the fixed point on the right of the figure is unstable. There are trajectories which start near $\widetilde{\mathbf{x}}$ and move far away from $\widetilde{\mathbf{x}}$.

The following proposition gives us a test to classify the stable or unstable fixed points of a discrete multidimensional nonlinear dynamical system (see Sheinerman (1996), pp. 126).

Proposition 2.1. *Let* $\widetilde{\mathbf{x}}$ *be a fixed point of the discrete dynamical system* $\mathbf{x}(k+1) = f(\mathbf{x}(k))$*, and* $Df(\widetilde{\mathbf{x}})$ *the Jacobian evaluated at* $\widetilde{\mathbf{x}}$*. It follows that:*

(i) If the eigenvalues λ *of the Jacobian all are such that* $|\lambda| < 1$*, then* $\widetilde{\mathbf{x}}$ *is a stable fixed point.*

(ii) If some eigenvalue λ *of the Jacobian is such that* $|\lambda| > 1$*, then* $\widetilde{\mathbf{x}}$ *is an unstable fixed point.*

Chapter 3

An Introduction to Estimation of Distribution Algorithms

This chapter provides an introduction to Estimation of Distribution Algorithms.

Section 3.1 offers a brief review of Evolutionary Algorithms, where the appearance of Estimation of Distribution Algorithms is motivated. In Section 3.2 Estimation of Distribution Algorithms proposed for the solution of combinatorial optimization problems and optimization in continuous domains are reviewed, ordered according to the complexity of the interrelations that they are able to express. In addition, in Section 3.2, recent approaches in this field are reviewed and briefly summarized.

3.1 Estimation of Distribution Algorithms: A novel class of Evolutionary Algorithms

Evolutionary Algorithm (EAs) is an umbrella term used to describe computer-based problem solving systems which use computational models of some of the known mechanisms of natural evolution (established by Darwin (1859)) as key elements in their design and implementation. They all share a common conceptual base of simulating the evolution of individual structures via processes of selection and random changes. These robust heuristic approaches have been successfully applied in several different fields such as all kinds of optimization problems, searching, classification and machine learning. Here we focus on the field of optimization.

A formal description of an optimization problem (in case of minimization) is given by:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}).$$

The set Ω, called the *search space*, can be finite or infinite and can be defined by a set of constraints. The optimization problem can have many levels of difficulty, for instance multimodality, non-linearity, nondifferentiability or complicated restrictions. The function to be optimized is termed the *objective function*.

It is common practice to use biological terms to describe the algorithms: An admissible solution $\mathbf{x} \in \Omega$ (and possibly some other information) is encoded into a data structure called an *individual*. A collection of such individuals forms a *population*. So-called *evolutionary operators* like *mutation* and *crossover* are used to modify individuals (the *parents*) at random yielding the *offspring*. Then the *selection* operator determines which offspring will serve as the new parents of the next iteration according to the objective function value of the solution encoded in each individual. For maximization problems the objective function value may be interpreted as a measure of *fitness* of the

individuals in the environment. The larger the objective function value, the higher the fitness of the individual. In case of minimization problems this metaphor does not carry over because the fitness of the individual is to be minimized. In order to avoid confusion the term "fitness" will be used here as a synonym of the term "objective function value". Since

$$-\left\{\min_{\mathbf{x}\in\Omega} f(\mathbf{x})\right\} = \max_{\mathbf{x}\in\Omega}\{-f(\mathbf{x})\}$$

this convention does not cause any problems. The algorithmic frame of EAs is sketched in Figure 3.1.

```
EA
  choose an initial population
  determine the fitness of each individual

  repeat

    perform random variation
    determine the fitness of each individual
    perform selection

  until some stopping criterion is satisfied
```

Figure 3.1. Pseudocode for a general EA.

A number of major schools of evolutionary algorithms have evolved over the last 30 years:

- Genetic Algorithms: mainly developed in the USA by Holland (1975), a set of earlier related works can be seen in Fogel (1998), and further developments are described in Goldberg (1989). Although Genetic Algorithms were initially applied in problems of combinatorial optimization, their development also reached the numerical field.
- Evolution Strategies: developed in Germany by Rechenberg (1973) and Schwefel (1981), they are applied to problems in continuous domains.
- Evolutionary Programming: has its roots in the work carried out by Fogel (1962, 1964) in the USA. It was initially applied to discrete optimization, but later its applications also spread to the numerical field.

Readers interested in reviews of these and other EAs are encouraged to peruse the literature: Goldberg (1989), Fogel (1994), Bäck (1996), Rudolph (1997), and Larrañaga and Lozano (2002).

If EAs are compared with standard methods of mathematical optimization, EAs have the following advantages. First, the field of application of EAs is vaster, they can deal with multimodalities, discontinuities and constraints, with noisy functions, multiple criteria decision-making processes or with problems given by a simulation model. Second, they do not make any assumptions about the search space. Third, EAs can be relatively easily adapted to a new problem. Finally, the parameters of EAs can be changed during execution.

Of course EAs also have drawbacks, including the following. First, optimal solutions are not guaranteed to be found, and there are no reliable stopping criteria. Second, even extremely simplified

algorithms are difficult to analyze theoretically. This difficulty implies a deficiency in theoretical foundations. Third, EAs are computationally expensive and it is very difficult to carry out a comparison between different EA approaches except experimentally. Fourth, it is not possible to know how far the solution obtained by the algorithm is from the global optimum of the optimization problem in hand, predicting the movements of the populations in the search space is extremely difficult. Fifth, some optimization problems may require the design of specific random operators. However, the worst characteristic probably resides in their strong dependence on a set of parameters (e.g., size of the population, number of generations, probabilities of applying the random operators, rate of generational reproduction, etc.) which have to be tuned experimentally for the particular problem in hand. Consequently, unless the user has experience resolving the concrete optimization problem in hand by means of a particular EA, choosing the suitable values for all the parameters itself becomes an optimization problem (Grefenstette, 1986). Evolution strategies are the most common EAs that we have mentioned that do not suffer from this drawback, as the parameter tuning process is done using a self-adaptation technique. As a result, it is important to understand that EAs are not a set of techniques that are ready to be applied, but a set of mechanisms that have to be modified and tailored to the optimization problem in hand. Nevertheless, the good results obtained in practical optimization problems justify the exponential growth of research about EAs.

Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2002; Mühlenbein and Paaβ, 1996) constitute a new and promising paradigm for EAs. EDAs have been proposed in a wide variety of scenarios with the purpose of overcoming some of the drawbacks that classical EAs present. Concretely, the most appealing advantage of EDAs over classical EAs is the reduction in the number of parameters to be tuned or assessed by the user. This reduction in the number of parameters stems from the non-existence of random operators needed to generate the next population from the current one at each iteration. This also avoids the sometimes necessary design of random operators tailored to some particular optimization problems. EDAs replace the application of random operators at each iteration by learning and subsequent simulation of a joint probability distribution for a database made up of some individuals that are selected from the current population. This results in a further advantage of EDAs over classical EAs: The relationships between the random variables that represent the genes of every individual selected from the current population are explicitly expressed through the joint probability distribution learnt from them, instead of being implicitly kept by the individuals of the successive populations as building blocks of an efficient and effective search. In fact, Holland (1975) already recognized that detecting interacting genes would be beneficial to GAs. This source of knowledge was called *linkage information*. This idea has been exploited by many researchers for the last few years in order to enhance the performance of GAs. The reader is referred to Goldberg (1989), Goldberg et al. (1993), Kargupta (1996), Kargupta and Goldberg (1996), Bandyopadhyay et al. (1998), Lobo et al. (1998), van Kemenade (1998), and Bosman and Thierens (1999) for further details.

Another advantage of EDAs over classical EAs such as GAs is concerned with their performance in what are known as *deceptive optimization problems*. It is well known that GAs exhibit poor performance in this class of optimization problems as application of the random operators does not guarantee that the combination of good solutions results in better solutions. According to the experimental results that many works report (e.g., Larrañaga and Lozano (2002)), EDAs naturally overcome the difficulties that are derived from deceptive optimization problems and provide the user with effective final solutions.

Before explaining the main steps of an EDA it is necessary to introduce some notation. Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ denotes an n-dimensional random variable where each unidimensional random

EDA

$D_0 \leftarrow$ Generate M individuals (the initial population) at random

Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met

$D_{t-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{t-1} according to the selection method

$\rho_t(\mathbf{x}) = \rho(\mathbf{x}|D_{t-1}^{Se}) \leftarrow$ Estimate the joint probability distribution of an individual being among the selected individuals

$D_t \leftarrow$ Sample M individuals (the new population) from $\rho_t(\mathbf{x})$

Figure 3.2. Pseudocode of the generic EDA.

variable X_i is associated with one of the n genes of an individual. Consequently, $\boldsymbol{X}$ is an n-dimensional discrete random variable and $\rho(\boldsymbol{x}) = p(\boldsymbol{x})$ is a joint probability mass function for $\boldsymbol{X}$ when dealing with discrete optimization problems. On the other hand, $\boldsymbol{X}$ is an n-dimensional continuous random variable and $\rho(\boldsymbol{x}) = f(\boldsymbol{x})$ is a joint probability density function for $\boldsymbol{X}$ when dealing with continuous optimization problems.

Figure 3.2 shows a schematic of the generic EDA. The algorithm consists of the iteration of three main steps, after the individuals of the initial population D_0 have been generated, usually at random, and evaluated. These three steps are as follows for the t^{th} iteration of the generic EDA for all t. First, N of the M individuals of the current population D_{t-1} are selected according to the selection method (usually based on the objective function values of the individuals). Then, the population of selected individuals, denoted by D_{t-1}^{Se}, is used to induce a joint probability distribution for $\boldsymbol{X}$, $\rho_t(\boldsymbol{x})$. Finally, M individuals are sampled from $\rho_t(\boldsymbol{x})$ and evaluated in order to create the new population D_t. These three steps (i.e., selection of promising individuals, induction of a joint probability distribution over the selected individuals, and subsequent simulation of the joint probability distribution in order to construct the new population) are repeated until the stopping criterion is met. Examples of stopping criteria are: Performance of a fixed number of generations, performance of a fixed number of evaluations of the objective function, uniformity in the current population, or no improvement over the best individual obtained in the previous generation. The final solution that is returned at the end of the problem optimization process is usually the best solution found in the whole process. Note that this solution may not be in the population of the last generation of the generic EDA when the selection method takes a *non-elitist* approach.

3.2 Review of Estimation of Distribution Algorithms

The bottleneck of this new heuristic lies in estimating the joint probability distribution associated with the database containing the selected individuals. Obviously, computing all the parameters that are needed to completely specify this joint probability distribution in the standard representation is often impractical. To avoid this problem, several authors have proposed different algorithms where simplified assumptions concerning the conditional (in)dependencies between the variables of the joint probability distribution are made. Therefore, classification of the algorithms can be

carried out taking the complexity of the probabilistic model used into account: from those that suppose that all the variables are independent, followed by those that consider order-two statistics, and finally those that, by means of probabilistic graphical models, do not consider any constraints on the relationship between the variables of the problem. A review of the different approaches in the combinatorial and numerical fields can be found in Larrañaga et al. (1999), Larrañaga et al. (2000a), Larrañaga et al. (2000b), Pelikan et al. (1999b).

Next, we offer a review of the EDAs proposed for the solution of combinatorial optimization problems and optimization in continuous domains, and also we review the latest approaches in this field. The work by Larrañaga and Lozano (2002) is an excellent, sound review of EDAs. The authors also present a considerable number of optimization problems, ranging from classical ones to those that appear in the machine learning field, which are solved with the help of a wide selection of the EDAs mentioned above.

3.2.1 EDA Approaches to Combinatorial Optimization

The different approaches presented are ordered according to the complexity of the probabilistic model used to learn the interdependencies between the variables from the database of selected individuals.

3.2.1.1 Without Dependencies

In all the algorithms belonging to this category it is assumed that the n-dimensional joint probability distribution factorizes as a product of n univariate and independent probability distributions. That is $p_t(\mathbf{x}) = \prod_{i=1}^{n} p_t(x_i)$ – see Figure 3.6 for a graphical representation. Obviously this assumption is very far from what happens in a difficult optimization problem, where the interdependencies among the variables usually exist.

UMDA

The Univariate Marginal Distribution Algorithm (UMDA) was proposed by Mühlenbein (1998). UMDA uses the simplest model to estimate the joint probability distribution of the selected individuals at each generation, $p_t(\mathbf{x})$. This joint probability distribution is factorized as a product of independent univariate marginal distributions:

$$p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) = \prod_{i=1}^{n} p_t(x_i)$$

A pseudocode for the UMDA algorithm can be seen in Figure 3.3.

Usually each univariate marginal distribution is estimated from marginal frequencies as follows:

$$p_t(x_i) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i|D_{t-1}^{Se})}{N}$$

where

$$\delta_j(X_i = x_i|D_{t-1}^{Se}) = \begin{cases} 1 & \text{if in the } j^{th} \text{ case of } D_{t-1}^{Se},\ X_i = x_i \\ 0 & \text{otherwise} \end{cases}$$

PBIL

The Population Based Incremental Learning Algorithm (PBIL) was introduced by Baluja (1994)

UMDA

$D_0 \leftarrow$ Generate M individuals (the initial population)

Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met

$D_{t-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{t-1} according to the selection method

$p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i | D_{t-1}^{Se})}{N} \leftarrow$ Estimate the joint probability distribution

$D_t \leftarrow$ Sample M individuals (the new population) from $p_t(\mathbf{x})$

Figure 3.3. Pseudocode for a general UMDA algorithm.

and further improved by Baluja and Caruana (1995). The objective of this algorithm is to obtain the optimum of a function defined in the binary space $\Omega = \{0, 1\}^n$. PBIL is based on the idea of substituting the individuals of a population by a set of their statistics. In each generation, the population of individuals is represented by a vector of probabilities:

$$\mathbf{p}_t = (p_1^{(t)}, \ldots, p_i^{(t)}, \ldots, p_n^{(t)}) \tag{3.1}$$

where $p_i^{(t)}$ refers to the probability of obtaining a value of 1 in the i^{th} component of D_t, the population of individuals in the t^{th} generation.

The algorithm works as follows. At each step t, drawing the probability vector $p_t(\mathbf{x})$, λ individuals are obtained and the μ best of them ($\mu \leq \lambda$), $\mathbf{x}_{1:\lambda}^{(t)}, \mathbf{x}_{2:\lambda}^{(t)}, \ldots, \mathbf{x}_{\mu:\lambda}^{(t)}$, are selected. These selected individuals will be used to modify the probability vector. A Hebbian-inspired rule is used to update the probability vector:

$$\mathbf{p}_{t+1} = (1 - \alpha)\mathbf{p}_t + \alpha \frac{1}{\mu} \sum_{k=1}^{\mu} \mathbf{x}_{k:\lambda}^{(t)}$$

where $\alpha \in (0, 1]$ is an algorithm's parameter. Figure 3.4 shows a pseudocode for the PBIL algorithm.

cGA

Harik et al. (1998) present an algorithm called compact Genetic Algorithm (cGA) that also belongs to this family. The algorithm (for binary representations) begins by initializing a vector of probabilities where each component follows a Bernouilli distribution with parameter 0.5. Next, two individuals are generated randomly from this vector of probabilities. After the individuals are evaluated, a competition between them is carried out. The competition is held at the level of each of the unidimensional variables, in such a way that if for the i^{th} position the conquering individual takes a value different from the loser, then the i^{th} component of the vector of probabilities increases or diminishes by a constant amount which depends on whether the i^{th} position of the conquering individual was a one or a zero. This process of adaptation of the vector of probabilities towards

PBIL

Obtain an initial probability vector $\mathbf{p}_0$

while no convergence **do**
begin

Using $\mathbf{p}_t$ obtain λ individuals $\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_\lambda^{(t)}$

Evaluate and rank $\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_\lambda^{(t)}$

Select the $\mu \leq \lambda$ best individuals $\mathbf{x}_{1:\lambda}^{(t)}, \mathbf{x}_{2:\lambda}^{(t)}, \ldots, \mathbf{x}_{\mu:\lambda}^{(t)}$

$\mathbf{p}_{t+1} = (1-\alpha)\mathbf{p}_t + \alpha\frac{1}{\mu}\sum_{k=1}^{\mu}\mathbf{x}_{k:\lambda}^{(t)}$
end

Figure 3.4. Pseudocode for the PBIL algorithm.

the winning individual continues until the vector of probabilities has converged. Figure 3.5 shows a pseudocode for the cGA.

Figure 3.6 is a graphical representation of the probability model of EDAs without interdependencies.

3.2.1.2 Bivariate Dependencies

Estimation of the joint probability distribution can be done quickly without assuming independence between the variables – which is very far from reality in some problems – by taking dependencies between pairs of variables into account. In this case, it is enough to consider second-order statistics. While in the algorithms of the previous section, learning of just the parameters was carried out – the structure of the model remained fixed – in this subsection, parametric learning is extended to structural learning too.

MIMIC

De Bonet et al. (1997) developed an algorithm called Mutual Information Maximization for Input Clustering (MIMIC). If we denote by $\pi = (i_1, i_2, \ldots, i_n)$ a permutation of the indexes $1, 2, \ldots, n$, the probabilistic model used by MIMIC can be written as:

$$p_t^\pi(\mathbf{x}) = p_t(x_{i_1}|x_{i_2}) \cdot p_t(x_{i_2}|x_{i_3}) \cdot \ldots \cdot p_t(x_{i_{n-1}}|x_{i_n}) \cdot p_t(x_{i_n})$$

To do the structural learning MIMIC uses a greedy algorithm that searches at each generation for the best permutation π among the variables, in order to minimize the Kullback-Leibler divergence between $p_t^\pi(\mathbf{x})$ and the empirical distribution of the set of selected points. Figure 3.7 shows a pseudocode for the estimation of the joint probability distribution carried out by MIMIC algorithm where $H(X) = -\sum_x \mathsf{P}(X = x)\log\mathsf{P}(X = x)$ denotes the Shannon entropy of the variable X, and $H(X\,|\,Y) = \sum_y \mathsf{P}(Y = y)H(X\,|\,Y = y)$, where $H(X\,|\,Y = y) = -\sum_x \mathsf{P}(X = x, Y = y)\log\mathsf{P}(X = x\,|\,Y = y)$, denotes the mean uncertainty in X given Y.

As in UMDA, the parameters are estimated by means of maximum likelihood estimators.

cGA

Step 1. Initialize the probability vector $p_0(\boldsymbol{x})$
$p_0(\boldsymbol{x}) = (p_0(x_1), \ldots, p_0(x_i), \ldots, p_0(x_n)) =$
$(0.5, \ldots, 0.5, \ldots, 0.5)$

Step 2. $t = t + 1$. Sampling $p_t(\boldsymbol{x})$ with $t = 0, 1, 2, \ldots$ obtain two individuals: $\boldsymbol{x}_1^{(t)}, \boldsymbol{x}_2^{(t)}$

Step 3. Evaluate and rank $\boldsymbol{x}_1^{(t)}$ and $\boldsymbol{x}_2^{(t)}$ obtaining:
$\boldsymbol{x}_{1:2}^{(t)}$ (the best of both) and $\boldsymbol{x}_{2:2}^{(t)}$ (the worst of both)

Step 4. Update the probability vector $p_t(\boldsymbol{x})$ towards $\boldsymbol{x}_{1:2}^{(t)}$
for $i = 1$ to n
if $x_{i,1:2}^{(t)} \neq x_{i,2:2}^{(t)}$ then
if $x_{i,1:2}^{(t)} = 1$ then $p_t(x_i) = p_{t-1}(x_i) + \frac{1}{K}$
if $x_{i,1:2}^{(t)} = 0$ then $p_t(x_i) = p_{t-1}(x_i) - \frac{1}{K}$

Step 5. Check if the probability vector $p_t(\boldsymbol{x})$ has converged
for $i = 1$ to n do
if $p_t(x_i) > 0$ and $p_t(x_i) < 1$ then
return to Step 2

Step 6. $p_t(\boldsymbol{x})$ represents the final solution

Figure 3.5. Pseudocode for the cGA.

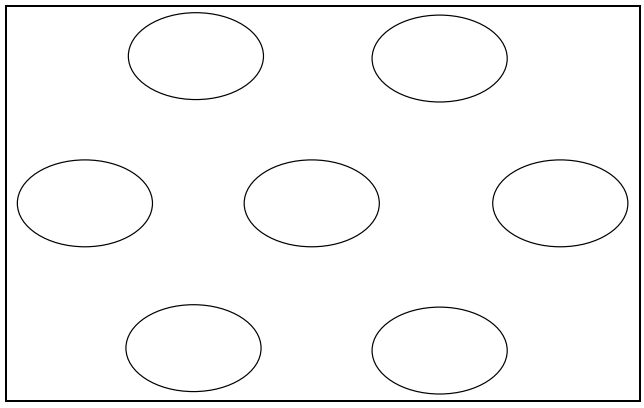


Figure 3.6. Graphical representation of the probability model of the proposed EDAs in combinatorial optimization without interdependencies (UMDA, PBIL, cGA).

COMIT and TREE

Baluja and Davies (1997a, 1997b) proposed an algorithm called COMIT. This algorithm uses a probabilistic model that considers second-order statistics. The dependency structure between the

Step 1. Choose $i_n = \arg\min_j \hat{H}_t(X_j)$

Step 2. **for** $k = n-1, \ldots, 1$

Choose $i_k = \arg\min_j \hat{H}_t(X_j \mid X_{i_{k+1}})$
$j \neq i_{k+1} \ldots, i_n$

Step 3. $p_t^{\pi}(\mathbf{x}) = p_t(x_{i_1}|x_{i_2}) \cdot p_t(x_{i_2}|x_{i_3}) \cdot \ldots \cdot p_t(x_{i_{n-1}}|x_{i_n}) \cdot p_t(x_{i_n})$

Figure 3.7. The MIMIC approach to estimation of the joint probability distribution at generation t. The symbols $\widehat{H}_t(X)$ and $\widehat{H}_t(X|Y)$ denote the empirical entropy of X and the empirical entropy of X given Y respectively. Both are estimated from D_t^{Se}.

variables forms a tree. The factorization produced in the probability distribution can be written as:

$$p_t(\mathbf{x}) = \prod_{i=1}^{n} p_t(x_i|x_{j(i)})$$

where $X_{j(i)}$ is the variable (possibly empty) on which X_i depends.

Estimation of the tree structure of the probability distribution of the selected individuals at each generation is done using the algorithm proposed by Chow and Liu (1968) (see Figure 3.8). The parameters, given the structure, are calculated by maximum likelihood estimation.

Step 1. From the given (observed) distribution $p(\mathbf{x})$ calculate $I(X_i, X_j)$, the mutual information for all variable pairs

Step 2. Consider the $n(n-1)/2$ mutual information values as branch weights of a complete graph and order them by magnitude

Step 3. Assign the two largest branches to the tree to be constructed

Step 4. Examine the next largest branch, and add it to the tree unless it forms a loop, in which case discard it and examine the next largest branch

Step 5. Repeat Step 4 until $n-1$ branches have been selected (at this point the spanning tree has been constructed)

Step 6. $p_t(\mathbf{x})$ can be computed by selecting an arbitrary root node and forming the product: $p_t(\mathbf{x}) = \prod_{i=1}^{n} p_t(x_i|x_{j(i)})$

Figure 3.8. The Chow and Liu algorithm.

In the original work COMIT applied a local optimizer to each generated individual. The algorithm in which this step is eliminated is called TREE (Larrañaga and Lozano, 2002).

BMDA

Pelikan and Mühlenbein (1999) propose a factorization of the joint probability distribution that only

needs second-order statistics. Their approach, BMDA (Bivariate Marginal Distribution Algorithm), is based on the construction of a dependency graph, which is always acyclic but does not necessarily have to be a connected graph. In fact the dependency graph can be seen as a set of trees that are not mutually connected.

The basic idea underlying the algorithm for construction of the dependency graph is simple. First, an arbitrary variable is chosen and added as a node of the graph. This first variable is the one with the greatest dependency – measured by Pearson's χ^2 statistic. Second, we need to add to the graph the variable with the greatest dependency between any of the previously incorporated variables and the set of not yet added variables. This last step is repeated until there is no dependency surpassing a previously fixed threshold between already added variables and the rest. If this is the case, a variable is chosen at random from the set of variables not yet used. The whole process is repeated until all variables have been added to the dependency graph.

In each generation the factorization obtained with the BMDA is given by:

$$p_t(\boldsymbol{x}) = \prod_{X_r \in R_t} p_t(x_r) \prod_{X_i \in \mathbf{X} \setminus R_t} p_t(x_i \mid x_{j(i)})$$

where $\mathbf{X}$ denotes the set of n variables, R_t denotes the set containing the root variable – in generation t – for each of the connected components of the dependency graph, and $X_{j(i)}$ returns the variable connected to the variable X_i and added before X_i.

The probabilities for the root nodes, $p_t(x_r)$, as well as the conditional probabilities, $p_t(x_i \mid x_{j(i)})$, are estimated from the database, D_{t-1}^{Se}, containing the selected individuals.

Figure 3.9 is a graphical representation of EDAs with pairwise dependencies.

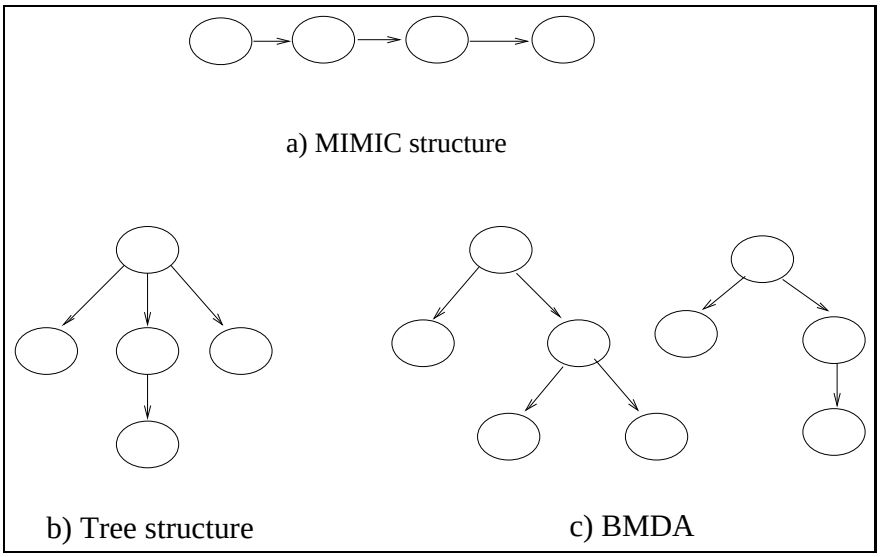


Figure 3.9. Graphical representation of the probability models for the proposed EDAs in combinatorial optimization with pairwise dependencies (MIMIC, tree structure, BMDA).

3.2.1.3 Multiple Dependencies

Several approaches to EDAs have been proposed in the literature where factorization of the joint probability distribution requires statistics of order greater than two.

As far as we know, the first work in which the possibility of adapting the methods of model induction developed by the scientific community working on probabilistic graphical models to EDAs approaches is that of Baluja and Davies (1997a). This possibility is mentioned again in their later work (Baluja and Davies, 1998), but unfortunately, they only mention it and do not show evidence of implementation.

EcGA

Harik (1999) presents an algorithm – Extended compact Genetic Algorithm (EcGA) – whose basic idea consists of using a marginal product model to estimate the joint probability distribution of the selected individuals in each generation. This means that, in each generation, factorization of the joint probability distribution is done as a product of marginal distributions of variable size. These marginal distributions of variable size are related to the variables that are contained in the same group and to the probability distributions associated with them. The grouping is carried out using a greedy forward algorithm that obtains a partition between the n variables. Each group of variables is assumed to be independent of the rest – as shown in Figure 3.13. In this way, factorization of the joint probability on the n variables is of the form:

$$p_t(\boldsymbol{x}) = \prod_{c \in C_t} p_t(\boldsymbol{x}_c) \tag{3.2}$$

where C_t denotes the set of groups in the t^{th} generation, and $p_t(\boldsymbol{x}_c)$ represents the marginal distribution of the variables $\boldsymbol{X}_c$, i.e., the variables that belong to the c^{th} group in the t^{th} generation.

As the EcGA obtains a partition of the set of variables, it turns out that for all t and for all $c, k \in C_t$:

$$\bigcup_{c \in C_t} \boldsymbol{X}_c = \{X_1, \ldots, X_n\}, \quad \boldsymbol{X}_c \cap \boldsymbol{X}_k = \emptyset.$$

The greedy algorithm that carries out the grouping begins with a partition with n clusters (a variable in each cluster). Then, the algorithm performs the union of the two variables that results in the greatest reduction of a measure that conjugates the sum of the entropies of the marginal distributions with a penalty for the complexity of the model based on the minimum description length principle (MDL) (Rissanen, 1978).

More precisely, the measure that the EcGA tries to minimize in each generation has two components:

- The *compressed population complexity* defined using the entropy of the marginal distributions, as follows:

$$N \sum_{c \in C_t} H(\boldsymbol{X}_c) = -N \sum_{c \in C_t} \sum_{\boldsymbol{x}_c} p(\boldsymbol{X}_c = \boldsymbol{x}_c) \log p(\boldsymbol{X}_c = \boldsymbol{x}_c)$$

  and

EcGA

$D_0 \leftarrow$ Generate M individuals (the initial population) at random

Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met

$D_{t-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{t-1} using the tournament selection method

$p_t(\boldsymbol{x}) = p(\boldsymbol{x}|D_{t-1}^{Se}) = \prod_{c \in C_t} p_t(\boldsymbol{x}_c|D_{t-1}^{Se}) \leftarrow$ Estimate the probability distribution of the selected individuals by means of a marginal product model. Model search using steepest ascent search, minimizing:
$-N \sum_{c \in C_t} \sum_{\boldsymbol{x}_c} p(\boldsymbol{X}_c = \boldsymbol{x}_c) \log p(\boldsymbol{X}_c = \boldsymbol{x}_c) + \log N \sum_{c \in C_t} \dim \boldsymbol{X}_c$

$D_t \leftarrow$ Sample M individuals (the new population) from $p_t(\boldsymbol{x})$

Figure 3.10. Pseudocode for the EcGA.

- The *model complexity* that takes into account the dimension of the model in this way:

$$\log N \sum_{c \in C_t} \dim \boldsymbol{X}_c$$

where $\dim \boldsymbol{X}_c$ represents the number of parameters needed to specify the marginal distribution of $\boldsymbol{X}_c$. If all the unidimensional variables belonging to the c^{th} group were binary, then we would obtain $\dim \boldsymbol{X}_c = 2^{|\boldsymbol{X}_c|} - 1$.

Taking into account both components, the measure that EcGA tries to minimize in each generation is:

$$-N \sum_{c \in C_t} \sum_{\boldsymbol{x}_c} p(\boldsymbol{X}_c = \boldsymbol{x}_c) \log p(\boldsymbol{X}_c = \boldsymbol{x}_c) + \log N \sum_{c \in C_t} \dim \boldsymbol{X}_c.$$

This measure is called *combined complexity* by Harik (1999).

The greedy search used by EcGA begins each generation by postulating that all the variables are independent. It performs a steepest ascent search, where at each step, the algorithm attempts to merge each pair of groups into a larger group. It judges the merit of these merges on their combined complexity. If the best combination leads to a decrease in combined complexity, then that merger is carried out. This process continues until no further pairs of groups can be merged. The resulting marginal product model is then the one that is used for that generation.

As can be seen in Figure 3.10, tournament selection is also used in each generation to obtain the set of selected individuals.

FDA

In the work of Mühlenbein et al. (1999) the FDA (Factorized Distribution Algorithm) is introduced. This algorithm applies to additively decomposed functions for which, using the running intersection property (Lauritzen, 1996), a factorization of the mass-probability based on residuals, $\boldsymbol{x}_{b_i}$, and separators, $\boldsymbol{x}_{c_i}$, is obtained.

A function $h(\boldsymbol{x})$ is additively decomposed if:

$$h(\boldsymbol{x}) = \sum_{s_i \in S} h_i(\boldsymbol{x}_{s_i})$$

where the set $S = \{s_1, \ldots s_k\}$ with $s_i \subset \{1, \ldots, n\}$ constitutes a covering of $\{1, \ldots, n\}$, and the following sets:

$$\begin{array}{l} d_i = \cup_{j=1}^{i} s_j \\ b_i = s_i \backslash d_{i-1} \\ c_i = s_i \cap d_{i-1} \end{array}$$

satisfy these three conditions:

- $b_i \neq \emptyset$ for all $i = 1, \ldots, k$
- $d_k = \{1, 2, \ldots, n\}$
- $\forall\ i \geq 2\ \exists\ j < i$ such that $c_i \subseteq s_j$.

Then the joint probability distribution can be factorized in this way:

$$p_t(\boldsymbol{x}) = \prod_{i=1}^{k} p_t(\boldsymbol{x}_{b_i} | \boldsymbol{x}_{c_i}).$$

This factorization remains valid for all the iterations. Changes are only in the estimation of the probabilities that in each iteration are done from the database containing the selected individuals. In any case, the requirement of specifying the factorization of the joint probability distribution is a drawback in applying the FDA approach to generic optimization problems. It is for this reason that, besides parametric learning, structural learning is also desirable.

PADA

In Soto et al. (1999) factorization is done using a Bayesian network with polytree structure (no more than one undirected path connecting every pair of variables). The proposed algorithm is called PADA (Polytree Approximation of Distribution Algorithms) and can be considered a hybrid between a method for detecting conditional (in)dependencies and a procedure based on score+search.

EBNA

The Estimation of Bayesian Networks Algorithm (EBNA) was introduced by Etxeberria and Larrañaga (1999) and Larrañaga et al. (2000a). A similar algorithm was independently proposed by Pelikan y col. (Pelikan and Goldberg, 2000a; Pelikan and Goldberg, 2000b; Pelikan et al., 1999; Pelikan et al., 2000c).

This algorithm allows statistics of unrestricted order in the factorization of the joint probability distribution. This distribution is encoded by a Bayesian network that is learnt from the database containing the selected individuals at each generation.

Let us briefly introduce Bayesian networks. Formally a Bayesian network (Castillo et al., 1997) is a pair $(S, \boldsymbol{\theta})$ representing a graphical factorization of a probability distribution. The structure S is a directed acyclic graph which reflects the set of conditional (in)dependencies among the variables. The factorization of the probability distribution is codified by S:

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

where $\mathbf{Pa}_i$ is the parent set of X_i (variables from which there exists an arc to X_i in the graph S). On the other hand $\boldsymbol{\theta}$ is a set of parameters for the local probability distributions associated with each variable. If the variable X_i has r_i possible values, $x_i^1, \ldots, x_i^{r_i}$, the local distribution $p(x_i|\mathbf{pa}_i^j, \boldsymbol{\theta}_i)$ is an unrestricted discrete distribution:

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

where $\mathbf{pa}_i^1, \ldots, \mathbf{pa}_i^{q_i}$ denote the values of $\mathbf{Pa}_i$ and the term q_i denotes the number of possible different instances of the parent variables of X_i. In other words, the parameter θ_{ijk} represents the conditional probability of variable X_i being in its k^{th} value, knowing that the set of its parent variables is in its j^{th} value. Therefore, the local parameters are given by $\boldsymbol{\theta}_i = (((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i})$. An example of a Bayesian network can be seen in Figure 3.11.

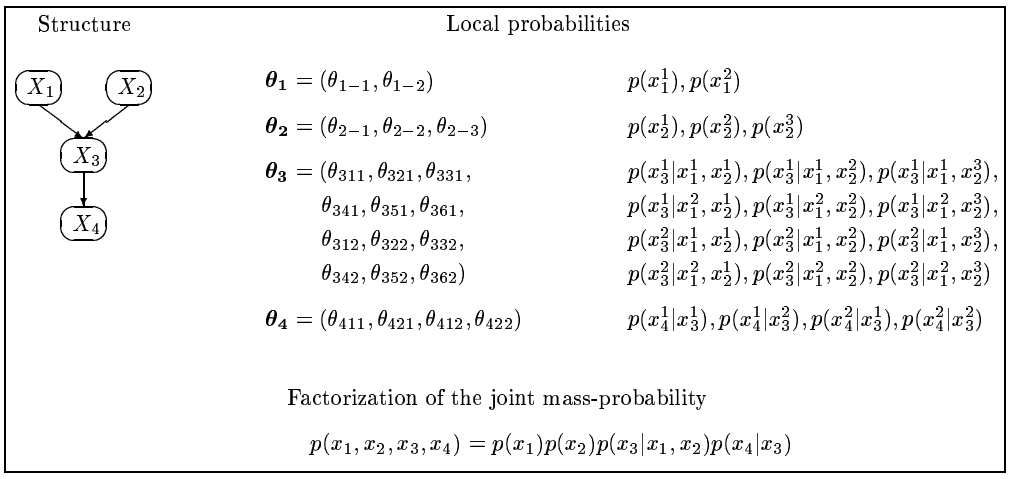


Figure 3.11. Example of a Bayesian network (X_1, X_2 and X_4 can take two possible values and X_3 three).

In EBNA learning the probabilistic model at each generation of the algorithm means learning a Bayesian network from the selected individuals. There are different strategies to learn the structure of a Bayesian network: by detecting conditional (in)dependencies or with a method called "score + search". Once the structure has been learnt, the conditional probability distributions required to completely specify the model are estimated from the database – using some of the different approaches to parameter learning – or are given by an expert.

Detecting conditional (in)dependencies Every algorithm that tries to recover the structure of a Bayesian network by detecting (in)dependencies has some conditional (in)dependence relations

between some subset of variables of the model as input, and a directed acyclic graph that represents a large percentage (and even all of them if possible) of these relations as output. The PC algorithm, introduced by Spirtes et al. (1991), starts by forming the complete undirected graph, then "thins" that graph by removing edges with zero order conditional independence relations, "thins" again with first order conditional independence relations, and so on. The set of variables that are conditioned on need only to be a subset of the set of variables adjacent to one of the variables being conditioned. Using the PC algorithm produces EBNA$_{PC}$.

Score + search method In "score + search" method, given a database D and a Bayesian network whose structure is denoted by S, a value which evaluates how well the Bayesian network represents the probability distribution of the database D is assigned. Different EBNA algorithms can be obtained by using different scores.

- BIC score (based on penalized maximum likelihood)

$$BIC(S,D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{1}{2}\log N\sum_{i=1}^{n} q_i(r_i-1)$$

  where N_{ij} is the number of individuals in D in which variables $\boldsymbol{Pa}_i^S$ take their j^{th} value and N_{ijk} is the number of individuals in D in which variable X_i takes its k^{th} value and variables $\boldsymbol{Pa}_i^S$ take their j^{th} value (this produces EBNA$_{BIC}$).

- K2+pen (based on a penalized Bayesian score)

$$K2+pen(S,D) = \log\left[\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!\right] - \frac{1}{2}\log N\sum_{i=1}^{n} q_i(r_i-1)$$

  (this produces EBNA$_{k2+pen}$).

Notice that in the second score the original $K2$ score (Cooper and Herskovits, 1992) has been modified by adding a penalization term in order to favor simple structures.

Once we have defined a score to evaluate Bayesian networks we have to set a search process to find the Bayesian network that maximizes the score given the set of selected individuals. As we need to find an adequate model structure as quickly as possible, a simple algorithm which returns a good structure, even if not optimal, is preferred. An interesting algorithm with these characteristics is Algorithm B (Buntine, 1991). Algorithm B is a greedy search which starts with an arc-less structure and, at each step, adds the arc with the maximum improvement in the score. The algorithm finishes when there is no arc whose addition improves the score.

In an EBNA approach the simulation of the Bayesian network is done using the *Probabilistic Logic Sampling* (PLS) method, proposed by Henrion (1998). Figure 3.12 represents a pseudocode of the EBNA algorithm.

BOA

Pelikan et al. (1999a, 2000a, 2000b) and Pelikan and Goldberg (2000c) propose the BOA (Bayesian Optimization Algorithm). BOA uses the BDe (Bayesian Dirichlet equivalence) metric to measure the goodness of each structure. This Bayesian metric has the property that the score of two structures

---

EBNA$_{PC}$, EBNA$_{BIC}$ and EBNA$_{K2+pen}$

$BN_0 \leftarrow (S_0, \boldsymbol{\theta}^0)$ where S_0 is an arc-less DAG, and $\boldsymbol{\theta}^0$ is uniform $p_0(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} \frac{1}{r_i}$

$D_0 \leftarrow$ Sample M individuals from $p_0(\mathbf{x})$

for $t = 1, 2, \ldots$ until the stopping criterion is met

$D_{t-1}^{Se} \leftarrow$ Select N individuals from D_{t-1}

$S_t^* \leftarrow$ Find the best structure according to a criterion:
conditional (in)dependencies tests $\rightarrow$ EBNA$_{PC}$
penalized maximum likelihood+search $\rightarrow$ EBNA$_{BIC}$
penalized Bayesian score+search $\rightarrow$ EBNA$_{K2+pen}$

$\boldsymbol{\theta}^t \leftarrow$ Calculate θ_{ijk}^l using D_{t-1}^{Se} as the data set

$BN_t \leftarrow (S_t^*, \boldsymbol{\theta}^t)$

$D_t \leftarrow$ Sample M individuals from BN_t using PLS

Figure 3.12. Pseudocode for the EBNA$_{PC}$, EBNA$_{BIC}$ and EBNA$_{K2+pen}$ algorithms.

that reflect the same conditional (in)dependencies is the same. The search used is a greedy search and it starts in each generation from scratch. In order to reduce the cardinality of the search space the constraint that each node in the Bayesian network has at most k parents is assumed. In Schwarz and Ocenasek (1999) some empirical comparisons between BOA and BMDA can be found.

LFDA, FDA$_L$, FDA-BC, FDA-SC

Mühlenbein and Mahnig (1999) introduce the LFDA (Learning Factorized Distribution Algorithm), which essentially follows the same approach as in EBNA$_{BIC}$. The main difference is that in the LFDA the complexity of the learnt model is controlled by the BIC measure in conjunction with a restriction on the maximum number of parents each variable can have in the Bayesian network.

Ochoa et al. (1999) propose an initial algorithm, FDA$_L$, to learn – by means of conditional (in)dependence tests – a junction tree – an undirected graph derived from the Bayesian network structure – from a database. The underlying idea is to return the junction tree that best satisfies the previous assertions once a list of dependencies and independencies between the variables is obtained.

Also in Ochoa et al. (2000a) a structure learning algorithm that takes into account questions of reliability and computational cost is presented. The algorithm, called FDA-BC, studies the class of Factorized Distribution Algorithm with Bayesian networks of bounded complexity.

Similar ideas are introduced by Ochoa et al. (2000b) in the FDA-SC. In this case factorization of the joint probability distribution is done using simple structures, i.e. trees, forests or polytrees.

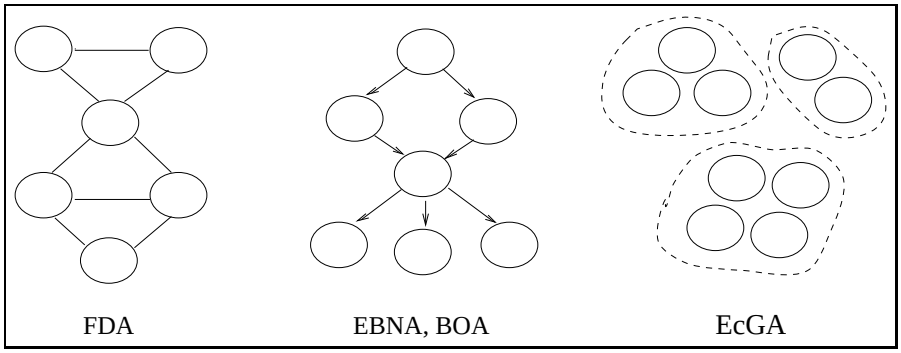


Figure 3.13. Graphical representation of probability models for the proposed EDAs in combinatorial optimization with multiple dependencies (FDA, EBNA, BOA, LFDA and EcGA).

3.2.2 EDA Approaches in Continuous Domains

In this section we review work on optimization in continuous domains with EDAs. The organization of the section is analogous to the previous one as the different approaches have again been grouped by the complexity of the interdependencies between the variables that the learnt density function is able to express. It reviews work where the density function is factorized as a product of n univariate marginal densities, after which an approach that uses two order densities is presented. Finally more general approaches that consider multiple dependencies are introduced.

3.2.2.1 Without Dependencies

In work that does not take into account dependencies between the variables it is usual to assume that the joint density function follows an n-dimensional normal distribution, which is factorized by a product of unidimensional and independent normal densities. Using the mathematical notation $\boldsymbol{X} \equiv \boldsymbol{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \sum)$, this is:

$$f_{\boldsymbol{\mathcal{N}}}(\boldsymbol{x}; \boldsymbol{\mu}, \sum) = \prod_{i=1}^{n} f_{\mathcal{N}}(x_i; \mu_i, \sigma_i^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}(\frac{x_i - \mu_i}{\sigma_i})^2}. \tag{3.3}$$

UMDA$_c$

The Univariate Marginal Distribution Algorithm for continuous domains (UMDA$_c$) was introduced by Larrañaga et al. (1999, 2000b). In every generation and for every variable the UMDA$_c$ carries out some statistical tests in order to find the density function that best fits the variable. Note that in this case, although the factorization of the joint density function is

$$f_t(\boldsymbol{x}; \boldsymbol{\theta}^t) = \prod_{i=1}^{n} f_t(x_i, \boldsymbol{\theta}_i^t)$$

the UMDA$_c$ is in fact a structure identification algorithm (something that does not happen with the UMDA for the discrete case) in the sense that the density components of the model are identified via

UMDA$_c$
Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met
for $i := 1$ to n do
(i) Select via hypothesis test the density function $f_t(x_i; \boldsymbol{\theta}_i^t)$ that best fits D_{t-1}^{Se,X_i}, the projection of the selected individuals over the i^{th} variable
(ii) Obtain the maximum likelihood estimates for $\boldsymbol{\theta}_i^t = (\theta_i^{t,k_1}, \ldots, \theta_i^{t,k_i})$
At each generation the learnt joint density function is expressed as:
$f_t(\boldsymbol{x}; \boldsymbol{\theta}^t) = \prod_{i=1}^n f_t(x_i, \widehat{\boldsymbol{\theta}_i^t})$

Figure 3.14. Pseudocode for learning the joint density function in UMDA$_c$.

hypothesis tests. The estimation of parameters is performed, once the densities have been identified, by their maximum likelihood estimates.

If all the univariate distributions are normal, then the two parameters to be estimated at each generation and for each variable are the mean, μ_i^t, and the standard deviation, σ_i^t. It is well known that their respective maximum likelihood estimates are:

$$\widehat{\mu_i^t} = \overline{X_i}^t = \frac{1}{N}\sum_{r=1}^{N} x_{i,r}^t; \qquad \widehat{\sigma_i^t} = \sqrt{\frac{1}{N}\sum_{r=1}^{N}(x_{i,r}^t - \overline{X_i}^t)^2} \ .$$

This particular case of the UMDA$_c$ will be denoted UMDA$_c^G$ (Univariate Marginal Distribution Algorithm for Gaussian models).

Figure 3.14 shows pseudocode for learning the joint density function in the UMDA$_c$.

SHCLVND

Rudlof and Köppen (1996), in their SHCLVND (Stochastic Hill Climbing with Learning by Vectors of Normal Distributions), estimate the joint density function as a product of unidimensional and independent normal densities. The vector of means $\boldsymbol{\mu} = (\mu_1, ..., \mu_i, ..., \mu_n)$ is adapted by means of the Hebbian rule:

$$\boldsymbol{\mu}^{(t+1)} = \boldsymbol{\mu}^{(t)} + \alpha \cdot (\boldsymbol{b}^{(t)} - \boldsymbol{\mu}^{(t)})$$

where $\boldsymbol{\mu}^{(t+1)}$ denotes the vector of means in the generation $t+1$, α denotes the learning rate, and $\boldsymbol{b}^{(t)}$ denotes the baricenter of the B (an amount fixed at the beginning) best individuals in the t^{th} generation. Adaptation of the vector of variances $\boldsymbol{\sigma}$ is carried out using a reduction policy in the following way:

$$\boldsymbol{\sigma}^{(t+1)} = \boldsymbol{\sigma}^{(t)} \cdot \beta$$

where β denotes a previously fixed constant ($0 < \beta < 1$).

PBIL$_c$

Sebag and Ducoulombier (1998) propose an extension (PBIL$_c$) of the boolean PBIL algorithm to continuous spaces. As with the previous authors, they assume a joint density function that follows a Gaussian distribution factorizable as a product of unidimensional and independent marginal

densities. The adaptation of each component of the vector of means is carried out using the following formula:

$$\mu_i^{(t+1)} = (1-\alpha)\cdot\mu_i^{(t)} + \alpha\cdot(x_{ibest,1}(t) + x_{ibest,2}(t) - x_{iworst}(t))$$

where $\mu_i^{(t+1)}$ represents the i^{th} component of the mean, $\boldsymbol{\mu}^{(t+1)}$, at generation $(t+1)$, $x_{ibest,1}(t)$ denotes the best individual of generation t, while $x_{ibest,2}(t)$ and $x_{iworst}(t)$ denote respectively the second best and the worst individual of the generation t, and α is a constant. For the adaptation of the vector of variances, they propose four heuristics: (i) use a constant value for all the marginals and all the generations; (ii) adjust it as in a $(1,\lambda)$ evolution strategy; (iii) calculate the sample variance of the K best individuals of each generation; (iv) by means of a Hebbian rule, similar to the adaptation of the means.

Notice the similarity of this approach to the $(1,\lambda)$–ES (Schwefel, 1995).

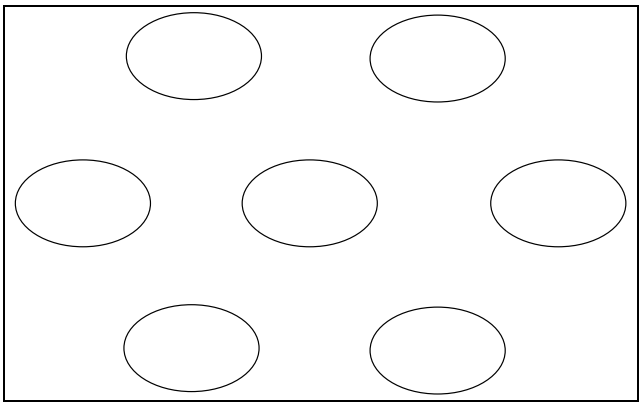


Figure 3.15. Graphical representation of the probability models for the proposed EDAs for optimization in continuous domains without dependencies between the variables (UMDA$_c$, SHCLVND, PBIL$_c$).

Servet et al.

Servet et al. (1997) introduce a progressive approach to the problem. At each generation, and for each dimension i, they store an interval $({a_i}^t, {b_i}^t)$ and a real number ${z_i}^t$ $(i = 1,\ldots,n)$. ${z_i}^t$ represents the probability that the i^{th} component of a solution is on the right half of the previous interval. In every generation the probabilities, ${z_i}^t$, for each dimension are calculated, and when ${z_i}^t$ is closer to 1(0) than a previously fixed quantity, the interval is reduced to its right (left) half.

3.2.2.2 Bivariate Dependencies

MIMIC$_c^G$

This approach was introduced by Larrañaga et al. (1999, 2000b) and constitutes an adaptation of the MIMIC algorithm (de Bonet et al., 1997) to continuous domains where the underlying probability model for every pair of variables is assumed to be a bivariate Gaussian.

The idea, as in MIMIC for combinatorial optimization, is to describe the true joint density function by fitting the model as closely as possible to the empirical data by using only one univariate

---

MIMIC_c^G
Step 1. Choose $i_n = \arg\min_j \hat{\sigma}_{X_j}$
Step 2. **for** $k = n-1, n-2, \ldots, 1$

$$\text{Choose } i_k = \arg\min_j \hat{\sigma}_{X_j} - \frac{\hat{\sigma}^2_{X_j X_{i_{k+1}}}}{\hat{\sigma}^2_{X_{i_{k+1}}}}$$

$j \neq i_{k+1}, \ldots, i_n$

Figure 3.16. Adaptation of the MIMIC approach to a multivariate Gaussian density function.

marginal density and $n-1$ pairwise conditional density functions. In order to do this, the following result is used:

Theorem 3.1 (Whittaker, 1990; pp. 167)**.** *Let $\boldsymbol{X}$ be an n-dimensional normal density function, $\boldsymbol{X} \equiv \boldsymbol{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \sum)$, then the entropy of $\boldsymbol{X}$ is:*

$$h(\boldsymbol{X}) = \frac{1}{2} n(1 + \log 2\pi) + \frac{1}{2} \log |\sum| . \tag{3.4}$$

Applying this result to univariate and bivariate normal density functions in order to define MIMIC_c^G, we obtain:

$$h(X) = \frac{1}{2}(1 + \log 2\pi) + \log \sigma_X \tag{3.5}$$

$$h(X \mid Y) = \frac{1}{2}\left[(1 + \log 2\pi) + \log(\frac{\sigma^2_X \sigma^2_Y - \sigma^2_{XY}}{\sigma^2_Y})\right] \tag{3.6}$$

where $\sigma^2_X (\sigma^2_Y)$ denotes the variance of the univariate $X(Y)$ variable and σ_{XY} denotes the covariance between the variables X and Y.

Structure learning in MIMIC_c^G – see Figure 3.16 – works as a straightforward greedy algorithm with two steps. In the first step, the variable with the smallest sample variance is chosen. In the second step, the variable X with the smallest estimation of $\frac{\sigma^2_X \sigma^2_Y - \sigma^2_{XY}}{\sigma^2_Y}$ with respect to the previous chosen variable, Y, t is chosen and linked to Y.

3.2.2.3 Multiple Dependencies

In this section we introduce some approaches to EDAs for continuous domains in which the density function learnt at each generation is not restricted. As a first approach we present a model where the density function corresponds to a non restricted multivariate normal density that is learnt from scratch at each generation. The following two models correspond to adaptation and incremental versions of the former model respectively. We also present approaches based on learning of Gaussian networks by edge exclusion tests, and also using two score+search approaches to learn the appropriate Gaussian network at each generation. We finish the section with the IDEA framework.

EMNA$_{global}$

This is an approach based on the estimation of a multivariate normal density function at each generation. As we can see in Figure 3.17, at each generation we estimate the vector of means,

EMNA$_{global}$
$D_0 \leftarrow$ Generate M individuals (the initial population) at random

Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met

$D_{t-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{t-1} according to the selection method

$f_t(\boldsymbol{x}) = f(\boldsymbol{x}|D_{t-1}^{Se}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \leftarrow$ Estimate the multivariate normal density function from the selected individuals

$D_t \leftarrow$ Sample M individuals (the new population) from $f_t(\boldsymbol{x})$, using an adaptation of the PLS algorithm to continuous domains

Figure 3.17. Pseudocode for the EMNA$_{global}$ approach.

$\boldsymbol{\mu}_t = (\mu_{1,t}, \ldots, \mu_{n,t})$, and the variance–covariance matrix, $\boldsymbol{\Sigma}_t$, whose elements are denoted by $\sigma_{ij,t}^2$ with $i, j = 1, \ldots, n$. This means that we need to estimate $2n + \begin{pmatrix} n-1 \\ 2 \end{pmatrix}$ parameters at each generation: n means, n variances and $\begin{pmatrix} n-1 \\ 2 \end{pmatrix}$ covariances. These parameter estimations use their maximum likelihood estimates in the following way:

$$\hat{\mu}_{i,t} = \overline{X}_i^t = \frac{1}{N} \sum_{r=1}^{N} x_{i,r}^t \quad i = 1, \ldots, n$$

$$\hat{\sigma}_{i,t}^2 = \frac{1}{N} \sum_{r=1}^{N} (x_{i,r}^t - \overline{X}_i^t)^2 \quad i = 1, \ldots, n$$

$$\hat{\sigma}_{ij,t}^2 = \frac{1}{N} \sum_{r=1}^{N} (x_{i,r}^t - \overline{X}_i^t)(x_{j,r}^t - \overline{X}_j^t) \quad i, j = 1, \ldots, n \quad i \neq j.$$

Although the number of parameters that this approach needs to estimate at each generation is greater than in the case where the joint density function is estimated by means of Gaussian networks, the mathematics needed to develop this approach are very simple. Note also that in the approach based on Gaussian networks where edge exclusion tests are used it is necessary to calculate the same number of parameters as in this approach and then carry out a hypothesis test on them. On the other hand in the approach where score+search is used in order to obtain the best Gaussian network it is mandatory to do a search over the space of possible models.

EMNA$_a$

We present in this section an adaptive version of the approach introduced in the previous one. Pseudocode for EMNA$_a$ (Estimation of Multivariate Normal Algorithm adaptive) is shown in Figure 3.18.

Once we obtain the first model, $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, whose parameters are estimated from the individuals selected from the initial population, the flow of EMNA$_a$ is similar to a steady–state genetic algorithm.

EMNA$_a$

$D_0 \leftarrow$ Generate M individuals (the initial population) at random
Select $N \leq M$ individuals from D_0 according to the selection method

Obtain the first multivariate normal density $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$

Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met

Generate an individual $\boldsymbol{x}_{ge}^t$ from $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$
if $\boldsymbol{x}_{ge}^t$ is better than the worst individual, $\boldsymbol{x}^{t,N}$, **then**
add $\boldsymbol{x}_{ge}^t$ to the population and drop $\boldsymbol{x}^{t,N}$ from it
Obtain $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$

Figure 3.18. Pseudocode for the EMNA$_a$ approach.

First, we simulate one individual from the current multivariate normal density model. Next, we compare the goodness of this simulated individual with the worst individual maintained in the population. If the fitness of the simulated individual is better than the worst individual we have in the population, then we replace this worst individual with the simulated individual. Once a new individual replaces an existing one, it is necessary to update the parameters of the multivariate normal density function.

This update will be done using the following formulae which can be obtained by simple algebraic manipulations:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{1}{N}(\boldsymbol{x}_{ge}^t - \boldsymbol{x}^{t,N})$$

$$\sigma_{ij,t+1}^2 = \sigma_{ij,t}^2 - \frac{1}{N^2}(x_{ge,i}^t - x_i^{t,N}) \cdot \sum_{r=1}^{N}(x_j^{t,r} - \mu_j^t) - \frac{1}{N^2}(x_{ge,j}^t - x_j^{t,N}) \cdot \sum_{r=1}^{N}(x_i^{t,r} - \mu_i^t) +$$

$$\frac{1}{N^2}(x_{ge,i}^t - x_i^{t,N})(x_{ge,j}^t - x_j^{t,N}) - \frac{1}{N}(x_i^{t,N} - \mu_i^{t+1})(x_j^{t,N} - \mu_j^{t+1}) +$$

$$\frac{1}{N}(x_{ge,i}^t - \mu_i^{t+1})(x_{ge,j}^t - \mu_j^{t+1}).$$

Note that with this EMNA$_a$ approach, the size of the population of individuals is kept fixed in every generation.

EMNA$_i$

In this section we describe the EMNA$_i$ (Estimation of Multivariate Normal Algorithm incremental) approach. The similarity of this approach to the previous one is that both approaches only generate one individual from each model. The difference is that in EMNA$_i$ each of the generated individuals is added – as can be seen in Figure 3.19 – to the population.

This procedure uses the following updating rules:

EMNA$_i$

$D_0 \leftarrow$ Generate M individuals (the initial population) at random
Select $N \leq M$ individuals from D_0 according to the selection method

Obtain the first multivariate normal density $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$

Repeat for $t = 1, 2, \ldots$ until the stopping criterion is met

Generate an individual $\boldsymbol{x}_{ge}^t$ from $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$

Add $\boldsymbol{x}_{ge}^t$ to the population

Obtain $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$

Figure 3.19. Pseudocode for the EMNA$_i$ approach.

$$\boldsymbol{\mu}_{t+1} = \frac{N_t}{N_t + 1}\boldsymbol{\mu}_t + \frac{1}{N_t + 1}\boldsymbol{x}_{ge}^t$$

$$\sigma_{ij,t+1}^2 = \frac{N_t}{N_t + 1}\sigma_{ij,t}^2 + \frac{1}{N_t + 1}(x_{ge,i}^t - \mu_i^t)(x_{ge,j}^t - \mu_j^t).$$

Notice that in EMNA$_i$ the number of individuals in the population increases as the algorithm evolves.

EGNA$_{ee}$, EGNA$_{BGe}$, EGNA$_{BIC}$

One proposal for optimization in continuous domains based on the learning and simulation of Gaussian networks is illustrated in Figure 3.21.

Here we introduce one example of the probabilistic graphical model paradigm where it is assumed that the joint density function is a multivariate Gaussian density (Whittaker, 1990). This paradigm is known as the Gaussian network paradigm (Shachter and Kenley, 1989) in which the number of parameters needed to specify a multivariate Gaussian density is reduced. In this graphical model case, each variable $X_i \in \boldsymbol{X}$ is continuous and each local density function is the linear-regression model:

$$f(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \equiv \mathcal{N}(x_i; m_i + \sum_{x_j \in \boldsymbol{Pa}_i} b_{ji}(x_j - m_j), v_i) \tag{3.7}$$

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

Interpretation of the components of the local parameters is as follows: m_i is the unconditional mean of X_i, v_i is the conditional variance of X_i given $\boldsymbol{Pa}_i$, and b_{ji} is a linear coefficient reflecting the strength of the relationship between X_j and X_i. Figure 3.20 is an example of a Gaussian network in a 4–dimensional space.

The basic steps of each iteration of EGNA algorithms are:

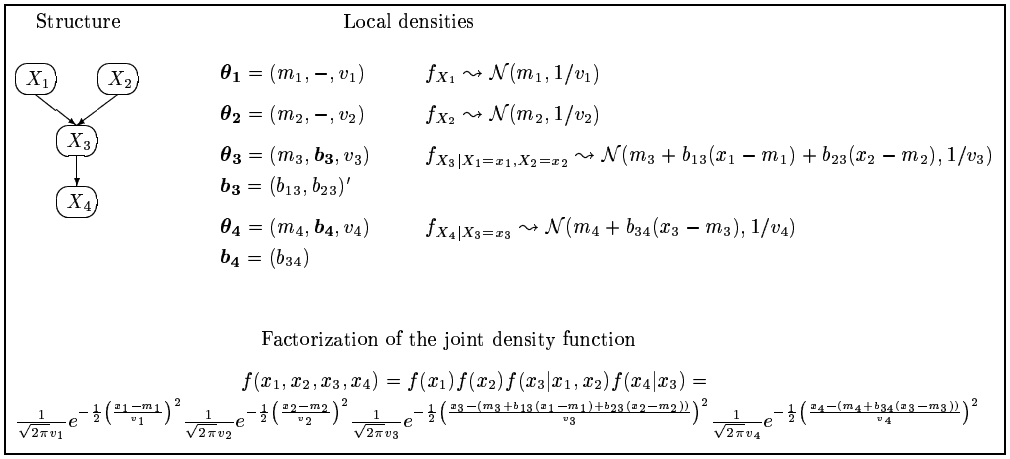


Figure 3.20. Structure, local densities and resulting factorization for a Gaussian network with four variables.

EGNA$_{ee}$, EGNA$_{BGe}$, EGNA$_{BIC}$

For $t = 1, 2, \ldots$ until the stopping criterion is met
$D_{t-1}^{Se} \leftarrow$ Select Se individuals from D_{t-1}

(i) $\hat{S}_t \leftarrow$ Structural learning via:
edge exclusion tests $\rightarrow$ EGNA$_{ee}$
Bayesian score+search $\rightarrow$ EGNA$_{BGe}$
penalized maximum likelihood+search $\rightarrow$ EGNA$_{BIC}$

(ii) $\hat{\boldsymbol{\theta}}^t \leftarrow$ Calculate the estimates for the parameters of $\hat{S}_t$

(iii) $M_t \leftarrow (\hat{S}_t, \hat{\boldsymbol{\theta}}^t)$

(iv) $D_t \leftarrow$ Sample M individuals from M_t using the continuous version of the PLS algorithm

---

Figure 3.21. Pseudocode for the EGNA$_{ee}$, EGNA$_{BGe}$, and EGNA$_{BIC}$ algorithms.

- Learning the Gaussian network structure by using one of the different methods: edge–exclusion tests (Smith and Whittaker, 1998), Bayesian score+search, penalized maximum likelihood+search.
- Computation of estimates for the parameters of the learnt Gaussian network structure.
- Creation of the Gaussian network model.
- Simulation of the joint probability distribution function encoded by the Gaussian network learnt in the previous steps. For this last step, we use an adaptation of the PLS algorithm to continuous domains.

While in the EGNA$_{ee}$ the Gaussian network is induced at each generation by means of edge exclusion tests, the model induction in the EGNA$_{BGe}$ and EGNA$_{BIC}$ is carried out by score+search

approaches. In the EGNA$_{BGe}$ a Bayesian score that gives the same value for Gaussian networks reflecting the same conditional (in)dependencies is used.

IDEA

Bosman and Thierens (1999, 2000) introduce the IDEA (Iterated Density Evolutionary Algorithms) approach to EDAs. Although IDEA is a general framework that can be used in discrete and continuous optimization, the proposed new approaches belong to continuous domains.

There are two main characteristics of IDEA. The first one is that in each generation individuals are sampled from a truncated distribution, where the truncation point is given by the worst individual found in the previous generation. The second characteristic is that only part of the population is replaced in each generation.

3.2.3 Recent Approaches for EDAs

EDAs have aroused the interest of a number of researchers as can be seen by the high number of works recently published in conferences and journals in this field. This section briefly summarizes and revises the latest main conferences on this topic – i.e., PPSN, CEC and GECCO conferences. We also include other works published in EC-related journals. The works have been classified into three sets: those devoted to a theoretical analysis of EDAs; those creating new EDAs; and works in which EDAs are applied. In addition, the work by Lozano et al. (2005) is an excellent reference for bringing oneself up to date on the latest approaches to EDAs.

3.2.3.1 Theoretical Analysis

In the work by Zlochin and Dorigo (2002) similarities and distinctive features between Ant Colony Optimization, Stochastic Gradient Ascent, Cross-Entropy and Estimation of Distribution Algorithms are discussed.

Toussaint (2003) presents a theoretical study of the structure of the offspring probability distribution, or exploration distribution, for a GA with mutation only, a GA with crossover, and an EDA. The results afford a precise characterization of the structure of the crossover exploration distribution. Essentially, the crossover operator destroys mutual information between loci by transforming it into entropy; it does the inverse of correlated exploration. In contrast, the objective of EDAs is to model the mutual information between loci in the fitness distribution, thereby inducing correlated exploration.

Zhang (2004) studies the advantages of using higher order statistics in EDAs. The author analyzes two EDAs with two-tournament selection for discrete optimization problems, UMDA, which uses only first-order statistics and FDA, which uses higher order statistics. The heuristic functions and the limit models of these two algorithms are introduced and stability of these limit models is analyzed. It is shown that the limit model of UMDA can be trapped at any local optimal solution for some initial probability models. However, degenerate probability density functions at some local optimal solutions are unstable in the limit model of FDA. In particular, the degenerate probability density function at the global optimal solution is the unique asymptotically stable point in the limit model of FDA for the optimization of an additively decomposable function. The results suggest that using higher order statistics could improve the chance of finding the global optimal solution.

The global convergence of EDAs in continuous domains is investigated in Zhang and Mühlenbein (2004) , proving that: i) if the distribution of the new elements matches that of the parent set exactly, the algorithms will converge to the global optimum under three widely used selection schemes and, ii) a factorized distribution algorithm converges globally under proportional selection.

Shapiro (2005) analyzes the drift phenomenon in EDAs. The author deals with this effect when the estimation of the parameters of the probability model in the EDA is not carried out by means of a soft estimator. Some ideas to avoid this phenomenon involve scaling of the algorithm's parameters in a strongly problem-dependent way. The author exemplifies the result using the size of the population as a parameter for several common algorithms and objective functions.

Gao and Culberson (2005) investigate the complexity issues related to the representation of sampling distributions in EDAs. Particularly, they pay attention to additive decomposable functions and the graph produced by these functions. Since the probability models of some EDAs are based on these graphs, they analyze the complexity of these kinds of graphs. Their results provide insight into how to design efficient EDAs and what the limitations of the algorithms are.

3.2.3.2 New Algorithms

We have classified the works that deal with the development of new EDAs in three groups. The first one contains works that propose new probability models for learning the independencies between the variables from the data base of selected individuals. The second one contains works on how new EDAs are used to efficiently solve particular problems. Finally, the third one contains works that combine EDAs and other techniques.

New Probability Models

Bosman and Thierens (2002) give a new tool for finding and using structure of permutation problems in evolutionary computation by estimating marginal product factorized probability distributions in the space of permutations. This new algorithm is called ICEE (Induced Chromosome Elements Exchanger).

Focusing mainly on the effective construction of Bayesian Networks with decision trees in the distributed environment, Oceanesek et al. (2003) suggest and simulate a new algorithm called Multithreaded Mixed Bayesian Optimization Algorithm (MMBOA).

The Gaussian-model based PBIL is improved in Yuan and Gallager (2003), proposing a new continuous PBIL employing a histogram probabilistic model.

With the aim of keeping the diversities in an EDAs population, Handa (2004) incorporates a mutation operator into conventional EDAs, creating new EDAs.

In Wright et al. (2004), a new framework for EDAs based on the principle of maximum entropy and conservation of schema frequencies is proposed.

Tsuji et al. (2004) combine linkage identification methods and EDAs to detect dependency between loci.

Cho and Zhang (2004) suggest a new continuous EDA with the variational Bayesian independent component analyzers mixture model for allowing any distribution to be modeled.

In Posik (2004) a new model of probability density function (the distribution tree) and its use in EDAs is described.

A new algorithm based on the learning and later simulation of a Bayesian classifier in every generation is introduced in Miquélez et al. (2004). In the method they propose, at each iteration the selected group of individuals of the population is divided into different classes depending on their respective fitness values. Afterwards, a Bayesian classifier is learned to model the corresponding supervised classification problem. The simulation of the latter Bayesian classifier provides the individuals that form the next generation.

Lu and Yao (2005) propose two new EDAs for continuous optimization. Both of them incorporate clustering techniques into the estimation process to break the single Gaussian distribution assumption when building the probabilistic models (which would normally mislead the search when dealing

with multimodal functions). These new algorithms also overcome the restriction of demanding prior knowledge needed before applying clustering (which is unreasonable in real life).

Gallager and Frean (2005) deal with continuous EDAs. The authors develop a new algorithm based on a stochastic gradient descent on the Kullback-Leibler divergence between a probability density and a model of the objective function. They associate the designed algorithm with a continuous version of the population-based incremental learning algorithm and the generalized means shift clustering framework.

Peña et al. (2005) suggest that EDAs can be used in globally multimodal problems. These problems characteristically present several global optima. As these kinds of problems are difficult for most common EDAs, the authors propose a new probabilistic graphical model which consists of 'multinets'. This kind of Bayesian network efficiently and elegantly solves the multimodal problem.

New EDAs for Solving Particular Problems

Shan et al. (2003) give a new approach of EDAs for program synthesis, which is named Program Evolution with Explicit Learning (PEEL).

In Yanai and Iba (2003) Estimation of Distribution Programming algorithm (EDP), is proposed. This algorithm is based on a probability distribution expression using a Bayesian network, and it is a search method that uses an EDA-like approach to solve Genetic Programming problems.

When applied to constrained optimization problems, most EDAs (as well EAs) use special techniques for handling invalid solutions. Grahl and Rothlauf (2004) present PolyEDA, a new EDA approach that is able to directly consider linear inequality constraints by using Gibbs sampling.

Pelikan and Lin (2004) describe, implement and test the Parameter-less Hierarchical BOA algorithm, which is a black-box optimization algorithm that can be applied to hierarchical and nearly decomposable problems without setting any parameters whatsoever.

Ahn et al. (2004) present a continuous EDA for solving decomposable, real-valued optimization problems, named real-coded Bayesian optimization algorithm (rBOA).

Bosman and de Jong (2004) propose a probability distribution over trees to be used in an EDA for GP.

Santana (2005) describes a new probability model to be used in EDAs. After analyzing the limitations of the most common probability models used in EDAs, the author proposes a new probability model based on what in statistical physics is known as the Kikuchi approximation. The author studies problems where this kind of model is superior to the ones previously used.

EDAs Combined with Other Techniques

Tsutsui (2002) presents new EDAs in permutation representation domain using edge histogram based sampling algorithms, and tests them in TSP problems.

Handa (2003) proposes a hybrid method of EDAs with a repair method for solving Constraints Satisfaction Problems.

Paul and Iba (2003) propose an algorithm for combinatorial optimization that uses reinforcement learning and EDA estimation of joint probability distributions of promising solutions to generate a new population of solutions named Reinforcement Learning Estimation of Distribution Algorithm (RELEDA).

The Distributed Probabilistic Model Building GA (DPMBGA) is proposed in the work by Hiroyasu et al. (2003). In the DPMBGA, correlation among the design variables is considered by principal component analysis when the offsprings are generated.

A new algorithm which combine the ideas that inspire EDAs and inductive machine learning, called SI3E is presented in Llorà and Goldberg (2003) .

De la Ossa et al. (2004) apply island models to EDAs in the field of combinatorial optimization.

Oceanesek et al. (2004) present a hybrid evolutionary strategy – called Adaptive Mixed Bayesian Optimization Algorithm (MBOA)– combining the Mixed Bayesian Optimization Algorithm (MBOA) with the variance adaptation as implemented in Evolution Strategies.

Mühlenbein and Höns (2005) relate EDAs to algorithms previously developed in statistics, artificial intelligence and statistical physics. The authors view all of these algorithms as minimizers of the Kullback-Leibler divergence between an unknown distribution $p(\mathbf{x})$ and a class of distributions. After that, the authors concentrate on the problems that must be faced in order to learn appropriate distributions for three particular algorithms.

3.2.3.3 Applications

Laummans and Ocenasek (2002), investigate the usefulness of EDAs in multi-objective optimization, integrating the model building and sampling techniques of BOA into an evolutionary multi-objective optimizer using a special selection scheme.

In Bengoetxea et al. (2002) EDAs are proposed as an approach for inexact graph matching. Also the performance of EDAs for inexact graph matching is compared with that of GAs.

UMDA and PBIL are used to learn a Bayesian network structure from a database of cases in a score + search framework in Blanco et al. (2003). A comparison with a GA approach is made using three different scores: penalized maximum likelihood, marginal likelihood, and information-theory-based entropy.

Cantú-Paz (2003) describes the application of four EAs (a simple GA, cGA, EcGA and BOA) to the pruning of neural networks used in classification problems.

Pelikan and Goldberg (2003) apply the Hierarchical BOA (hBOA) to two classes of real-world problems: Ising spin-glass systems and Maximum Satisfiability (MAXSAT).

The Distributed probabilistic Model Building Genetic Algorithm (DPMBGA) with the penalty and pulling back methods is applied in Shimosaka et al. (2003) to truss structural optimization problems.

Tsutsui et al. (2003) explore the effect of introducing a tag node in string representation for solving flow shop scheduling problems with the algorithms proposed in Tsutsui (2002).

A method for optimization of composite video processing systems based on a genetic algorithm with Hill-Climbing, Simplex and EDAs, is proposed in Ali and Topchy (2004).

Sastry and Goldberg (2004) present a selectomutative GA in which EcGA is used to automatically identify key building blocks of the search problem.

Munetomo et al. (2004) present empirical results on parallelization of the linkage identification compared to that of parallel BOA.

Li et al. (2004) apply the Bayesian optimization and classifier systems to nurse scheduling.

A new EDA for the identification of informative genes for molecular classification is given in Paul and Iba (2004).

UMDA and PBIL are used in Romero et al. (2004) as a search engine in the Bayesian network structure learning problem, to obtain the optimal ordering of variables for the K2 algorithm. The authors also check whether the individual representation and its relation to the corresponding ordering play important roles, and whether MIMIC outperforms the results of UMDA.

A method for segmentation and recognition of image structures based on graph homomorphisms is presented in Cesar et al. (2005). The search for the best homomorphism is carried out by optimizing an objective function based on similarities between object and relational attributes defined on the graphs. The authors compare and discuss the following optimization procedures: deterministic tree search, for which new algorithms are detailed, GAs and EDAs.

Mendiburu et al. (2005) propose new parallel versions of some EDAs. In discrete domains, the authors explain the parallelization of EBNA$_{BIC}$ and EBNA$_{PC}$, while in continuous domains, the selected algorithms are EGNA$_{BIC}$ and EGNA$_{ee}$.

Part II

Convergence Results

Chapter 4

General Convergence Results of Discrete EDAs

The standard measures of performance for optimization algorithms involve convergence properties (whether or "under what assumptions" can it be guaranteed that the algorithm reaches a (n optimal) solution) as well as time complexity (how quickly they are found). This chapter discusses results obtained concerning convergence properties of EDAs.

In spite of successful applications of EDAs in combinatorial optimization, little attention has been given to the theoretical aspects of this class of algorithms, and it is still unclear in theory when EDAs work. In fact, as far as we know, there is only one other work (here we include a summary of it) which deals with convergence properties of this class of algorithms in discrete spaces. A study of global convergence of EDAs in continuous spaces can be found in Zhang and Mühlenbein (2004), where the convergence of FDA under proportional selection is also analyzed.

This chapter builds the first general analytic framework given in this dissertation for analyzing convergence of EDAs: the Markov chain model. In this approach an EDA is characterized as a Markov chain and then its convergence is studied obtaining a "general" convergence result (a result valid for the class of all EDAs, not only for a particular instance of them). The most common discrete EDAs are analyzed using this theorem, resulting in convergence and non-convergence algorithms. For those algorithms that do not converge, some conditions have been imposed on estimating their probability distribution parameters to guarantee convergence. This chapter summarizes a study that published earlier in González et al. (2002a).

The chapter is organized as follows: in Section 4.1, we model EDAs with Markov chains, and introduce a new general theorem about the limit behavior of these algorithms. Section 4.2 deals with the other general result found in the literature. The chapter ends with a brief summary.

4.1 EDAs and Markov Chain Models

Due to the stochastic nature of EDAs (Figure 3.2 shows a general pseudocode), random process theory would seem to provide an appropriate set of tools for describing their behavior. In particular Markov chains constitute a proper and natural mathematical tool for this purpose. This section explains how we model EDAs by using Markov chains. First, we give a general theorem about the limit behavior of EDAs and apply it to some of these algorithms.

Without lack of generality we consider that the optimization problem to solve is:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \tag{4.1}$$

where $f : \Omega \to I\!R$ is the objective function and Ω denotes the search space.

Definition 4.1. *Let $\mathbf{x}^* \in \Omega$ be an individual such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \Omega$. This individual is said to be* **a global optimum** *(minimum in our case) of equation (4.1).*

The search space is represented by:

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n$$

where $\Omega_i = \{1, 2, \ldots, r_i\}$ for all $i = 1, 2, \ldots, n$ and $n \in I\!N$ denotes the length or dimension of each individual $\mathbf{x} \in \Omega$. The cardinality of the search space is $|\Omega| = r_1 \cdot r_2 \cdot \ldots \cdot r_n = m$. Suppose that $\Omega = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, $\mathbf{x}_i$ is called the i^{th} individual of Ω, with $i = 1, 2, \ldots, m$. A list of the different individuals $\mathbf{x} \in \Omega$ can be obtained as follows:

$$\begin{aligned} \mathbf{x}_1 &= (x_{1,1}, x_{1,2}, \ldots, x_{1,n}) \\ \mathbf{x}_2 &= (x_{2,1}, x_{2,2}, \ldots, x_{2,n}) \\ &\vdots \\ \mathbf{x}_m &= (x_{m,1}, x_{m,2}, \ldots, x_{m,n}) \end{aligned}$$

A population D in Ω is a subset of size M of individuals of Ω. Note that a population can have repeated elements. The set $\mathcal{D}$ of all the populations of the algorithm can be represented as follows:

$$\mathcal{D} = \{D = (d_1, d_2, \ldots, d_m) \,|\, d_i = \text{number of copies of the } i\text{-th individual in population } D\}$$

Of course, $\sum_{i=1}^{m} d_{it} = M$. The number of different populations, v, is equal to the number of different ways to place $m-1$ balls into $M+m-1$ boxes, i.e.:

$$v = \left(\begin{array}{c} M+m-1 \\ m-1 \end{array} \right).$$

If we denote the population at step t by D_t, it is clear that, in a general EDA the resulting new population, D_t only depends on the state of the previous population, D_{t-1} in a probabilistic manner. This fact, known as the *Markov property*, reveals that Markov processes are appropriate models for modeling the probabilistic behavior of EDAs. Different Markov models can be used. The three most natural are those in which the state space E is given by:

Markov model I: The different populations that the algorithm can take.

Markov model II: All the possible probability distributions, $\rho_t(\mathbf{x})$, that can be formed from a population of selected individuals of size N.

Markov model III: All the possible selected populations (of size N) that the algorithm can take.

In this case we have chosen the first possibility. We model a general EDA using a finite Markov chain whose state space is formed from the different populations that the algorithm can take:

$$E = \{D_1, D_2, \ldots, D_v\} \ .$$

So, at each step t there exists an $l \in \{1, 2, \ldots, v\}$ such that $D_t = D_l$. Moreover, no operation used for the calculation of the transition probabilities depends on the step parameter t, so the Markov chain is homogeneous.

4.1.1 General Theorem for the Convergence of Discrete EDAs

In principle, the question whether an EDA will visit a global optimum and if so, whether it will converge in some mode to the optimum or not, may be answered by modeling the EDAs under consideration like the previous Markov chain, so that the existing powerful results from Markov chain theory can be exploited.

The following theorem is a new general result about the limit behavior of discrete EDAs. We find a sufficient condition for the convergence of these algorithms. Notice that the deterministic concept of *convergence to the optimum* is not appropriate here because the state transitions of an EDA are of a stochastic nature. Since there are a number of possible definitions of convergence, it is important to clarify what we understand by *convergence* in this section. The idea of convergence is formalized using the set of populations $\mathcal{D}^*$ that contain a global optimum:

$$\mathcal{D}^* = \{D \mid \exists\ \mathbf{x} \in D \text{ such that } f(\mathbf{x}) = f(\mathbf{x}^*)\} \ .$$

Definition 4.2. *Let A be a discrete EDA. We say that A* **converges to a population that contains a global optimum***, when :*

$$\exists t_0 \text{ such that } \forall t \geq t_0 \quad D_t \in \mathcal{D}^*.$$

Therefore if A converges to a population that contains a global optimum, this means that once a population of $\mathcal{D}^*$ is reached the chain will never visit another population such that $D \notin \mathcal{D}^*$, i.e. the algorithm never loses this global optimum.

Theorem 4.1. *Let A be a discrete EDA such that:*

$$p_t(\mathbf{x}) \geq \delta > 0, \text{ for all } \mathbf{x} \in \Omega, \text{ and for all step } t = 1, \ldots \tag{4.2}$$

Then A visits populations of $\mathcal{D}^*$ infinitely often with probability one. If additionally the selection is elitist, then A converges to a population that contains a global optimum.

Proof. Suppose that algorithm A is non elitist. In this case we show that the Markov chain has a probability transition matrix $Q = [q_{rs}]_{r,s=1,2,\ldots,v}$ whose entries are all positive.

The probability of going from population D_r to population D_s at step t of the algorithm is given by:

$$q_{rs} = P(D_s|D_r) = \sum_{D_r^{Se}} p_{sel}(D_r^{Se}) \frac{M!}{d_{1s}! d_{2s}! \ldots d_{ms}!} \prod_{i=1}^{m} \underbrace{p_{D_r^{Se}}(\mathbf{x}_i)^{d_{is}}}_{>0} > 0$$

where $p_{sel}(D_r^{Se})$ is the probability of selecting D_r^{Se} from D_r, and $p_{D_r^{Se}}(\mathbf{x})$ is the estimated joint probability distribution from D_r^{Se}. $p_t(\mathbf{x})$ coincides with some $p_{D_r^{Se}}(\mathbf{x})$ at step t of the algorithm.

Hence the Markov chain is irreducible (all the states are intercommunicated), and the chain is aperiodic. Since the chain is finite and irreducible, it is positive persistent. This results in the existence of a limit distribution:

$$\lim_{k\to\infty} q_{rs}^{(k)} = \pi_s$$

where $q_{rs}^{(k)}$ is the probability of going from population D_r to population D_s in k steps and the π_s are positive for all $s = 1, 2 \ldots, v$. Therefore the chain will visit $\mathcal{D}^*$ infinitely often with probability one. In fact it visits all the states infinitely often. This proves the first part of the theorem.

For the second part, if the selection is elitist, when a global optimum is found it will never be lost, and therefore the algorithm converges to a population that contains a global optimum. To describe the transition probability matrix in this case, suppose that all the possible populations are grouped and ordered by the number of optima that contain:

$$\begin{aligned}
\mathcal{D}_M^* &= \{D \in \mathcal{D} | D \text{ contains } M \text{ optima } \} \\
\mathcal{D}_{M-1}^* &= \{D \in \mathcal{D} | D \text{ contains } M-1 \text{ optima } \} \\
&\vdots \\
\mathcal{D}_2^* &= \{D \in \mathcal{D} | D \text{ contains 2 optima } \} \\
\mathcal{D}_1^* &= \{D \in \mathcal{D} | D \text{ contains 1 optima } \} \\
\mathcal{D}_0^* &= \{D \in \mathcal{D} | D \text{ contains 0 optima } \}
\end{aligned}$$

Taking this grouping into account the transition probability matrix of the chain can be written as the following lower triangular matrix:

$$\begin{pmatrix}
\mathbf{R}_{M,M} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\mathbf{R}_{M-1,M} & \mathbf{R}_{M-1,M-1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
\mathbf{R}_{2,M} & \mathbf{R}_{2,M-1} & \cdots & \mathbf{R}_{2,2} & \mathbf{0} & \mathbf{0} \\
\mathbf{R}_{1,M} & \mathbf{R}_{1,M-1} & \cdots & \mathbf{R}_{1,2} & \mathbf{R}_{1,1} & \mathbf{0} \\
\mathbf{R}_{0,M} & \mathbf{R}_{0,M-1} & \cdots & \mathbf{R}_{0,2} & \mathbf{R}_{0,1} & \mathbf{R}_{0,0}
\end{pmatrix}$$

where the entries of each sub-matrix $\mathbf{R}_{i,j}$ represent the probabilities of going from each population that contains i optima to each population that contains j optima, and each $\mathbf{R}_{i,j}$ has dimension $|\mathcal{D}_i^*| \times |\mathcal{D}_j^*|$.

□

4.1.2 Applying the General Theorem of Convergence to Some EDAs

Theorem 4.1 offers an easy way to argue that some instances of EDAs converge to a global optimum. To demonstrate that an algorithm converges to a global optimum it is sufficient to prove that at each step the estimated joint probability distribution from selected individuals assigns positive probability to each point of the search space. Next we analyze the convergence of different examples of EDAs by checking whether condition (4.2) is fulfilled:

4.1.2.1 UMDA

UMDA (Mühlenbein, 1998) uses the simplest model to estimate the joint probability distribution of the selected individuals at each generation, $p_t(\mathbf{x})$. This joint probability distribution is factorized as a product of independent univariate marginal distributions:

$$p_t(\mathbf{x}) = \prod_{i=1}^{n} p_t(x_i) = \prod_{i=1}^{n} p(x_i|D_{t-1}^{Se}).$$

These univariate marginal distributions are estimated from marginal frequencies:

$$p(x_i|D_{t-1}^{Se}) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i|D_{t-1}^{Se})}{N}$$

where

$$\delta_j(X_i = x_i|D_{t-1}^{Se}) = \begin{cases} 1 & \text{if in the } j^{th} \text{ case of } D_{t-1}^{Se},\ X_i = x_i \\ 0 & \text{otherwise.} \end{cases}$$

Hence, taking into account the way in which the probabilities are estimated, there could be some situations where an $\mathbf{x}$ exists such that $p_t(\mathbf{x}) = 0$. For example, when the selected individuals at a previous step are such that $\delta_j(X_i = x_i|D_{t-1}^{Se}) = 0$ for all $j = 1, \ldots, N$, given an individual $\mathbf{x}$ with x_i in the i^{th} component, it turns out that $p(x_i|D_{t-1}^{Se}) = 0$ and therefore:

$$p_t(\mathbf{x}) = p(x_i|D_{t-1}^{Se}) \prod_{\substack{k=1 \\ k \neq i}}^{n} p(x_k|D_{t-1}^{Se}) = 0.$$

Therefore condition (4.2) is not fulfilled and we can not ensure that UMDA visits a global optimum.

In fact, the Markov chain that models the UMDA has m absorbing states. Those absorbing states correspond to uniform populations. A uniform population is formed by M copies of the same individual, and can be represented by:

$$D_r = (0, \ldots, 0, M, 0, \ldots, 0).$$

In this case, the probability of visiting a population D_s from a uniform population D_r is:

$$P(D_s|D_r) = \begin{cases} 0 \text{ if } D_s \neq D_r \\ 1 \text{ otherwise.} \end{cases}$$

Therefore if the chain visits one of these populations it will be trapped in it.

Clearly UMDA's non-convergence is due to the way in which the probabilities $p(x_i|D_{t-1}^{Se})$ are estimated. To overcome this problem the Laplace correction (Cestnik, 1990) could be applied. In this case if the parameters $p(x_i|D_{t-1}^{Se})$ are estimated as

$$\frac{\sum_{j=1}^{N} \delta_j(X_i = x_i|D_{t-1}^{Se}) + \varepsilon}{N + \varepsilon \cdot r_i}, \quad 0 < \varepsilon \leq 1,$$

we ensure that condition (4.2) is fulfilled.

4.1.2.2 MIMIC

Unlike UMDA, MIMIC (de Bonet et al., 1997) takes into account pairwise dependencies among variables in order to estimate the joint probability distribution.

At each step of this algorithm, a permutation of the indexes $i_1, i_2, \ldots, i_n$ that fulfills an entropy related condition must be found before the probabilities can be estimated. Then the joint probability distribution is factorized as:

$$p_t(\mathbf{x}) = p_t(x_{i_1}|x_{i_2}) \cdot p_t(x_{i_2}|x_{i_3}) \cdot \ldots \cdot p_t(x_{i_{n-1}}|x_{i_n}) \cdot p_t(x_{i_n})$$

where each conditional probability is estimated from the database D_{t-1}^{Se}, by using conditional relative frequencies. Hence if we use the same argument that we did for UMDA, we can not state that MIMIC visits a global optimum. But in order to ensure that condition (4.2) is fulfilled by MIMIC, it is sufficient to make similar changes in the estimation of the conditional probabilities that we have shown for UMDA.

4.1.2.3 EBNA Algorithms

EBNA (Etxeberria and Larrañaga, 1999; Larrañaga et al., 2000a; Pelikan and Goldberg, 2000a; Pelikan and Goldberg, 2000b; Pelikan et al., 1999; Pelikan et al., 2000c; Mühlenbein and Mahnig, 1999) are a set of algorithms which allows statistics of unrestricted order in the factorization of the joint probability distribution. This joint probability distribution is encoded by a Bayesian network that is learnt from the database containing the selected individuals at each generation. The factorization can be written as:

$$p_t(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|\mathbf{pa}_i^t)$$

where $\mathbf{pa}_i^t$ is an instantiation of $\mathbf{Pa}_i^t$, the set of parents of variable X_i.

In EBNA learning the probabilistic model at each generation of the algorithm means learning a Bayesian network from the selected individuals. Different algorithms can be obtained by varying the structural search method. Two structural search methods are usually considered: score+search and detecting conditional (in)dependencies (EBNA$_{PC}$). Particularly, two scores are used in the score+search approach, the BIC score (EBNA$_{BIC}$) and the K2+penalization score (EBNA$_{K2+pen}$).

For all above described EBNAs the convergence is only affected by the calculus of the parameters θ_{ijk}, where θ_{ijk}, as explained in Section 3.2.1.3, represents the conditional probability of variable X_i being in its k^{th} value, given that the set of its parent variables is in their j^{th} value. The parameters of the local probability distributions can be calculated for every generation using either:

- Their expected values as obtained in Cooper and Herskovits (1992):

$$\mathsf{E}[\theta_{ijk}|D_{t-1}^{Se}] = \frac{N_{ijk}+1}{N_{ij}+r_i} \tag{4.3}$$

  or

- The maximum-likelihood estimates:

$$\widehat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}} \tag{4.4}$$

where N_{ijk} denotes the number of cases in D_{t-1}^{Se} in which the variable X_i takes its k^{th} value and its parents $\mathbf{Pa}_i$, are instantiated as their j^{th} value. N_{ij} represents the number of cases in which the parents of variable X_i take their j^{th} value.

In the first case, we can conclude that when the selection is elitist, EBNAs converge to a population that contains a global optimum because (4.3) is always a positive value. In the second case, as with UMDA and MIMIC, we can not ensure that EBNAs reach a global optimum because the quantity (4.4) could be zero.

4.1.2.4 BOA

BOA (Pelikan et al., 1999; Pelikan and Goldberg, 2000a; Pelikan and Goldberg, 2000b; Pelikan et al., 2000c) uses Bayesian networks to encode the joint probability distribution. The structural search is driven by the BDe score (Heckerman et al., 1995). In this case the parameters of the local distributions are calculated following a Bayesian approach that avoids taking a zero value. Hence we can say that, when the selection is elitist, the algorithm converges to a population that contains a global optimum.

4.1.2.5 LFDA

LFDA (Mühlenbein and Mahnig, 1999), like EBNAs and BOA, encodes the joint probability distributions with Bayesian networks. LFDA uses the same score as EBNA$_{BIC}$ but limits the number of parents that a variable can take. In the case that the parameters of the local distributions are calculated using the maximum-likelihood estimates (4.4) the convergence of the algorithm can not be ensured. As in the previous algorithms the use of the Laplace correction will provide convergence.

4.2 Convergence for BEDA with Infinite Populations

In this section we summarize the only work in discrete spaces (Mühlenbein et al., 1999) – apart from ours – where a convergence analysis of a general discrete EDA is carried out. The authors introduce an EDA which uses Boltzmann selection: Boltzmann Estimation of Distribution Algorithm (BEDA). BEDA is similar to a simulated annealing algorithm. But it generates a population of points instead of a single point at each step, using the exact Boltzmann distribution (simulated annealing only approximates the Boltzmann distribution). In their work the authors show the convergence of a general BEDA for infinite populations.

The proof is based on an interesting property of the Boltzman selection that establishes that when the points have been generated according to a Boltzmann distribution ($u > 0$):

$$p_0(\mathbf{x}) = \frac{u^{f(\mathbf{x})}}{\sum_{\mathbf{y}\in\Omega} u^{f(\mathbf{y})}},$$

and Boltzman selection is used with basis $v > 1$:

$$p_{t,sel}(\mathbf{x}) = p_t(\mathbf{x}) \frac{v^{f(\mathbf{x})}}{\sum_{\mathbf{y}\in\Omega} p_t(\mathbf{y}) v^{f(\mathbf{y})}}$$

then after selection the selected points are also distributed according to a Boltzmann distribution, with parameter $u \cdot v > u$

$$p_1(\mathbf{x}) = \frac{(u \cdot v)^{f(\mathbf{x})}}{\sum_{\mathbf{y} \in \Omega} (u \cdot v)^{f(\mathbf{y})}} \quad .$$

This fact allows us to write the probability distribution at step t for a BEDA as:

$$p_t(\mathbf{x}) = \frac{(u \cdot v^t)^{f(\mathbf{x})}}{\sum_{\mathbf{y} \in \Omega} (u \cdot v^t)^{f(\mathbf{y})}} \tag{4.5}$$

Using the previous arguments the authors prove the following general theorem of convergence for a BEDA algorithm:

Theorem 4.2. *Let $f(\mathbf{x}^*) = \min_{\mathbf{x} \in \Omega} f(\mathbf{x})$. The minimum need not be unique. Let the distribution $p_t(\mathbf{x})$ be given by equation (4.5). Let $v > 1$. Then*

$$f(\mathbf{x}) > f(\mathbf{x}^*) \Longrightarrow \lim_{t \to \infty} p_t(\mathbf{x}) = 0.$$

If the minimum is unique, then

$$\lim_{t \to \infty} \mathsf{E}_{p_t}[f(\mathbf{x})] = f(\mathbf{x}^*).$$

Intuitively, Theorem (4.5) proves that almost all the individual points in the population will move to globally optimal points (in this case to the minima) as time tends to infinity and taking into account infinite populations.

4.3 Summary

In this chapter we have modeled EDAs using Markov chains. This framework allows us to prove a general theorem of convergence for these algorithms. The result consists of giving a sufficient condition which guarantees that the algorithm visits populations that contain a global optimum infinitely often with probability one. If additionally the selection is elitist, then the EDA converges to a population that contains a global optimum.

We also include the other general result on convergence of discrete EDAs existing in the literature. The result proves that EDA – with Boltzmann selection and infinite populations – converges to a population in which almost all individuals are copies of globally optimal solutions, as time tends to infinity.

Chapter 5

Analysis of the PBIL algorithm

In order to offer a mathematical analysis of the convergence properties of the PBIL algorithm, in this chapter the analysis is carried out under two analytic frameworks: Markov chains and discrete dynamical systems.

PBIL (see Section 3.2.1.1) is a simple example of EDAs, and uses a probability model which is a product of independent probabilities for each component in the string. This instance of EDAs does not exactly comply with the general model given in Figure 3.2, because at each step PBIL does not maintain a population of individuals. At each step PBIL maintains a vector of probabilities that represents the population of individuals. Therefore the general Markov chain model built in the previous chapter can not be applied to PBIL, and consequently Theorem 4.1 can not be used to analyze the convergence behavior of this algorithm.

However, like the other EDAs, PBIL can be modeled using a Markov model, but the Markov chain used in the analysis is different from the one used in the previous chapter. Such an approach exactly models the behavior of PBIL, but the analysis of the chain is very complex, and our study has been reduced to a simple problem and a simple case of the algorithm. In spite of this, the Markov chain model allows us to prove a strong dependence of PBIL convergence on the initial vector $p_0(\mathbf{x})$ and on the α parameter value.

The discrete dynamical systems framework enables us to perform a more general analysis than Markov chains do. A general version of the algorithm is modeled, in the case of optimization of a general injective objective function. The conclusions obtained can be easily extended to noninjective functions. Associating an appropriate dynamical system to PBIL, we show that this algorithm follows the iterates of the dynamical system when α is near zero. Stability analysis of the dynamical system carried out shows that PBIL can only converge to local optima (when α is near zero), meaning, in the case of unimodal functions, that PBIL converges to the global optimum.

The chapter is structured in four sections. Section 5.1 presents an analysis published earlier in González et al. (1999) and was extended in González et al. (2001), where Markov chains are used to model the simplest version of PBIL. Section 5.2 deals with the discrete dynamical system proposed for the modeling of PBIL, and reports the study that appears in González et al. (2000). Section 5.3 reviews other works concerning the theoretical analysis of PBIL. Finally we draw conclusions in Section 5.4.

5.1 Modeling PBIL by Means of Markov Chains

In this section a preliminary study is made of the convergence behavior of PBIL based on Markov chains. However, it is not possible to use the same Markov chain utilized in the previous chapter, arguing through the theorem proven there. This is because PBIL is based on the idea of substituting the individuals of a population by a set of their statistics. In each generation, the population of individuals is represented by a vector of probabilities $\mathbf{p}_t$. That is, while in most EDAs a population of individuals is maintained, in PBIL all this information is encapsulated in a vector of probabilities, thus requiring a different sort of argument.

As a preliminary analysis, we perform a number of experiments in order to make out the behavior of the algorithm. These empirical results are used only as an incidental tool, while in the next section the results suggested by them are mathematically proved. The objective of the experiments is to answer (empirically, of course) the following questions: given initial values of the probability vector, $\mathbf{p}_0$ and of α, does the algorithm converge to any point of the search space? And if so, does the algorithm converge to the global optimum, or does the limit point depend on the initial values of $\mathbf{p}_0$ and of α? If the latter, how does it depend on those initial values?

We carry out the analysis on the simplest version of PBIL and when it is used to minimize the $OneMax$ function in two dimensions. In other words, the optimization problem to solve is:

$$\min_{\mathbf{x}\in\Omega} OneMax(\mathbf{x}) \tag{5.1}$$

where the search space is $\Omega = \{0,1\}^2$, and $OneMax(\mathbf{x}) = \sum_{i=1}^{2} x_i$. The optimum of $OneMax$ is, clearly, at point $(0,0)$ and the worst value is at $(1,1)$.

The version of PBIL used here to solve problem (5.1) is the simplest one; i.e. we consider that at each step two individuals ($\lambda = 2$) are generated and the best ($\mu = 1$) is selected to modify the probability vector $\mathbf{p}_t$.

In the experiments different values of $\mathbf{p}_0$ and α are used. In particular we take four values for $\mathbf{p}_0$: $(.05,.05)$, $(.50,.50)$, $(.95,.05)$ and $(.95,.95)$, and three for α: .05, .50 and .95.

For each combination of these parameters we carried out 15×10^4 executions. Table 5.1 shows the results. Each entry of the table represents the number of executions in which PBIL with initial probability $\mathbf{p}_0$ and parameter α got a particular point of the search space. For instance, with $\mathbf{p}_0 = (.95,.95)$ and $\alpha = .5$ the point $(1,1)$ was reached 104,056 times.

In view of the experimental results, we can deduce the following. PBIL does not converge to the global optimum. In every case the algorithm converges, but the limit point depends on the values of $\mathbf{p}_0$ and α. It seems that when the value of α is near zero, the algorithm converges to the optimum with high probability, but in other situations this does not happen. For instance, when $\mathbf{p}_0$ takes values near $(1,1)$ and α takes values near 1, the algorithm goes to $(1,1)$ with a very high probability. Similarly, when the algorithm starts with $\mathbf{p}_0$ near $(1,0)$ and α near 1 we obtain similar results for the point $(1,0)$. The next section proves mathematically some of the results suggested by these experiments.

5.1.1 A Markov Chain that Models PBIL

As we have seen empirically the behavior of PBIL depends on the initial vector $\mathbf{p}_0$ and on the value of the parameter α. Thus, the experimental results obtained appear to show mathematically that the sequence $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ (and hence the algorithm) can go to any point of the search space for suitable values of $\mathbf{p}_0$ and α.

$\mathbf{p}_0$	α	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$(.05,.05)$	.05	150,000	0	0	0
	.50	149,760	132	108	0
	.95	148,818	547	635	0
$(.5,.5)$	.05	150,000	0	0	0
	.50	107,220	20,335	20,405	2,040
	.95	69,213	37,195	35,376	8,216
$(.95,.05)$	.05	120,034	0	29,966	0
	.50	24,824	74	125,033	69
	.95	14,865	413	134,465	257
$(.95,.95)$	.05	88,178	28,055	27,948	5,819
	.50	2,942	21,501	21,501	104,056
	.95	897	14,445	13,922	120,736

Table 5.1. Results of the experiments performed to see whether the behavior of PBIL depends on $\mathbf{p}_0$ and α.

As above mentioned, the Markov chain used to model PBIL can not be the same as the one used for the EDAs based on populations. But this does not mean that Markov chains are not an appropriate tool for modeling PBIL. Because the probability vector $\mathbf{p}_t$ only depends on $\mathbf{p}_{t-1}$, we can model PBIL by using a Markov chain whose state space E is given by all the possible probability vectors, $\mathbf{p}_t$, that can be obtained from a population of selected individuals of size μ:

$$E = \left\{ (1-\alpha)^t \mathbf{p}_0 + \alpha \left(\sum_{k=0}^{t-1} (1-\alpha)^k \mathbf{x}_{1:2}^{(t-k-1)} \right) \Big| \quad t = 1, 2, \ldots, \text{ and } \mathbf{x}_{1:2}^{(l)} \in \{0,1\}^2, \quad l = 0, \ldots, t-1 \right\}.$$

Note that the state space of this chain is infinite numerable and that for each $\mathbf{p}_0$ and α we have a different Markov chain.

Given an initial probability vector $\mathbf{p}_0$ and a value of parameter α, the above described Markov chain exactly models the behavior of PBIL. Unfortunately, analysis of this chain is very complex because every state is transient. However it is easier to study the probabilities of certain transitions in the chain, making it possible to calculate the probability of the chain path driving the algorithm towards convergence to a particular point.

In particular, for each point of the search space we find a lower bound for the probability of a chain path driving the algorithm towards convergence to this point.

For example, using the point (1,1), we see how to calculate a lower bound of the probability of the chain path driving the algorithm convergence to this point. The arguments used for the point $(1,1)$ illustrate the procedure used in the other points. In this case the chain path is composed of the probability vectors that would be generated if at each step of the algorithm the point $(1,1)$ were obtained after the selection. It is important to realize that in this case the sequence $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ converges to the point $(1,1)$.

Hence we want to calculate the probability that the Markov chain visits the states:

$$\mathbf{p}_0,\ \mathbf{p}_1, \ldots,\ \mathbf{p}_t, \cdots$$

where $\mathbf{p}_t = (1-\alpha)\mathbf{p}_{t-1} + \alpha \mathbf{x}_{1:2}^{(t-1)}$ and $\mathbf{x}_{1:2}^{(t-1)} = (1,1)$ for all t. It is similar to say that at each iteration of PBIL the point $(1,1)$ is obtained.

As seen in Höhfeld and Rudolph (1997), the probability of obtaining the point $(1,1)$ at the $(t+1)^{th}$ step of PBIL, starting from a probability vector $\mathbf{p}_t = (p_1^{(t)}, p_2^{(t)})$, is given by:

$$(p_1^{(t)} \cdot p_2^{(t)})^2. \tag{5.2}$$

In Höhfeld and Rudolph (1997) the probabilities of obtaining the remainder points of Ω are also calculated.

Therefore the probability of the chain path is expressed by an infinite product of type (5.2) probabilities. By calculating the probabilities of paths we can establish the following theorem:

Theorem 5.1. *The sequence $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ generated by PBIL in the minimization of OneMax function converges to $(a,b) \in \Omega = \{0,1\}^2$, with probability as near to 1 as we want when $\mathbf{p}_0$ and α go to (a,b) and 1 respectively.*

Proof. We demonstrate the case $(a,b) = (1,1)$, but all others are similar.
We want to calculate the probability that the chain visits the states:

$$\mathbf{p}_0,\ \mathbf{p}_1, \ldots,\ \mathbf{p}_t, \ldots$$

where:

$$\begin{aligned} \mathbf{p}_t &= (1-\alpha)\mathbf{p}_{t-1} + \alpha \mathbf{x}_{1:2} \\ &= (1-\alpha)\mathbf{p}_{t-1} + \alpha(1,1) = (1-\alpha)\mathbf{p}_{t-1} + (\alpha,\alpha)\ . \end{aligned}$$

Here,using (5.2), we calculate the probabilities of visiting $\mathbf{p}_t = (1-\alpha)\mathbf{p}_{t-1} + (\alpha,\alpha)$ given $\mathbf{p}_{t-1}$. In other words, we calculate the probability of obtaining in one step the point $(1,1)$:

$$\begin{aligned} &\mathsf{P}\left(\mathbf{p}_t = (1-\alpha)\mathbf{p}_{t-1} + (\alpha,\alpha) \ \mid\ \mathbf{p}_{t-1}\right) \\ &\quad = \ \mathsf{P}(\text{to obtain point } (1,1) \text{ starting from } \mathbf{p}_{t-1}) = (p_1^{(t-1)} p_2^{(t-1)})^2\ . \end{aligned}$$

Hence, the probability we want to calculate, the probability of the chain path, can be expressed as the (infinite) product of the probabilities of each transition. To simplify notation we suppose that $p_1^{(0)} = p_2^{(0)} = p$. After several basic algebraic operations the probability can be written as:

$$\begin{aligned} &\mathsf{P}\left(\lim_{t\to\infty} \mathbf{p}_t = (1,1) \mid p_0 = (p,p)\right) \\ &\quad = \ \mathsf{P}\left(\text{to obtain vector } (1,1) \mid \mathbf{p}_0 = (p,p)\right) = \left[\prod_{t=0}^{\infty} \left[1 + (1-\alpha)^t (p-1)\right]\right]^4 . \end{aligned} \tag{5.3}$$

We study the convergence of this infinite product, knowing that:

$$\prod_{n=0}^{\infty} x_n, \quad (x_n > 0) \ \text{ converges} \Leftrightarrow \sum_{n=0}^{\infty} \ln x_n \text{ converges.}$$

Therefore it is enough to see the behavior of the series

$$\sum_{t=0}^{\infty} \ln\left(1 + (1-\alpha)^t (p-1)\right). \tag{5.4}$$

Using a classical criterion, we show that this series converges when α goes to 1. Instead of calculating the value of the product (5.3) we will give a lower bound for this product, bounding the series (5.4):

$$\sum_{t=0}^{\infty} \ln\left(1+(1-\alpha)^t(p-1)\right) = \ln p + \sum_{t=1}^{\infty} \ln\left(1+(1-\alpha)^t(p-1)\right) > \ln p + \sum_{t=1}^{\infty} \ln\left(1-(1-\alpha)^t\right).$$

Since the power series expression of $\ln(1+x)$ when x is near zero is:

$$\ln(1-x) = -\sum_{k=1}^{\infty} \frac{x^k}{k},$$

we obtain:

$$\begin{aligned}
&\sum_{t=0}^{\infty} \ln\left(1+(1-\alpha)^t(p-1)\right) \\
&> \ln p - \left[\sum_{t=1}^{\infty}\left(\sum_{k=1}^{\infty}\frac{(1-\alpha)^{tk}}{k}\right)\right] = \ln p - \sum_{k=1}^{\infty}\left(\frac{1}{k}\,\frac{(1-\alpha)^k}{1-(1-\alpha)^k}\right) \geq \ln p - \sum_{k=1}^{\infty}\left(\frac{(1-\alpha)^k}{1-(1-\alpha)^k}\right) \\
&\geq \ln p - \sum_{k=1}^{\infty}\left(\frac{(1-\alpha)}{\alpha}\right)^k = \ln p - \left(\frac{1-\alpha}{2\alpha-1}\right).
\end{aligned}$$

Taking the limit of the last expression, when $\alpha \to 1$ and $p \to 1$ we find that

$$\lim_{(\alpha,p)\to(1,1)} \sum_{t=0}^{\infty} \ln\left(1+(1-\alpha)^t(p-1)\right) \geq \lim_{(\alpha,p)\to(1,1)} \left[\ln p - \left(\frac{1-\alpha}{2\alpha-1}\right)\right] = 0 \ .$$

If we take:

$$\ln\left[\prod_{t=0}^{\infty}\left(1+(1-\alpha)^t(p-1)\right)\right] = \ln x$$

we obtain:

$$\ln x \overset{(\alpha,p)\to(1,1)}{\longrightarrow} 0 \Longrightarrow x = \prod_{t=0}^{\infty}\left[1+(1-\alpha)^t(p-1)\right] \longrightarrow 1 \ .$$

□

Theorem 5.1 shows the dependence of the convergence behavior of PBIL on the initial parameters ($\mathbf{p}_0$ and α). If some of them are sufficiently close to 0 or 1, the global optimum may never be found. The result can be generalized to the case in which $\Omega = \{0,1\}^n$ by extending this proof to the general case of a search space of dimension 2^n.

5.2 Modeling PBIL by Means of Discrete Dynamical systems

A new approach can be obtained if we model PBIL by means of discrete dynamical systems. This idea has been used previously for the simple GA (Vose, 1999a; Vose, 1999b) obtaining important results. Our approach and notation are strongly inspired by these previous works. The key question is to associate PBIL with a discrete dynamical system, such that the trajectories followed by the probability vectors $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ in PBIL are related to the iterations of the discrete dynamical system. This fact allows us to study the discrete dynamical system instead of the iterations of PBIL.

5.2.1 Assigning a Discrete Dynamical System to PBIL

For this analysis we consider that the optimization problem to solve is:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$$

where $f : \Omega \to I\!R$ is the objective function and $\Omega = \{0,1\}^n$ denotes the search space. The cardinality of the search space is $|\Omega| = 2^n = m$. The version of PBIL analyzed in this section can be seen in Figure 3.4.

Before presenting the discrete dynamical system associated with PBIL it is useful to note that the behavior of the algorithm is the same in two different functions f_1 and f_2 if:

$$\forall\ \mathbf{x}, \mathbf{x}' \in \Omega\ ,\ f_1(\mathbf{x}) > f_1(\mathbf{x}') \Leftrightarrow f_2(\mathbf{x}) > f_2(\mathbf{x}').$$

Moreover the important thing is not the particular value that a function f has in an individual $\mathbf{x}$ but the ranking implied by this function in Ω. With this argument in mind, we can consider a function $f : \Omega \longrightarrow I\!R$ as a permutation of the elements of Ω. In this case the last individual of Ω, $\mathbf{x}_m$ has the smallest function value, the penultimate $\mathbf{x}_{m-1}$ the second smallest and so on, with the first individual $\mathbf{x}_1$ being the one with the highest function value:

$$f(\mathbf{x}_1) \geq \cdots \geq f(\mathbf{x}_{m-1}) \geq f(\mathbf{x}_m).$$

Therefore the number of different injective functions in Ω is given by $m!$. In order to simplify notation, we assume in the rest of this section that we have to optimize an injective function. However, the results can easily be extended to noninjective functions.

PBIL can be considered as a sequence of probability vectors, each one given by a transition stochastic rule τ:

$$\mathbf{p}_0 \xrightarrow{\tau} \mathbf{p}_1 \xrightarrow{\tau} \mathbf{p}_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} \mathbf{p}_t \xrightarrow{\tau} \mathbf{p}_{t+1} \xrightarrow{\tau} \cdots$$

that is, $\mathbf{p}_{t+1} = \tau(\mathbf{p}_t) = \tau^{t+1}(\mathbf{p}_0)$. We are interested in the trajectories followed by the iterations of τ, and in particular, in its limit behavior:

$$\lim_{t\to\infty} \tau^t(\mathbf{p}_0).$$

We define a new operator $\mathcal{G}$:

$$\mathcal{G} : [0,1]^n \longrightarrow [0,1]^n$$

such that $\mathcal{G}(\mathbf{p}) = (\mathcal{G}_1(\mathbf{p}), \ldots, \mathcal{G}_n(\mathbf{p})) = \mathsf{E}[\tau(\mathbf{p})]$. The operator $\mathcal{G}$ is a deterministic function that gives the expected value of the random operator τ. The iterations of $\mathcal{G}$ are defined as $\mathcal{G}^t(\mathbf{p}) = \mathcal{G}(\mathcal{G}^{t-1}(\mathbf{p}))$. We are interested in the relation between the iterations of τ and the iterations of $\mathcal{G}$, and in particular we want to answer the question: Is there any relation between $\tau^t(\mathbf{p})$ and $\mathcal{G}^t(\mathbf{p})$? However, before looking for this relation let us calculate the expression of $\mathcal{G}(\mathbf{p})$.

The operator $\mathcal{G}$ can be expressed as follows:

$$\mathcal{G}(\mathbf{p}) = \mathsf{E}\left[(1-\alpha)\mathbf{p} + \alpha\frac{1}{\mu}\sum_{k=1}^{\mu}\mathbf{X}_{k:\lambda}\right] = (1-\alpha)\mathbf{p} + \alpha\frac{1}{\mu}\mathsf{E}\left[\sum_{k=1}^{\mu}\mathbf{X}_{k:\lambda} \mid \mathbf{p}\right].$$

Hence, we have to calculate the expected sum of the μ best individuals given that λ have been sampled from the probability vector $\mathbf{p}$. Our analysis is restricted to the case $\mu = 1$ (the most

frequent in the literature). In this case the $\mathcal{G}$ operator can be calculated explicitly, which we do in several steps.

First, the expected value $\mathsf{E}[\sum_{k=1}^{\mu} \mathbf{X}_{k:\lambda} \mid \mathbf{p}]$ is reduced for the case $\mu = 1$ to:

$$\mathsf{E}[\mathbf{X}_{1:\lambda}|\mathbf{p}] = \sum_{i=1}^{n} \mathbf{x}_i \mathsf{P}(\mathbf{X}_{1:\lambda} = \mathbf{x}_i \mid \mathbf{p})$$

where $\mathsf{P}(\mathbf{X}_{1:\lambda} = \mathbf{x}_i \mid \mathbf{p})$ denotes the probability of obtaining $\mathbf{x}_i$ as the best individual after an iteration of the algorithm. Let $\mathsf{P}(\mathbf{S} = \mathbf{x}_i \mid \mathbf{p})$ denote the probability of sampling vector $\mathbf{x}_i$. The probability that $\mathbf{x}_i$ is the best individual, given that we have sampled λ individuals and the probability vector is $\mathbf{p}$, can be expressed as follows (Höhfeld and Rudolph, 1997):

$$\mathsf{P}(\mathbf{X}_{1:\lambda} = \mathbf{x}_i \mid \mathbf{p}) = \mathsf{P}(\mathbf{S} = \mathbf{x}_i \mid \mathbf{p}) \sum_{k=1}^{\lambda} \mathsf{P}(\Omega_i^{>} \mid \mathbf{p})^{k-1} \mathsf{P}(\Omega_i^{\geq} \mid \mathbf{p})^{\lambda-k}$$

where:

$$\begin{aligned} \Omega_i^{>} &= \{\mathbf{x}_j \in \Omega \mid f(\mathbf{x}_j) > f(\mathbf{x}_i)\} \\ \Omega_i^{\geq} &= \{\mathbf{x}_j \in \Omega \mid f(\mathbf{x}_j) \geq f(\mathbf{x}_i)\}. \end{aligned}$$

Finally, it is important to note that, given a probability vector $\mathbf{p}$, the probability of sampling a particular individual $\mathbf{x} = (x_1, \ldots, x_l, \ldots, x_n)$ is:

$$q_{\mathbf{x}}(\mathbf{p}) = \mathsf{P}(\mathbf{S} = \mathbf{x} \mid \mathbf{p}) = \prod_{l=1}^{n} p_l^{x_l} (1 - p_l)^{1-x_l}. \tag{5.5}$$

According to these results it can be said that the operator $\mathcal{G}$ can be expressed as a polynomial function of the probability vector $\mathbf{p}$. In fact, if we take into account that each function can be considered as a permutation of Ω (only the individuals preceding $\mathbf{x}_i$ have a higher function value than $\mathbf{x}_i$), then:

$$\begin{aligned} \mathsf{P}(\Omega_i^{>} \mid \mathbf{p}) &= \sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{p}) \\ \mathsf{P}(\Omega_i^{\geq} \mid \mathbf{p}) &= \sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{p}). \end{aligned}$$

Finally $\mathcal{G}(\mathbf{p})$ can be expressed in our study case ($\mu = 1$) as follows:

$$\mathcal{G}(\mathbf{p}) = (1 - \alpha)\mathbf{p} + \alpha \sum_{i=1}^{m} \mathbf{x}_i q_{\mathbf{x}_i}(\mathbf{p}) \left(\sum_{k=1}^{\lambda} \left(\sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{p}) \right)^{k-1} \left(\sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{p}) \right)^{\lambda-k} \right).$$

5.2.2 Relationship Between $\boldsymbol{\tau}^t(\mathbf{p})$ and $\boldsymbol{\mathcal{G}}^t(\mathbf{p})$

In this section we demonstrate that when the parameter α is near 0, then the stochastic operator τ follows the deterministic operator $\mathcal{G}$ for a long time. First we set up the relation between $\mathcal{G}$ and τ and then we study the relation between their iterates.

Lemma 5.1. *Given $\epsilon > 0$ and $\gamma < 1$, there exists $\alpha_0 > 0$ independent of the probability vector $\mathbf{p}$ such that with probability at least γ:*

$$\alpha < \alpha_0 \Rightarrow ||\tau(\mathbf{p}) - \mathcal{G}(\mathbf{p})|| < \epsilon.$$

Proof. Suppose that $\tau(\mathbf{p}) = (1-\alpha)\mathbf{p} + \alpha\mathbf{x}$. The discrepancy between $\mathcal{G}$ and τ can be bounded by:

$$\begin{aligned} ||\tau(\mathbf{p}) - \mathcal{G}(\mathbf{p})|| &= ||(1-\alpha)\mathbf{p} + \alpha\mathbf{x} - (1-\alpha)\mathbf{p} - \alpha\mathsf{E}[\mathbf{X}_{1:\lambda} \mid \mathbf{p}]|| \\ &= \alpha||\mathbf{x} - \mathsf{E}[\mathbf{X}_{1:\lambda} \mid \mathbf{p}]|| \leq \alpha n. \end{aligned}$$

Since the righthand side goes to zero as $\alpha \longrightarrow 0$ the proof is completed. □

Theorem 5.2. *Given $k > 0$, $\epsilon > 0$, and $\gamma < 1$, there exists α_0 such that with probability at least γ and for all $0 \leq t \leq k$:*

$$\alpha < \alpha_0 \Rightarrow ||\tau^t(\mathbf{p}) - \mathcal{G}^t(\mathbf{p})|| < \epsilon.$$

Proof. We make the proof by induction on k. The base case $k = 1$ coincides with the Lemma 1. Given that $\mathcal{G}$ is uniformly continuous (it is continuous in the compact $[0,1]^n$), choose δ such that

$$||\tau^{k-1}(\mathbf{p}) - \mathcal{G}^{k-1}(\mathbf{p})|| < \delta \Rightarrow ||\mathcal{G}(\tau^{k-1}(\mathbf{p})) - \mathcal{G}(\mathcal{G}^{k-1}(\mathbf{p}))|| < \frac{\epsilon}{2}.$$

By the inductive hypothesis, if $\alpha < \alpha_1$ then with probability at least $1 - (1-\gamma)/2$ we have

$$||\tau^{k-1}(\mathbf{p}) - \mathcal{G}^{k-1}(\mathbf{p})|| < \delta.$$

By the Lemma 1 (applied to $\tau^{k-1}(\mathbf{p})$ instead of $\mathbf{p}$) let α_2 be such that with probability at least $1 - (1-\gamma)/2$

$$\alpha < \alpha_2 \Rightarrow ||\tau^k(\mathbf{p}) - \mathcal{G}(\tau^{k-1}(\mathbf{p}))|| < \frac{\epsilon}{2}.$$

It follows that if $\alpha < \alpha_0 = \min\{\alpha_1, \alpha_2\}$, then with probability at least γ

$$||\tau^k(\mathbf{p}) - \mathcal{G}^k(\mathbf{p})|| \leq ||\mathcal{G}(\tau^{k-1}(\mathbf{p})) - \mathcal{G}^k(\mathbf{p})|| + ||\tau^k(\mathbf{p}) - \mathcal{G}(\tau^{k-1}(\mathbf{p}))|| < \frac{\epsilon}{2} + \frac{\epsilon}{2}.$$

□

Theorem 1 means that when α is near 0, the stochastic operator τ follows, with high probability and for a long time, the iterations of the deterministic operator $\mathcal{G}$.

The operator $\mathcal{G}$ can be thought of as a discrete dynamical system:

$$\mathbf{p}, \mathcal{G}(\mathbf{p}), \ldots, \mathcal{G}^t(\mathbf{p}), \ldots,$$

5.2.3 The Discrete Dynamical System $\mathcal{G}$

In this section we try to find properties of the discrete dynamical system $\mathcal{G}$ that will provide information concerning the behavior of the PBIL algorithm. Before studying the discrete dynamical system $\mathcal{G}$, it is important to note that the following discrete dynamical system $\overline{\mathcal{G}}$:

$$\overline{\mathcal{G}}(\mathbf{p}) = \sum_{i=1}^{m} \mathbf{x}_i q_{\mathbf{x}_i}(\mathbf{p}) \left(\sum_{k=1}^{\lambda} \left(\sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{p}) \right)^{k-1} \left(\sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{p}) \right)^{\lambda-k} \right) \tag{5.6}$$

has the same behavior as $\mathcal{G}$ in the points of the search space. All the results obtained here for $\overline{\mathcal{G}}$ are valid for $\mathcal{G}$. In particular they have the same singular points, so we study the dynamical system of equation (5.6) instead of the operator $\mathcal{G}$.

Theorem 5.3. *All the points of Ω are fixed points of $\overline{\mathcal{G}}$.*

Proof. Given $\mathbf{x} \in \Omega$, clearly $\mathsf{E}[\mathbf{X}_{1:\lambda} | \mathbf{p} = \mathbf{x}] = \mathbf{x}$, because the probability of sampling an individual different from $\mathbf{x}$ given $\mathbf{p} = \mathbf{x}$ is zero. Hence:

$$\overline{\mathcal{G}}(\mathbf{x}) = \mathsf{E}[\mathbf{X}_{1:\lambda} | \mathbf{p} = \mathbf{x}] = \mathbf{x}.$$

□

Before introducing Theorem 5.4, we define what we mean by a "local optimum" for the Hamming distance. We also give a result borrowed from page 126 in Sheinerman (1996), that will be useful in the proof of the theorem.

Definition 5.1. *Given a real function f defined in Ω, a point $\mathbf{x}$ is a local minimum for the Hamming distance d_H if:*

$$\textit{for all } \mathbf{x}' \in \Omega \textit{ such that } d_H(\mathbf{x}', \mathbf{x}) = \sum_{l=1}^{n} |x'_l - x_l| = 1 \Rightarrow f(\mathbf{x}') \geq f(\mathbf{x}).$$

Lemma 5.2. *Let $\mathbf{x}$ be a fixed point of a discrete dynamical system $\mathcal{G}$, and $D\mathcal{G}(\mathbf{x})$ the Jacobian matrix of $\mathcal{G}(\mathbf{x})$.*

- *If the eigenvalues of $D\mathcal{G}(\mathbf{x})$ all have absolute value less than 1, then $\mathbf{x}$ is a stable fixed point of $\mathcal{G}$.*
- *If some eigenvalue of $D\mathcal{G}(\mathbf{x})$ has absolute value greater than 1, then $\mathbf{x}$ is an unstable fixed point of $\mathcal{G}$.*

Theorem 5.4. *Given a real function f defined on Ω, we have the following:*

- *All the local optima of f with respect to the Hamming distance are stable fixed points of $\overline{\mathcal{G}}$.*
- *All the nonlocal optima of f with respect to the Hamming distance are unstable fixed points of $\overline{\mathcal{G}}$.*

Proof. We use the Lemma 5.2 in order to prove these affirmations.

We first show that all the local optima are stable points; and, in a second step, the last result.

Let $\mathbf{x} \in \Omega$ be a local optimum of a function f (i.e., all the individuals in Ω whose Hamming distance from $\mathbf{x}$ is one, precede $\mathbf{x}$ in the order imposed by f in Ω). We will see that in this case $D\overline{\mathcal{G}}|_{\mathbf{x}} = \mathbf{0}$. In fact, we will see that

$$\left.\frac{\partial \overline{\mathcal{G}}_r}{\partial p_l}\right|_{\mathbf{x}} = 0 \quad \text{for all} \quad r, l = 1, 2, \ldots, n, \tag{5.7}$$

where $\overline{\mathcal{G}}_r$ is the r^{th} component of equation (5.6):

$$\overline{\mathcal{G}}_r(\mathbf{p}) = \sum_{i=1}^{m} y_{i,r} q_{\mathbf{x}_i}(\mathbf{p}) \left(\sum_{k=1}^{\lambda} \left(\sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{p}) \right)^{k-1} \left(\sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{p}) \right)^{\lambda-k} \right).$$

To show equation (5.7), we must first take into account the following results which can be checked easily using the definition of $q_{\mathbf{x}}(\mathbf{p})$ (equation (5.5)):

$$q_{\mathbf{x}_k}(\mathbf{x}) = 0 \quad \text{for all} \quad \mathbf{x} \neq \mathbf{x}_k \tag{5.8}$$

$$q_{\mathbf{x}_k}(\mathbf{x}_k) = 1 \text{ for all } \mathbf{x}_k \in \Omega \tag{5.9}$$

$$\left.\frac{\partial q_{\mathbf{x}_k}}{\partial p_l}\right|_{\mathbf{x}_k} = \begin{cases} 1 & \text{if } x_{k,l} = 1 \\ -1 & \text{if } x_{k,l} = 0 \end{cases} \tag{5.10}$$

$$\left.\frac{\partial q_{\mathbf{x}_k}}{\partial p_l}\right|_{\mathbf{x}} = 0 \quad \text{for all} \quad \mathbf{x} \text{ such that } d_H(\mathbf{x}, \mathbf{x}_k) \geq 2 \tag{5.11}$$

$$\left.\frac{\partial q_{\mathbf{x}_k}}{\partial p_l}\right|_{\mathbf{x}} = \begin{cases} 1 & \text{if } d_H(\mathbf{x}, \mathbf{x}_k) = 1 \text{ and } x_{k,l} = 1,\ x_l = 0 \\ -1 & \text{if } d_H(\mathbf{x}, \mathbf{x}_k) = 1 \text{ and } x_{k,l} = 0,\ x_l = 1 \end{cases} \tag{5.12}$$

$$\left.\frac{\partial q_{\mathbf{x}_k}}{\partial p_l}\right|_{\mathbf{x}} = 0 \text{ if } d_H(\mathbf{x}, \mathbf{x}_k) = 1 \text{ and } x_{k,l} = x_l. \tag{5.13}$$

The partial derivative of $\overline{\mathcal{G}}_r$ with respect to p_l can be expressed as:

$$\left.\frac{\partial \overline{\mathcal{G}}_r}{\partial p_l}\right|_{\mathbf{x}} = \sum_{i=1}^{m} \frac{\partial}{\partial p_l} \left[y_{i,r} q_{\mathbf{x}_i} \left(\sum_{k=1}^{\lambda} \left(\sum_{j=1}^{i-1} q_{\mathbf{x}_j} \right)^{k-1} \left(\sum_{j=1}^{i} q_{\mathbf{x}_j} \right)^{\lambda-k} \right) \right] \Bigg|_{\mathbf{x}}. \tag{5.14}$$

We analyze each adding term of $\partial \overline{\mathcal{G}}_r / \partial p_l|_{\mathbf{x}}$ separately and split it into three different cases. The partial derivative of a term of equation (5.14) must be written first ($y_{i,r}$ has been eliminated, because it is a constant term):

$$\begin{aligned} & \left. \frac{\partial q_{\mathbf{x}_i} \left(\sum_{k=1}^{\lambda} (\sum_{j=1}^{i-1} q_{\mathbf{x}_j})^{k-1} (\sum_{j=1}^{i} q_{\mathbf{x}_j})^{\lambda-k} \right)}{\partial p_l} \right|_{\mathbf{x}} \\ = & \left. \frac{\partial q_{\mathbf{x}_i}}{\partial p_l} \right|_{\mathbf{x}} \left(\sum_{k=1}^{\lambda} \left(\sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{x}) \right)^{k-1} \left(\sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{x}) \right)^{\lambda-k} \right) \\ + & q_{\mathbf{x}_i}(\mathbf{x}) \left. \frac{\partial \left(\sum_{k=1}^{\lambda} (\sum_{j=1}^{i-1} q_{\mathbf{x}_j})^{k-1} (\sum_{j=1}^{i} q_{\mathbf{x}_j})^{\lambda-k} \right)}{\partial p_l} \right|_{\mathbf{x}}. \end{aligned} \tag{5.15}$$

We split the problem into the following three different cases.

1 Let $\mathbf{x}_i \in \Omega$ be an individual such that $d_H(\mathbf{x}, \mathbf{x}_i) \geq 2$. In this case, using $\partial q_{\mathbf{x}_i}/\partial p_l|_{\mathbf{x}} = 0$ (equation (5.11)) and $q_{\mathbf{x}_i}(\mathbf{x}) = 0$ (equation (5.8)), equation (5.15) has a value of 0.

2 Let $\mathbf{x}_i \in \Omega$ be an individual such that $d_H(\mathbf{x}, \mathbf{x}_i) = 1$. In this case, in the second term we again have $q_{\mathbf{x}_i}(\mathbf{x}) = 0$ (equation (5.8)) but, in the first, $\partial q_{\mathbf{x}_i}/\partial p_l|_{\mathbf{x}}$ could be different from zero. However in this case (because $\mathbf{x}_i$ is before $\mathbf{x}$ in the order in Ω and then $q_{\mathbf{x}_j}(\mathbf{y}) = 0$ for all $j = 1, \ldots, i$) the second multiplicative term $\left(\sum_{k=1}^{\lambda}(\sum_{j=1}^{i-1} q_{\mathbf{x}_j})^{k-1}(\sum_{j=1}^{i} q_{\mathbf{x}_j})^{\lambda-k}\right)$ has a value of zero in $\mathbf{x}$. Hence equation (5.15) has a value of zero for this kind of individual.

3 Finally we take into account the term corresponding to individual $\mathbf{x}$. If $x_l = 0$ this term does not appear in the sum. Otherwise ($x_l = 1$), if i represents the place that individual $\mathbf{x}$ takes in the ordering of Ω ($\mathbf{x} = \mathbf{x}_i$), equation (5.15) can be expressed by:

$$\begin{aligned}
&\frac{\partial q_{\mathbf{x}}}{\partial p_l}\Big|_{\mathbf{x}} \left(\sum_{k=1}^{\lambda}\left(\sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{x})\right)^{k-1}\left(\sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{x})\right)^{\lambda-k}\right) \\
&+ \quad q_{\mathbf{x}}(\mathbf{y})\left.\frac{\partial\left(\sum_{k=1}^{\lambda}(\sum_{j=1}^{i-1} q_{\mathbf{x}_j})^{k-1}(\sum_{j=1}^{i} q_{\mathbf{x}_j})^{\lambda-k}\right)}{\partial p_l}\right|_{\mathbf{x}} \\
&= \quad \frac{\partial q_{\mathbf{x}}}{\partial p_l}\Big|_{\mathbf{x}} \left(\sum_{k=1}^{\lambda} A_i(\mathbf{x})^{k-1} B_i(\mathbf{x})^{\lambda-k}\right) + q_{\mathbf{x}}(\mathbf{y})\left.\frac{\partial\left(\sum_{k=1}^{\lambda} {A_i}^{k-1} {B_i}^{\lambda-k}\right)}{\partial p_l}\right|_{\mathbf{x}}
\end{aligned} \tag{5.16}$$

where $A_i(\mathbf{x}) = \sum_{j=1}^{i-1} q_{\mathbf{x}_j}(\mathbf{x})$ and $B_i(\mathbf{x}) = \sum_{j=1}^{i} q_{\mathbf{x}_j}(\mathbf{x})$.

In the next reasoning, note that $A_i(\mathbf{x}) = 0$ and $B_i(\mathbf{x}) = 1$.

The first term of equation (5.16) is:

$$\frac{\partial q_{\mathbf{x}}}{\partial p_l}\Big|_{\mathbf{x}} \left(B_i(\mathbf{x})^{\lambda-1} + A_i(\mathbf{x})B_i(\mathbf{x})^{\lambda-2} + \cdots + A_i(\mathbf{x})^{\lambda-1}\right) = 1.$$

The second term of equation (5.16) can be expressed as:

$$\begin{aligned}
&\underbrace{q_{\mathbf{x}}(\mathbf{x})}_{=1}\left.\frac{\partial\left(B_i^{\lambda-1} + A_i B_i^{\lambda-2} + \cdots + A_i^{\lambda-1}\right)}{\partial p_l}\right|_{\mathbf{x}} \\
&= \quad (\lambda-1)B_i(\mathbf{x})^{\lambda-2}\frac{\partial B_i}{\partial p_l}\Big|_{\mathbf{x}} + \frac{\partial A_i}{\partial p_l}\Big|_{\mathbf{x}} B_i(\mathbf{x})^{\lambda-2} \\
&+ \quad A_i(\mathbf{x})(\lambda-2)B_i(\mathbf{x})^{\lambda-3}\frac{\partial B_i}{\partial p_l}\Big|_{\mathbf{x}} + \ldots + (\lambda-1)A_i(\mathbf{x})^{\lambda-2}\frac{\partial A_i}{\partial p_l}\Big|_{\mathbf{x}} \\
&= \quad (\lambda-1)B_i(\mathbf{x})^{\lambda-2}\frac{\partial B_i}{\partial p_l}\Big|_{\mathbf{x}} + \frac{\partial A_i}{\partial p_l}\Big|_{\mathbf{x}} B_i(\mathbf{x})^{\lambda-2}.
\end{aligned} \tag{5.17}$$

Substituting again the values of $A_i(\mathbf{x})$ and $B_i(\mathbf{x})$ in equation (5.17):

$$
\begin{aligned}
&(\lambda-1)\sum_{j=1}^{i}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}+\sum_{j=1}^{i-1}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}} \\
&= (\lambda-1)\left(\underbrace{\sum_{\substack{d_H(\mathbf{x}_j,\mathbf{x})\geq 2\\ j<i}}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}}_{=0}+\sum_{d_H(\mathbf{x}_j,\mathbf{x})=1}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}+\underbrace{\left.\frac{\partial q_{\mathbf{x}}}{\partial p_l}\right|_{\mathbf{x}}}_{=1}\right) \\
&+ \left(\underbrace{\sum_{\substack{d_H(\mathbf{x}_j,\mathbf{x})\geq 2\\ j<i}}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}}_{=0}+\sum_{d_H(\mathbf{x}_j,\mathbf{x})=1}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}\right).
\end{aligned}
$$

Taking into account that $\mathbf{x}_k$ is such that $d_H(\mathbf{x},\mathbf{x}_k)=1$, $x_{k,l}=0$, and $x_l=1$ we find that the second term of equation (5.16) is:

$$
(\lambda-1)\left(\underbrace{\left.\frac{\partial q_{\mathbf{x}_k}}{\partial p_l}\right|_{\mathbf{x}}}_{=-1}+\underbrace{\sum_{\substack{d_H(\mathbf{x}_j,\mathbf{x})=1\\ x_{j,l}=x_l}}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}}_{=0}+1\right)+\left(\underbrace{\left.\frac{\partial q_{\mathbf{x}_k}}{\partial p_l}\right|_{\mathbf{x}}}_{=-1}+\underbrace{\sum_{\substack{d_H(\mathbf{x}_j,\mathbf{x})=1\\ x_{j,l}=x_l}}\left.\frac{\partial q_{\mathbf{x}_j}}{\partial p_l}\right|_{\mathbf{x}}}_{=0}\right)=-1.
$$

Hence $D\overline{\mathcal{G}}(\mathbf{x})=\mathbf{0}$ for all local optimum points $\mathbf{x}\in\Omega$ and all the eigenvalues have a value of zero. Moreover, we have shown that the local optimum points for a function f are stable fixed points of the discrete dynamical system $\overline{\mathcal{G}}$.

In the case of a point $\mathbf{x}\in\Omega$ that is not a local optimum, following similar arguments to the previous case, it can be shown that:

$$
\left.\frac{\partial\overline{\mathcal{G}}_r}{\partial p_l}\right|_{\mathbf{x}}=0 \text{ for all } r\neq l.
$$

In all the previous cases the adding terms have a value of zero, except for the case that there exists $\mathbf{x}'$ such that $d(\mathbf{x},\mathbf{x}')=1$, $x'_l\neq x_l$ and in addition $f(\mathbf{x}')>f(\mathbf{x})$ ($\mathbf{x}$ is before $\mathbf{x}'$ in the order imposed by f in Ω), the adding terms corresponding to $\mathbf{x}$ and $\mathbf{x}'$ are different from zero but their sum is zero.

In the case $r=l$, there exists $l\in\{1,\ldots,n\}$ such that $\left.\frac{\partial\overline{\mathcal{G}}_l}{\partial p_l}\right|_{\mathbf{x}}>1$.

□

From Theorem 5.4 it can be deduced that when the value of α is near 0, then the PBIL algorithm can only converge to the local optima of f. So, the PBIL algorithm converges to the global optimum in unimodal functions (functions that have exactly one local optimum).

5.3 Other Works that Mathematically Analyze PBIL

We include a brief summary of other works on theoretical aspects of PBIL that can be found in the literature.

5.3.1 Reinforcement Learning, PBIL, and Gradient Dynamical Systems

Berny (2000) shows that Reinforcement Learning and PBIL algorithms can be derived from gradient dynamical systems acting on the probability vectors $\mathbf{p}$.

To do so, the author shows the equivalence of searching for an optimal string of function f and searching for the probability distribution implied by probability vector $\mathbf{p}$ over Ω, that we denoted by $q(\mathbf{p})$, which maximizes the function expectation:

$$J_1(q(\mathbf{p})) = \mathsf{E}_{q(\mathbf{p})}[f] = \sum_{\mathbf{x}\in\Omega} q_{\mathbf{x}}(\mathbf{p}) f(\mathbf{x}),$$

(where $\mathsf{E}_{q(\mathbf{p})}[f]$ denotes the expectation of f with respect to the probability distribution implied by probability vector $\mathbf{p}$), or which maximizes the log-expectation of the exponential of the function:

$$J_2(q(\mathbf{p})) = log\mathsf{E}_{q(\mathbf{p})}[e^{\beta f}] \quad \text{with} \quad \beta > 0 \ .$$

If we try to optimize $J_1(q(\mathbf{p}))$ and $J_2(q(\mathbf{p}))$ by means of a gradient search and take into account that the probability distribution $q(\mathbf{p})$ depends on the probability vector $\mathbf{p}$, then two gradient dynamical systems can be obtained. The first for $J_1(q(\mathbf{p}))$ can be written as:

$$\begin{aligned} \mathbf{p}' &= \boldsymbol{\varphi}(\mathbf{p}) \\ \boldsymbol{\varphi}_i(\mathbf{p}) &= p_i(1-p_i)\frac{\partial J_1(q(\mathbf{p}))}{\partial p_i} \end{aligned}$$

and the second for $J_2(q(\mathbf{p}))$ as:

$$\begin{aligned} \mathbf{p}' &= \boldsymbol{\varphi}(\mathbf{p}) \\ \boldsymbol{\varphi}_i(\mathbf{p}) &= p_i(1-p_i)\frac{\partial J_2(q(\mathbf{p}))}{\partial p_i} \ . \end{aligned}$$

From the first dynamical system Reinforcement Learning can be obtained by using stochastic approximation with a comparison scheme. PBIL is obtained from the second dynamical system with a Lagrange technique and stochastic approximation.

The author carried out a stability analysis of vertices and states and concluded that Reinforcement Learning and PBIL perform as well as hill climbing, since they can only converge to locally optimal solutions. This result might appear to contradict the result in Section 5.1, but Berny obtains it because his stability analysis is carried out without taking into account parameter α, which we have proven that can be highly relevant to the convergence behavior of PBIL.

Similar developments were made in Berny (2000b) for real function optimization, but the author did not give any stability results.

5.3.2 Limit Behavior of PBIL

Höhfeld and Rudolph (1997) present an analysis of the convergence behavior of the PBIL algorithm when the search space is $\Omega = \{0,1\}^l$. They prove that a simplified version of PBIL's update rule (only the best of λ trials vectors is involved in updating the vector of probabilities):

$$\mathbf{p}_t = (1-\alpha)\mathbf{p}_{t-1} + \alpha \mathbf{x}_{1:\lambda}^{(t-1)}$$

ensures convergence with probability one to the global optimum in the case of pseudo-boolean linear functions.

The aim of these authors is to show that the stochastic sequence $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ converges in mean (and therefore in probability) to the global optimum of the search space. In order to do this, they require that, for a linear pseudo-boolean function:

$$\lim_{t\to\infty} \mathsf{E}[\mathbf{p}_t] = \mathbf{y}^*$$

where $\mathsf{E}[\mathbf{p}_t]$ is the expectation of the probability vector at step t, and $\mathbf{y}^*$ is the optimum point in Ω.

Thus, studying the (deterministic) sequence $\{\mathsf{E}[\mathbf{p}_t]\}_{t=0,1,2,\ldots}$, the points in Ω to which PBIL's stochastic process $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ will eventually converge are identified and they obtain global convergence in mean for PBIL in linear pseudo-boolean functions. Nonetheless, as we have seen in this chapter, in PBIL the convergence in mean reveals little. We have found that for many values of α the algorithm never converges to a good solution, although the mean may converge.

5.3.3 How α Controls PBIL Performance

To explain why PBIL depends on parameter α, Shapiro (2003) considers the performance of this algorithm with $\mu = 1$ in flat search spaces (all candidate solutions are equally good), and with $\mathbf{p}_0 = (1/2,\ldots,1/2)$. So at each step, t, the best point is $\mathbf{x}_{1:\lambda}^{(t)} = \mathbf{x}^*$ and:

$$\mathsf{E}[(X^*)] = \mathbf{p}_0$$

which yields

$$\mathsf{E}[\mathbf{p}_{t+1}] = \mathbf{p}_t,$$

and the parameters remain unchanged on average. But this does not imply that the system goes nowhere. Considering $D(t)$ as the average distance between the parameters $\mathbf{p}_t$ and $\mathbf{p}_0 = (1/2,\ldots,1/2)$:

$$D(t) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2} - p_i^{(t)}\right)^2,$$

$D(t)$ is zero when $\mathbf{p}_t = (1/2,\ldots,1/2)$ and takes the value $1/4$ when all the parameters $p_i^{(t)}$ are 0 or 1. Shapiro computes the expectation of $D(t)$, and finds that:

$$\mathsf{E}[D(t)] = \frac{1-(1-\alpha^2)^t}{4}$$

which converges to 1/4 in time τ:

$$\tau = \frac{-1}{\log(1-\alpha^2)} \approx 1/\alpha^2, \quad \text{when } \alpha \to 0.$$

This explains why α must be chosen very small to avoid premature convergence.

Shapiro (2003) proves that as the system size increases, α must go to zero in a problem-dependent way. For the *Needle-In-A-Haystack* problem (which has a landscape with flat subspaces) α must be exponentially small in n, while for the *OneMax* problem (which also has a landscape with flat subspaces) α must be smaller than $r/\sqrt{n}$, with $r \in \mathbb{R}$.

He also proposes a modification of the algorithm by ensuring that the dynamics obeys a condition called "detailed balance", which is a reversibility condition used in the Markov chain Monte Carlo method to ensure that the Markov chain converges to a given probability distribution. In simulations the author sees that this modification makes the algorithm largely insensitive to α and can improve the run time because it can use a higher α.

5.4 Summary

There have been two mathematical tools used in this chapter for the theoretical study of PBIL: Markov chains and discrete dynamical systems.

The analysis of PBIL based on Markov chains lets us show that for PBIL with $\lambda = 2$ and $\mu = 1$ applied to the *OneMax* function :

$$\mathsf{P}\left(\lim_{t\to\infty} \mathbf{p}_t = (a,b)\right) \longrightarrow 1$$

when $\alpha \to 1$, $\mathbf{p}_0 \to (a,b)$, and $(a,b) \in \{0,1\}^2$. This result implies a strong dependence of PBIL convergence on the initial vector $\mathbf{p}_0$ and on the α parameter value. Although when the value of α is small, and the experimental results seem to be more stable, we can not conclude that PBIL converges to the optimum.

We have opened a new approach to the theoretical study of PBIL: using discrete dynamical systems. A discrete dynamical system is associated with the PBIL algorithm. We demonstrate that the behavior of the PBIL algorithm follows the iterates of the discrete dynamical system for a long time when the parameter α is near zero. We show – performing a stability analysis – that all the points of the search space are fixed points of the dynamical system, and that the local optimum points for the function to optimize coincide with the stable fixed points. Hence it can be deduced that the PBIL algorithm converges to the global optimum in unimodal functions.

Chapter 6

Analysis of the UMDA algorithm

This chapter is devoted to the UMDA algorithm, which will be analyzed, as in previous cases, using two mathematical tools: Markov chains and discrete dynamical systems.

Under the first framework, an empirical analysis of the convergence behavior of the algorithm is described. This study can be found in Section 6.1 and provides some insights into how the absorption probability to the optimum and the expected absorption times evolve when the size of the population increases. This analysis was published earlier in González et al. (2003). Section 6.2 is devoted to modeling UMDA with infinite population and proportionate selection using discrete dynamical systems. The modeling sets out from the schema used in Section 5.2 and arrives at the same expression as reported by Mühlenbein and Mahnig (2002a). The chapter finishes with the conclusions in Section 6.3.

6.1 Analysis of the UMDA Algorithm Modeled by Markov Chains

In Chapter 4, and based on what we call Markov model I (whose absorbing states correspond to the individuals of the search space), we studied the convergence behavior of UMDA. There we concluded that not to use the Laplace correction could imply that the chain which models the algorithm (and then the algorithm itself) might be trapped at any point of the search space. In such a situation some natural questions immediately arise. What is the absorption probability to any absorbing state (particularly to the optimal point)?; how long must we wait until the set of absorbing states are visited? Answering these questions enables us to learn the effects that changes in population size have on the absorption probability to the optimum and the expected absorption times.

The aim of present section is to offer an experimental analysis of the above questions, when UMDA is used to maximize a number of pseudo-boolean functions. This experimental study was carried out in the hope that further theoretical studies can be based on it.

The analysis is based on modeling UMDA using a type II Markov chain. The state space is given by all the possible probability distributions, $p_t(\mathbf{x})$, that can be formed from a population of selected individuals of size N. Since we have choosen a version of UMDA where Laplace correction is not used in estimating the parameters, the absorbing states of the chain – as in the case of the type I Markov chain mentioned above – coincide with the points of the search space.

We calculate the absorption probability to the optimum and the expected absorption times for UMDA on the maximization of an example of linear, pseudo-modular, unimax and almost positive functions, for different values of selected population size N and individual length n. This information is used to provide some insights into how the absorption probability to the optimum and the expected absorption times evolve when population size increases. The results show the different behaviors of the algorithm in the analyzed functions.

6.1.1 The Version of UMDA to be Analyzed and other General Considerations

In this section it is suppose that UMDA is used to solve the following optimization problem:

$$\max_{\mathbf{x}\in\Omega} f(\mathbf{x}) \tag{6.1}$$

where $f : \Omega \longrightarrow \mathbb{R}$ is the objective function, and $\Omega = \{0,1\}^n$, with $n \in \mathbb{N}$ being the search space. The cardinality of the search space is $|\Omega| = 2^n = m$. We denote by $\Omega = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ the different elements of Ω.

The version of UMDA analyzed here to solve problem (6.1) is very simple. In this version elitism and the Laplace correction are not taken into account (in the estimation of parameters maximum-likelihood is used) it works as follows: at each step t a population of size $2N$, D_{t-1} is maintained, the N best individuals of D_{t-1} (truncation selection) are selected, constituting the population of selected individuals D^{Se}_{t-1}. Later, using these selected individuals the joint probability distribution $p_t(\mathbf{x})$ is estimated. Finally we obtain the new population D_t sampling $2N$ individuals from $p_t(\mathbf{x})$.

6.1.2 A Markov Chain that Models UMDA

Our objective is to calculate (experimentally) the absorption probability to the optimum and the expected absorption times for UMDA applied to the maximization of a number of functions, for different values of selected population size N and individual length n. In order to do that we have modeled UMDA by means of Markov chains. Our analysis is inspired by De Jong et al. (1995) where the authors use a transient Markov chain analysis to model and understand the behavior of finite population Genetic Algorithms for Function Optimization.

Now we describe the Markov chain chosen to model the UMDA algorithm introduced above. In this case the Markov model II is used (see Chapter 4).

Given that the probability distribution at step t only depends on the probability distribution at step $t-1$, the above described UMDA algorithm can be modeled using a Markov chain, where the states of the chain are all the possible probability distributions, $p_t(\mathbf{x})$, that can be formed from a population of selected individuals of size N.

Taking into account that each probability distribution can be represented as a probability vector $\mathbf{p} = (p_1 \ldots, p_n)$ (where p_i is the probability of obtaining a 1 in the i^{th} gene) the set of states of the chain can be expressed as:

$$E = \left\{ (p_1, \ldots, p_n) \,\middle|\, p_i \in \{0, \frac{1}{N}, \ldots, \frac{N-1}{N}, 1\}, \quad i \in \{1, \ldots, n\} \right\},$$

where the i^{th} component of $\mathbf{p}$, p_i is obtained using the selected population at previous step D^{Se} as follows:

$$p_i = \frac{\sum_{j=1}^{N} \delta_j(X_i = 1 | D^{Se})}{N}.$$

Taking into account that the cardinality of the state space is $c = |E| = (N+1)^n$, E can be written as follows:

$$E = \{\mathbf{p}_1, \ldots, \mathbf{p}_c\}.$$

Two question are important to note. On the one hand, here the subindex i in $\mathbf{p}_i$ does not denote the probability vector in generation i, but that $\mathbf{p}_i$ is the i^{th} element of set E. At each step t there exists an i such that $\mathbf{p}_t = \mathbf{p}_i$, with $i \in \{1, \ldots, c\}$. On the other hand, the cardinality of the state space, c, increases exponentially as the size of the individual increases. This is the reason why in order to have a reasonable computational cost, the experiments will be carried out for small values of n.

We also want to stress that the Markov chain is not irreducible. More precisely, the absorbing states of the Markov chain correspond to the individuals of the search space, while the rest are transient states, i.e. the states with some component not equal to 0 or 1. If we denote by A and T the set of absorbing states, and the set of transient states respectively, then:

$$\begin{aligned} A &= \{(p_1, \ldots, p_n) \mid p_i \in \{0,1\}, \quad i \in \{1, \ldots, n\}\}, \quad |A| = 2^n \\ T &= \{(p_1, \ldots, p_n) \mid \exists i \in \{1, \ldots, n\}, \quad p_i \notin \{0,1\}\}, \quad |T| = (N+1)^n - 2^n. \end{aligned}$$

To aid in comprehension, note that each absorbing state is associated with a uniform population of selected individuals (which is formed by N copies of the same individual). For example the absorbing state $\mathbf{p} = (1, \ldots, 1) \in A$ is associated with the population formed by N copies of the individual $(1, \ldots, 1)$.

Taking into account that the absorbing states are the individuals of the search space, the algorithm could converge to any of them. Therefore it is interesting to calculate the absorption probabilities to any absorbing state (especially to the optimal point) and the expected absorption times.

6.1.2.1 Calculation of the Absorption Probabilities and the Expected Absorption Times

Let's suppose that the states of the Markov chain are ordered. The first states are the transient states, while the absorbing states are in the last places. Therefore the transition matrix $\mathbf{P}$, associated with the Markov chain can be written as follows:

$$\mathbf{P} = \begin{pmatrix} \mathbf{R} & \mathbf{Q} \\ \mathbf{0} & \mathbf{I}_{2^n} \end{pmatrix},$$

where $\mathbf{I}_{2^n}$ is the identity matrix of dimension 2^n. Its entries represent the probability of going from one absorbing state to another absorbing state. Matrix $\mathbf{Q}$ has dimension $((N+1)^n - 2^n) \times 2^n$ and its entries represent the probability of going from a transient state to an absorbing state. Finally we describe matrix $\mathbf{R}$, which is a matrix of dimension $((N+1)^n - 2^n) \times ((N+1)^n - 2^n)$ and contains the probabilities of going from one transient state to another transient state.

The formulas for the absorption probability and expected absorption times can be obtained from matrices: $\mathbf{W} = (\mathbf{I}_{2^n} - \mathbf{R})^{-1}$ and $\mathbf{U} = \mathbf{WQ}$. Both results can be consulted in Section (2.3).

The *expected absorption time m_i starting from the i^{th} state* is given by the expression:

$$m_i = \sum_j w_{ij}.$$

The *absorption probabilities to an absorbing state j starting from the i^{th} state*, u_{ij} are given by the elements of the matrix $\mathbf{U} = (u_{ij})$.

It is important to note that the computational cost in the calculation of the above quantities is given by the cost of inverting $\mathbf{W}$. Since the dimension of $\mathbf{W}$ is $(N+1)^n - 2^n \times (N+1)^n - 2^n$ it is directly related to the individual size n.

6.1.2.2 Calculation of the Transition Probability Matrix P

Calculating the transition probability matrix $\mathbf{P}$ is essential for obtaining the remaining quantities. Each entry of $\mathbf{P} = (P_{ij})$ is the probability of going from state $\mathbf{p}_i$ to state $\mathbf{p}_j$ in a step of the algorithm:

$$P_{ij} = \mathsf{P}(\mathbf{p}_j|\mathbf{p}_i), \text{ with } i, j \in \{1, \ldots, c\}. \tag{6.2}$$

To obtain each P_{ij} it is necessary to take into account all the possible populations that can be obtained sampling vector $\mathbf{p}_i$ and whose associated probability vector after selection is $\mathbf{p}_j$ (note that there exist some populations whose associated probability vector is the same). The elements of $\mathbf{P}$ can be obtained as follows:

$$P_{ij} = \mathsf{P}(\mathbf{p}_j|\mathbf{p}_i) = \sum_{D_j} \mathsf{P}(\text{obtain population } D_j \mid \text{vector } \mathbf{p}_i \text{ is sampled}) \tag{6.3}$$

where D_j varies in the populations (of size N) that can be obtained sampling $\mathbf{p}_i$. From those D_j, by selecting individuals, D_j^{Se} are obtained, after which, the vector $\mathbf{p}_j$ is estimated. (Note that in the calculation of $\mathbf{p}_j$ only the individuals of D_j^{Se} are used.) Hence, in order to calculate (6.3) it is necessary to obtain all populations D_j.

Let D_j^{Se} be the selected population from population D_j, and:

$$D_j^{Se} = \{\mathbf{x}_{1j}, \ldots, \mathbf{x}_{Nj}\}, \quad \mathbf{x}_{ij} \in \Omega, \quad \forall i \in \{1, \ldots N\}, \tag{6.4}$$

where each $\mathbf{x}_{ij} = (x_{ij}^1, \ldots, x_{ij}^n)$, $i \in \{1, \ldots N\}$.

If $\mathbf{p}_j = (\frac{a_1}{N}, \ldots, \frac{a_n}{N})$ then the population D_j^{Se} has a_k individuals with a "1" in the k^{th} position, $k \in \{1, \ldots, n\}$.

The rest of N individuals of D_j can vary over no better individuals than the worst individual of D_j^{Se}. Therefore, in order to find all the populations D_j it is sufficient to know D_j^{Se}, and vary the remaining N individuals of D_j over the remaining possibilities. To find D_j^{Se} we have to solve the following equations system with $2^n - 2n$ degrees of freedom:

$$\begin{aligned}
\sum_{k=1}^{N} x_{kj}^1 &= a_1 \\
&\cdots \\
\sum_{k=1}^{N} x_{kj}^n &= a_n \\
\sum_{k=1}^{N} 1 - x_{kj}^1 &= N - a_1 \\
&\cdots \\
\sum_{k=1}^{N} 1 - x_{kj}^n &= N - a_n.
\end{aligned} \tag{6.5}$$

Unfortunately to solve the equations, system (6.5) has a high computational cost due to the fact that it has an exponential number of degrees of freedom. This is the reason why we have to estimate experimentally the elements of matrix **P**. In Section 6.1.4.1 we explain in detail how the estimation was carried out.

6.1.3 The Functions Used

In the experiments we have chosen one particular function of each of the following classes of pseudo-boolean functions: linear, pseudo-modular, unimax and almost positive functions. In this section an introduction to these classes of functions can be found. First of all we define a pseudo-boolean function:

Definition 6.1. *A function f is said to be **pseudo-boolean** if*

$$f : \{0,1\}^n \longrightarrow I\!R.$$

Every pseudo-boolean function can be (nonuniquely) expressed as a polynomial in its variables $x_1, x_2, \ldots, x_l$ and their complements $\bar{x}_i = 1 - x_i$ of the following form (Crama et al., 1990):

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{m} c_i \prod_{j \in I(i)} x_j^{\alpha_{ji}},$$

where m denotes the number of terms of the polynomial, $I(i) \subseteq \{1, 2 \ldots, n\}$ denotes the index set of term i and $\alpha_{ji} \in \{0,1\}$ is used to identify complemented variables by the convention $x^1 = x$ and $x^0 = \bar{x}$.

In the analysis of each objective function we consider that the neighboring solutions $N(\mathbf{x})$ of a given solution $\mathbf{x} \in \{0,1\}^n$, are given for all the points at the Hammimg distance equal or less than 1, i.e.:

$$N(\mathbf{x}) = \left\{ \mathbf{y} \in \{0,1\}^n \quad | \quad d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \leq 1 \right\}$$

6.1.3.1 Linear Functions

Definition 6.2. *A pseudo-boolean function $f : \{0,1\}^n \longrightarrow I\!R$ is said to be **linear** if*

$$f(\mathbf{x}) = c_0 + \sum_{i=1}^{n} c_i x_i,$$

where coefficients $c_i \in I\!R$, $i = 0, \ldots, n$.

This is the simplest case of a pseudo-boolean function.

The particular linear function analyzed in this work is the function:

$$f(\mathbf{x}) = c_0 + \sum_{i=1}^{n} c_i x_i, \quad x_i \in \{0,1\}, \quad c_i \in I\!R \text{ such that } c_i > \sum_{j=0}^{i-1} c_j. \tag{6.6}$$

It is clear that $(1, \ldots, n)$ is the only global maximum for the function (6.6).

6.1.3.2 Pseudo-modular Functions

Definition 6.3. *A pseudo-boolean function $f : \{0,1\}^n \longrightarrow I\!R$ is said to be* ***pseudo-modular*** *if*

$$min\{f(\mathbf{x}), f(\mathbf{y})\} \leq max\{f(\mathbf{x} \wedge \mathbf{y}), f(\mathbf{x} \vee \mathbf{y})\}$$
$$min\{f(\mathbf{x} \wedge \mathbf{y}), f(\mathbf{x} \vee \mathbf{y})\} \leq max\{f(\mathbf{x}), f(\mathbf{y})\},$$

The pseudo-modular function we have used is:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \quad x_j \in \{0,1\}. \tag{6.7}$$

For fitness function (6.7) the optimal solution is $(1, \ldots, 1)$, and the value of $f(\mathbf{x})$ at this point is n.

6.1.3.3 Unimax Functions

Definition 6.4. *Let $f : \{0,1\}^n \longrightarrow I\!R$ be a pseudo-boolean function. A point $\mathbf{x}^* \in \{0,1\}^n$ is called a* ***local maximum*** *of f if:*

$$f(\mathbf{x}^*) \geq f(\mathbf{x}), \text{ for all } \mathbf{x} \in N(\mathbf{x}^*).$$

A pseudo-boolean function $f : \{0,1\}^n \longrightarrow I\!R$ is said to be ***unimax*** *if there exists exactly one local maximal point $\mathbf{x}^* \in \{0,1\}^n$.*

In the experiments we have carried out, a well known unimax function is used, namely the long path function (Horn et al., 1994).

A long path is constructed as follows: Let P_l be a long path in odd dimension n. Create subpath S_{00} by prepending "00" to each point in path P_l, and another subpath S_{11} by prepending "11" in the reverse order of the path P_l. The bridge point is built from the last point in path P_l prepended by "01". Finally concatenate subpath S_{00}, the bridge point and subpath S_{11} to obtain path P_{n+2} of dimension $n+2$. The length of the paths is described by the recurrence equations:

$$\begin{aligned} |P_1| &= 2 \\ |P_{l+2}| &= 2|P_l| + 2, \end{aligned}$$

whose solution is $|P_l| = 3 \cdot 2^{(l-1)/2} - 1$ for odd $l \geq 1$.

Using the initial long path $P_1 = (0,1)$, the long path of dimension 3 is
$P_3 = (000, 001, 011, 111, 110)$. Table 6.1 shows the long path P_5.

Given a point $\mathbf{x}$ on a path P_l, define $Pos(\mathbf{x})$ to be the position of $\mathbf{x}$ on the path which is numbered from 0 to $3 \cdot 2^{(l-1)/2} - 2$. For a point not on the path P_l, define $Pos(\mathbf{x})$ to be -1. Then define the function $f(\mathbf{x})$ as:

$$f(\mathbf{x}) = -(3 \cdot 2^{(l-1)/2} - 2) + \begin{cases} Pos(\mathbf{x}) & \text{if } Pos(\mathbf{x}) \geq 0 \\ -\sum_{i=1}^{l} x_i & \text{otherwise.} \end{cases} \tag{6.8}$$

This function has been proved to be optimized in exponential time by local search algorithms.

$Pos(\mathbf{x})$	$\mathbf{x}$	$Pos(\mathbf{x})$	$\mathbf{x}$
0	00000	6	11110
1	00001	7	11111
2	00011	8	11011
3	00111	9	11001
4	00110	10	11000
5	01110		

Table 6.1. Long path P_5.

6.1.3.4 Almost Positive Functions

Definition 6.5. *A pseudo-boolean function $f : \{0,1\}^n \longrightarrow I\!R$ is said to be **almost positive** if the coefficients of all nonlinear terms are non-negative.*

The almost positive function analyzed in this paper is:

$$f(\mathbf{x}) = n - \sum_{i=1}^{n} x_i + (n+1) \prod_{i=1}^{n} x_i, \quad x_i \in \{0,1\}. \tag{6.9}$$

For the function (6.9), the optimal solution is $(1,\ldots,1)$, while the individual $(0,\ldots,0)$ is the second best (maximum) point.

It is useful to note the classification established in Crama (1999) . If we denote:

$$\begin{aligned} L &= \{f : \{0,1\}^n \longrightarrow I\!R \mid f \text{ is linear}\}, \\ P &= \{f : \{0,1\}^n \longrightarrow I\!R \mid f \text{ is pseudo-modular}\}, \\ U &= \{f : \{0,1\}^n \longrightarrow I\!R \mid f \text{ is unimax}\}, \end{aligned}$$

we find that:

$$L \subseteq P \subseteq U.$$

Notice that the subset of almost positive functions is not included in any of the previous ones.

6.1.4 Experimental Results

Our aim is to find the absorption probability to the optimum and the expected absorption times to some absorbing states when the UMDA algorithm is used to maximize the pseudo-boolean functions introduced in the previous section.

Once we have those quantities we analyze how they evolve when the size of population N varies. In other words, we want to know what is the optimal size of the population that maximizes the absorption probability and minimizes the expected absorption times.

We made our analysis when the size of the individual n and the population of selected individuals were:

- $n = 2$ and $2 \leq N \leq 8$
- $n = 3$ and $3 \leq N \leq 8$.

6.1.4.1 Estimation of the Transition Probability Matrix P

The computational complexity of the exact calculation of $\mathbf{P} = (P_{ij})$ forces us to estimate these values instead of carrying out an exact calculation.

To obtain the values of the j^{th} row of $\mathbf{P}$ corresponding to probability vector $\mathbf{p}_j$, we carry out k times the following two steps for each j:

- $\mathbf{p}_j$ is taken as the initial vector of probabilities, each time carrying out the basic steps of UMDA: (i) the initial population is obtained by sampling $\mathbf{p}_j$, (ii) the N best individuals are selected, giving us the selected population, (iii) the new vector of probabilities $\mathbf{p}_i$ is obtained from the selected population.
- After the previous step the obtained probability vector is picked up.

If we denote by k_i the number of times that we reach the state $\mathbf{p}_i$, then each value P_{ij}, $j \in \{1, \ldots, c\}$ of row j is estimated as $P_{ij} = \frac{k_j}{k}$.

It is clear that when the number of experiments carried out, k, increases, the estimation improves. In our experiments we choose k in order to obtain a reasonable computational cost. The number of experiment carried out was $k = 5,000,000$, which fixes the 5^{th} decimal of P_{ij}.

6.1.4.2 The Absorption Probability to the Optimum and the Expected Absorption Time to Some Absorbing State

In practice, the first probability vector used to be $(1/2, \ldots, 1/2)$, so we have calculated the absorption probability and the absorption time to the optimum from this state when N is even. In case of odd N we have used the mean of the probability vectors that are the nearest to $(1/2, \ldots, 1/2)$ (see Table 6.2).

N/ n	2	3
2	$(\frac{1}{2}, \frac{1}{2})$	
3	$(\frac{1}{3}, \frac{2}{3})$ $(\frac{2}{3}, \frac{1}{3})$	$(\frac{1}{3}, \frac{1}{3}, \frac{2}{3})$ $(\frac{1}{3}, \frac{2}{3}, \frac{1}{3})$ $(\frac{2}{3}, \frac{1}{3}, \frac{1}{3})$ $(\frac{1}{3}, \frac{2}{3}, \frac{2}{3})$ $(\frac{2}{3}, \frac{1}{3}, \frac{2}{3})$ $(\frac{2}{3}, \frac{2}{3}, \frac{1}{3})$
4	$(\frac{2}{4}, \frac{2}{4})$	$(\frac{2}{4}, \frac{2}{4}, \frac{2}{4})$
5	$(\frac{2}{5}, \frac{3}{5})$ $(\frac{3}{5}, \frac{2}{5})$	$(\frac{2}{5}, \frac{2}{5}, \frac{3}{5})$ $(\frac{2}{5}, \frac{3}{5}, \frac{2}{5})$ $(\frac{3}{5}, \frac{2}{5}, \frac{2}{5})$ $(\frac{2}{5}, \frac{3}{5}, \frac{3}{5})$ $(\frac{3}{5}, \frac{2}{5}, \frac{3}{5})$ $(\frac{3}{5}, \frac{3}{5}, \frac{2}{5})$
6	$(\frac{3}{6}, \frac{3}{6})$	$(\frac{3}{6}, \frac{3}{6}, \frac{3}{6})$
7	$(\frac{3}{7}, \frac{4}{7})$ $(\frac{4}{7}, \frac{3}{7})$	$(\frac{3}{7}, \frac{3}{7}, \frac{4}{7})$ $(\frac{3}{7}, \frac{4}{7}, \frac{3}{7})$ $(\frac{4}{7}, \frac{3}{7}, \frac{3}{7})$ $(\frac{3}{7}, \frac{4}{7}, \frac{4}{7})$ $(\frac{4}{7}, \frac{3}{7}, \frac{4}{7})$ $(\frac{4}{7}, \frac{4}{7}, \frac{3}{7})$
8	$(\frac{4}{8}, \frac{4}{8})$	$(\frac{4}{8}, \frac{4}{8}, \frac{4}{8})$

Table 6.2. Starting probability vectors chosen.

6.1.5 Summarizing the Results

The results can be seen in Figure 6.1. The absorption probability to the optimum (and the expected absorption time to some absorbing state) is given in a graph, where the y axis shows the absorption probability (resp. the expected absorption time) and the x axis shows the size N of the population. The same graph shows the results for $n = 2$ and $n = 3$.

Several comments can be made in view of the graphs. We can distinguish the behavior of the first three functions (the easiest to optimize) from the almost positive.

As was expected, for the first three functions, the absorption probability increases with N. This probability is near to one in the linear and pseudo-modular functions, which means that in almost all executions the algorithm will converge to the optimum. On the other hand, in the unimax function this probability is lower than 0.6. However, it seems that the growth of this probability with N is higher in this third function. Similarly this probability is smaller when n increases. The small number of generations to reach convergence is noteworthy.

We obtained surprising results regarding the expected absorption times. In the linear and pseudo-modular functions this time does not increase with population size. We think it is related to the absorption probability. Given that the absorption probability is higher the algorithm converges faster. On the other hand in the unimax function this time increases linearly with N. In this last case the function is harder to optimize than the others so the algorithm needs more time to converge. The same as before, absorption time increases with n.

The case of the almost positive function is the most interesting. While with $n = 2$ the absorption probability increases a little with N, in dimension 3 this probability decreases, apparently going to 0 as N increases. It seems the algorithm can be absorbed by the local optimum point $(0, \ldots, 0)$ (see Figure 6.2). So this function is very difficult to optimize with UMDA.

This analysis also shows the behavior of the algorithm when the complexity of the function increases: the absorption probability decreases while the expected absorption time increases. Even in the almost positive function the absorption probability approaches zero when N increases.

6.2 UMDA and Dynamical Systems

Once we have carried out the analysis for PBIL by means of discrete dynamical systems (see Chapter 5) we wonder if the same dynamical system model used in the case of PBIL could be used to model UMDA. There were two reasons for posing this question. On one hand the fact that UMDA can be considered as a particular case of PBIL with $\alpha = 1$. On the other, UMDA and PBIL are the two only algorithms in the family of EDAs which have the property that their probability distributions can be represented as probability vectors (because their probability models consider only first-order statistics), and the key question in the analysis of PBIL is to associate a discrete dynamical system with the algorithm such that the trajectories followed by the probability vectors will be related to the iterations of that discrete dynamical system.

Unfortunately we realize that such an approach is not valid: we only have proven that the dynamical system associated with PBIL follows the trajectories of probability vectors of PBIL when α is near zero, and UMDA is a PBIL algorithm with $\alpha = 1$, so we can not ensure that the dynamical system used for PBIL can represent the dynamics of UMDA. Nonetheless the model schema used in the case of PBIL is valid for UMDA until the moment when we prove that the behavior of the PBIL algorithm follows the iterates of the discrete dynamical system for a long time when the parameter α is near zero. From then on, we are forced to use a discrete dynamical system model which considers infinite populations to ensure that such a model predicts the limit behavior of UMDA.

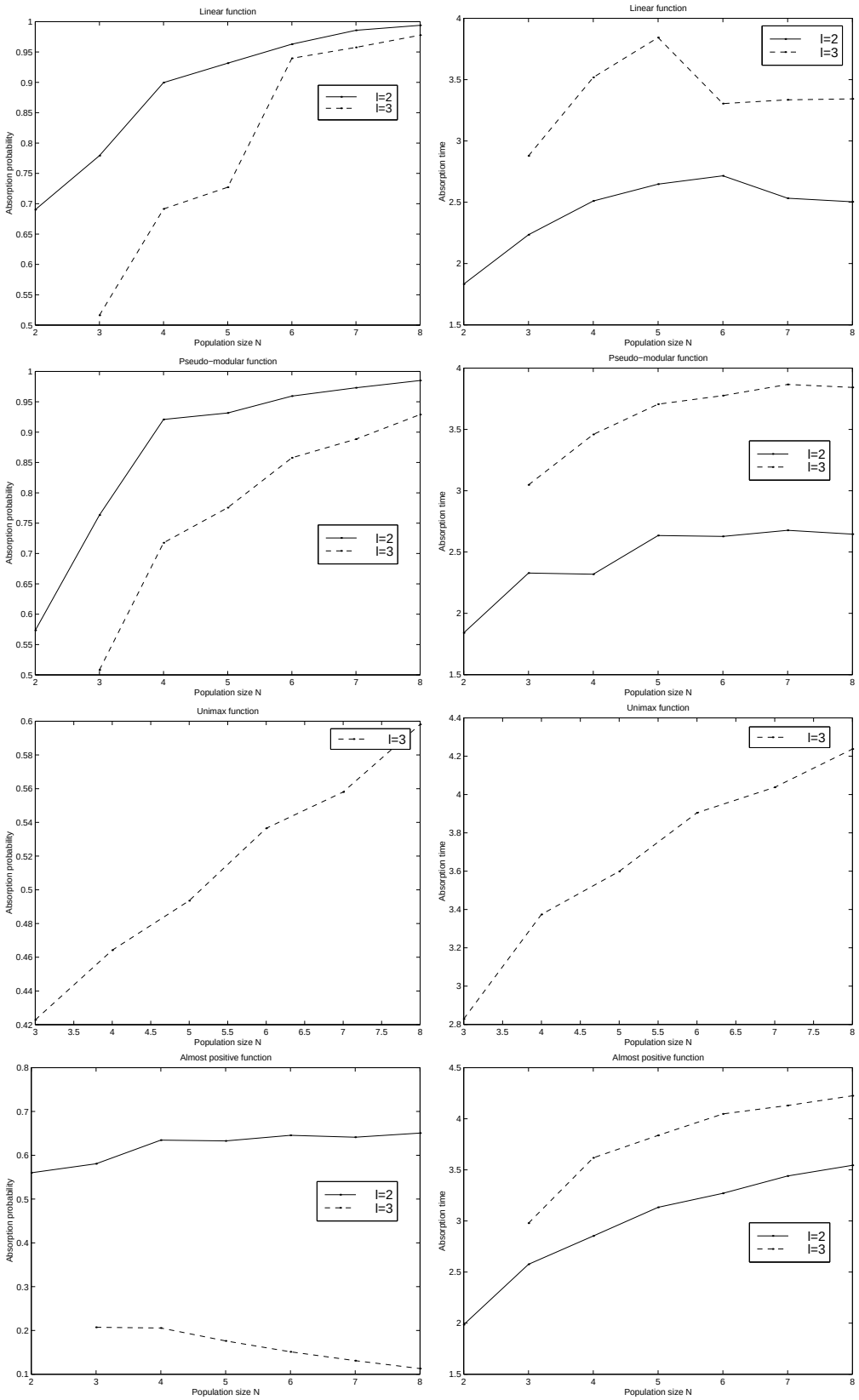


Figure 6.1. Absorption probability and absorption time for the linear, pseudo-modular, unimax and almost positive functions.

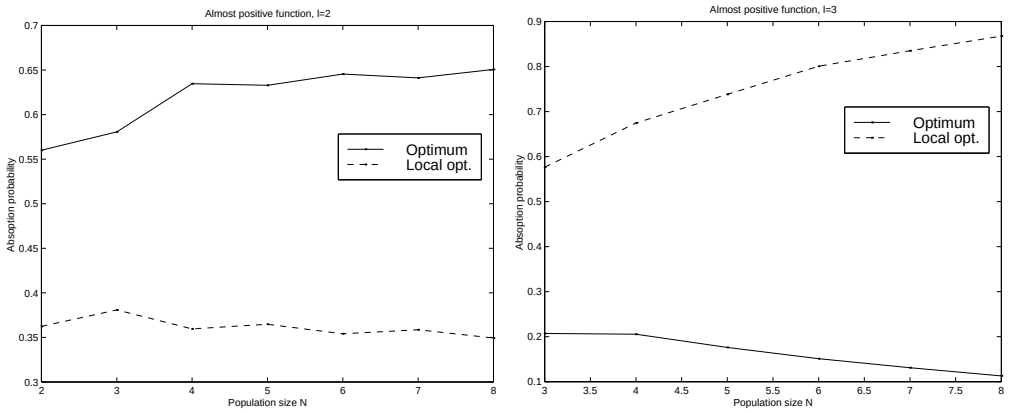


Figure 6.2. Comparison of absorption probability to the optimum and to the local optimum for $n = 2$ and $n = 3$ for the almost positive function.

6.2.1 A Dynamical System for UMDA

In this subsection we see that the dynamical system scheme used in Chapter 5 to model PBIL can also be valid (until a given moment) for the UMDA algorithm. Here it is important to write a pseudocode for UMDA in terms of probability vectors (see Figure 6.3, where $p_i^{(t)}$ is the probability of obtaining a 1 in the i^{th} component in the t^{th} generation).

UMDA

Obtain an initial vector $\mathbf{p}_0 = (p_1^{(0)}, p_2^{(0)}, \ldots, p_n^{(0)})$

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

$D_{t-1} \leftarrow$ Sample M individuals from $\mathbf{p}_{t-1}$

$D_{t-1}^{Se} \leftarrow$Select $N \leq M$ individuals from D_{t-1} according to proportional selection

$\mathbf{p}_t = \left(\frac{\sum_{j=1}^{N} \delta_j(X_1=1|D_{t-1}^{Se})}{N}, \frac{\sum_{j=1}^{N} \delta_j(X_2=1|D_{t-1}^{Se})}{N}, \ldots, \frac{\sum_{j=1}^{N} \delta_j(X_l=1|D_{t-1}^{Se})}{N} \right)$

$\leftarrow$ Find the new probability vector

Figure 6.3. Pseudocode for UMDA in terms of probability vectors.

Then the key problem is to associate a discrete dynamical system with UMDA, such that the trajectories followed by the probability vectors $\{\mathbf{p}_t\}_{t=0,1,2,\ldots}$ will be related to the iterations of that discrete dynamical system.

UMDA can be considered as a sequence of probability vectors, each one given by a stochastic transition rule τ:

$$\mathbf{p}_0 \xrightarrow{\tau} \mathbf{p}_1 \xrightarrow{\tau} \mathbf{p}_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} \mathbf{p}_{t-1} \xrightarrow{\tau} \mathbf{p}_t \cdots$$

i.e. $\mathbf{p}_t = \tau(\mathbf{p}_{t-1}) = \tau^t(\mathbf{p}_0)$. We are interested in the trajectories followed by the iterations of τ, and in particular in its limit behavior:

$$\lim_{t\to\infty} \tau^t(\mathbf{p}_0). \tag{6.10}$$

The stochastic nature of operator τ makes the study of limit (6.10) difficult. This is the reason why a new operator $\mathcal{G}$ is defined:

$$\mathcal{G} : [0,1]^n \longrightarrow [0,1]^n$$

such that $\mathcal{G}(\mathbf{p}) = \mathsf{E}[\tau(\mathbf{p})] = (\mathsf{E}[\tau_1(\mathbf{p})], \mathsf{E}[\tau_2(\mathbf{p})], \ldots, \mathsf{E}[\tau_n(\mathbf{p})])$. The operator $\mathcal{G}$ is a deterministic function that gives the expected value of the random operator τ. The iterations of $\mathcal{G}$ are defined as $\mathcal{G}^t(\mathbf{p}) = \mathcal{G}(\mathcal{G}^{t-1}(\mathbf{p}))$ with $\mathcal{G}_i(\mathbf{p}) = \mathsf{E}[\tau_i(\mathbf{p})]\ \forall\ i = 1, 2, \ldots, l$. The operator $\mathcal{G}$ can be thought as a discrete dynamical system:

$$\mathbf{p}, \mathcal{G}(\mathbf{p}), \ldots, \mathcal{G}^t(\mathbf{p}), \ldots$$

If we consider infinite populations after one step, the next probability vector converges in probability to its expectation, so it is natural to use $\mathcal{G}$ to define an infinite population model. Thus $\mathcal{G}$ defines a discrete dynamical system on Ω, given an initial probability vector $\mathbf{p}_0$. The trajectory of this probability vector is the sequence $\mathbf{p}_0$, $\mathcal{G}(\mathbf{p}_0)$, $\mathcal{G}^2(\mathbf{p}_0)$, $\mathcal{G}^3(\mathbf{p}_0)$, ...

6.2.2 An Expression for the Discrete Dynamical System Associated with UMDA

The aim of this section is to calculate explicitly an expression for $\mathcal{G}(\mathbf{p})$. In order to do this we have to use infinite populations. For this reason in the representation of each population we use proportions instead of numbers of individuals. Although starting from different principles, the expression will be the same as the one obtained by Mühlenbein and Mahnig (2000a) .

$$D = \{(d_1, d_2, \ldots, d_n) \mid d_i = \text{the proportion of the } i\text{-th individual in } D\}.$$

The calculation of each component of $\mathsf{E}[\tau(\mathbf{p})]$ for finite populations can be done using the formula:

$$\mathsf{E}[\tau_i(\mathbf{p})] = \sum_{\mathbf{x}\in\Omega} \sum_{D\in\mathcal{D}_\mathbf{x}} x_i \cdot \mathsf{P}(\text{obtain population } D \mid \mathbf{p}) \cdot \mathsf{P}(\text{select } \mathbf{x} \mid D)$$

with $\mathcal{D}_\mathbf{x} = \{D \mid \mathbf{x} \in D\}$. However, if we work with infinite populations, after drawing an infinite number of individuals from probability vector $\mathbf{p}$, a single population $D_\mathbf{p}$ is obtained. This population can be expressed as:

$$D_\mathbf{p} = (q_\mathbf{p}(\mathbf{x}_1), \ldots, q_\mathbf{p}(\mathbf{x}_m))$$

where $q_\mathbf{p}(\mathbf{x}) = \prod_{k=1}^n (p_k)^{x_k}(1-p_k)^{(1-x_k)}$ is the proportion of the individual $\mathbf{x}$ in population $D_\mathbf{p}$. Hence the i^{th} component of $\mathsf{E}[\tau(\mathbf{p})]$ can be written for infinite populations as:

$$\mathsf{E}[\tau_i(\mathbf{p})] = \sum_{\mathbf{x}\in\Omega} x_i \cdot \mathsf{P}(\text{ select } \mathbf{x} \mid D_\mathbf{p}).$$

In the case in which proportionate selection is taken into account the expression can be written as:

$$\begin{aligned}\mathsf{E}[\tau_i(\mathbf{p})] &= \sum_{\mathbf{x}\in\Omega} x_i \cdot \frac{q_{\mathbf{p}}(\mathbf{x})\cdot f(\mathbf{x})}{\sum_{\mathbf{y}\in\Omega} q_{\mathbf{p}}(\mathbf{y})f(\mathbf{y})} = \sum_{\mathbf{x}\in\Omega} x_i \cdot \frac{q_{\mathbf{p}}(\mathbf{x})\cdot f(\mathbf{x})}{\mathsf{E}_{\mathbf{p}}[f]} \\ &= \frac{p_i}{\mathsf{E}_{\mathbf{p}}[f]} \sum_{\substack{\mathbf{x}\in\Omega \\ x_i=1}} \frac{q_{\mathbf{p}}(\mathbf{x})}{p_i}\cdot f(\mathbf{x})\end{aligned}$$

where $\mathsf{E}_{\mathbf{p}}[f]$ denotes the expectation of f with respect to the probability distribution implied by probability vector $\mathbf{p}$. If we develop $q_{\mathbf{p}}(\mathbf{x})$, the dynamical system expression obtained is:

$$\begin{aligned}\mathcal{G}_i(\mathbf{p}) = \mathsf{E}[\tau_i(\mathbf{p})] &= \frac{p_i}{\mathsf{E}_{\mathbf{p}}[f]} \sum_{\substack{\mathbf{x}\in\Omega \\ x_i=1}} \frac{\prod_{k=1}^{n}(p_k)^{x_k}(1-p_k)^{(1-x_k)}}{p_i}\cdot f(\mathbf{x}) \\ &= \frac{p_i}{\mathsf{E}_{\mathbf{p}}[f]} \sum_{\substack{\mathbf{x}\in\Omega \\ x_i=1}} f(\mathbf{x}) \prod_{k\neq i}^{l}(p_k)^{x_k}(1-p_k)^{(1-x_k)}. \qquad (6.11)\end{aligned}$$

Expression (6.11) is the same as the one Mühlenbein and Mahnig (2002a) obtained from "linkage analysis". These authors also give another equivalent expression:

$$p_i^{(t+1)} = p_i^{(t)} + p_i^{(t)}(1-p_i^{(t)})\frac{\frac{\partial \mathsf{E}_{\mathbf{p}_t}[f]}{\partial p_i^{(t)}}}{\mathsf{E}_{\mathbf{p}_t}[f]}.$$

Using this last expression they state that the stable attractors of UMDA with infinite population and proportionate selection are the corners of Ω, which in the case of maximization are the local maxima of $f(\mathbf{x})$, and that in the interior of Ω there are only saddle points or local minimum.

Zhang (2004) also used discrete dynamical systems to model UMDA with infinite populations (in this case with 2-tournament selection), with the aim of studying the advantages of using higher-order statistics in EDAs. He compares the behavior of UMDA and FDA with infinite population and 2-tournament selection modeling both algorithms by means of discrete dynamical systems. The authors show that, in the case of a general objective function, the dynamical system associated with FDA has a better chance of obtaining a global optimum than that of UMDA. In the case of an additively decomposable objective function, they prove that the unique asymptotically stable fixed point of the dynamical system associated with FDA is the degenerate probability density function at the global optimal solution, which implies that FDA can converge to the global optimal solution. They also give an example to show that an additively decomposable objective function has some local optimal solutions where the UMDA can become trapped.

6.3 Summary

Each of the two previous sections of this chapter describes how the UMDA algorithm has been studied in this dissertation.

In the first section we used Markov chains to model and analyze some interesting questions about UMDA behavior on pseudo-boolean functions. For each analyzed function, we calculated the

absorption probabilities to the optimal point and the expected absorption times. This calculation enables us to know the effects that changes in the population size have on these two quantities. The analysis shows the behavior of the algorithm when the complexity of the function increases: the absorption probability decreases while the expected absorption time increases. Even in the almost positive function the absorption probability goes near to zero when N increases.

The second section is devoted to modeling UMDA with infinite population and proportionate selection using discrete dynamical systems, arriving at the same expression that Mühlenbein and Mahnig (2002a) obtained.

Chapter 7

Analysis of the UMDA$_c$ algorithm

This dissertation is primarily focused on the theoretical study of discrete EDAs, but we also want to deal with at least a continuous instance of EDAs. This chapter presents the only analysis made for a continuous EDA in this thesis: the theoretical study of the behavior of the Univariate Marginal Distribution Algorithm for continuous domains (UMDA$_c$) in dimension n. The election of the UMDA$_c$ was based on its simplicity: this algorithm does not take into account dependencies among the variables. Therefore it is assumed that the n-dimensional joint probability density factorizes as a product of n independent univariate marginal densities. Here the mathematical modeling was carried out using classical probability theory.

The analysis for the progressive behavior of UMDA$_c$ cannot be accomplished for an arbitrary test function because of mathematical difficulties, and therefore typical fitness functions are preferred for this analysis. These functions should represent certain characteristics of other fitness functions, and the analysis should be feasible for them. The sphere (7.1), and linear (7.2) functions are test functions widely used in the theoretical study of Evolution Strategies:

$$Q(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{7.1}$$

$$L(\mathbf{x}) = a_0 + \sum_{i=1}^{n} a_i x_i, \quad \text{with } a_i \in I\!R,\ i = 0, 1, \ldots, n. \tag{7.2}$$

Usually the quadratic function (7.1) is used to simulate the behavior of the algorithm near an optimum (intuitively, the shape of any function near an optimum is similar to the shape of the sphere function), while the linear function (7.2) serves when the algorithm is far from an optimum (the shape of any function far from an optimum is similar to the shape of the linear function).

The study of UMDA$_c$ carried out in this chapter consists of modeling the application of this algorithm (using tournament selection) to the minimization of $L(\mathbf{x})$, and $Q(\mathbf{x})$ functions. The objective is to know if the algorithm works as expected far from (or near to) an optimum. The study carried out for the linear function appeared for the first time in González et al. (2002c), and later was extended to quadratic functions in González et al. (2002b).

The remainder of this chapter is organized as follows: Section 7.1 explains in detail UMDA$_c$ with tournament selection. Section 7.2 is devoted to the mathematical modeling of the algorithm.

Section 7.3 analyzes the modeling of linear functions, while Section 7.4 analyzes the case of quadratic functions. Finally, we draw conclusions in Section 7.5.

7.1 The UMDA$_c$ Algorithm with Tournament Selection

We model a UMDA$_c$ that considers tournament selection, assuming an infinite number of tournaments. This section describes in detail how the UMDA$_c$ algorithm with tournament selection works.

The algorithm works as follows. At each step t, an n-dimensional random variable $\mathbf{X}^t$=(X_1^t,. . ., X_n^t) is maintained. In the literature related to UMDA$_c$ it is usual to assume that the joint probability distribution of $\mathbf{X}^t$ follows an n-dimensional normal distribution which is factorized by a product of n unidimensional and independent normal densities. This assumption will be made here. Therefore each component of $\mathbf{X}^t$ is distributed as a unidimensional normal, that is $X_i^t \rightsquigarrow \mathcal{N}(\mu_i^t, \sigma_i^t)$, where $f_{\mathcal{N}(\mu_i^t,\sigma_i^t)}(x_i) = \frac{1}{\sqrt{2\pi}\sigma_i^t} e^{-(x_i-\mu_i^t)^2/2(\sigma_i^t)^2}$ with $i = 1, \ldots, n$. In other words $f_{\mathcal{N}(\mu_i^t,\sigma_i^t)}(x_i)$ denotes the density function of a unidimensional normal with mean μ_i^t and standard deviation σ_i^t.

Drawing the above n-dimensional random variable, two individuals are obtained, and the better one is selected, i.e. a tournament selection is made. This process is repeated N times, obtaining the population of selected individuals. This population is used to obtain the mean and standard deviation vectors of the random variable $\mathbf{X}^{t+1}$. These parameters are estimated by using their corresponding maximum likelihood estimators. In this way the new unidimensional distributions at step $t+1$ are achieved. Figure 7.1 shows a pseudocode for this algorithm for the minimization of function $G(\mathbf{x})$.

```
Obtain randomly the parameters of a normal probability
distribution for each variable
while no convergence do
  begin
    for (j = 1; j ≤ N; j++)
      begin
        Drawing X^t obtain 2 individuals:
            x^t_{1,j} = (x^{1,t}_{1,j}, ..., x^{n,t}_{1,j})
            x^t_{2,j} = (x^{1,t}_{2,j}, ..., x^{n,t}_{2,j})
        Evaluate x^t_{1,j}, x^t_{2,j}
        Select the best one:
            x^t_{(1:2),j} = argmin_x{G(x^t_{1,j}), G(x^t_{2,j})}
      end
    for (i = 1; i ≤ n; i++)
      begin
        Estimate the parameters of the new density functions:
            μ_i^{t+1} = (Σ_{j=1}^N x^{i,t}_{(1:2),j}) / N

            σ_i^{t+1} = sqrt( (Σ_{j=1}^N (x^{i,t}_{(1:2),j} − μ_i^{t+1})^2) / N )
      end
  end
```

Figure 7.1. Pseudocode for UMDA$_c$ with tournament selection.

Our objective is to learn how the density functions change with time, which is achieved knowing how μ_i^t and σ_i^t evolve when t increases.

7.2 Mathematical Modeling

To model the UMDA$_c$ algorithm with tournament selection for continuous optimization problems with n variables, we take a case in which at each step an infinite number of tournaments is made. The mathematical model will depend on the function being optimized. At first we try to model the algorithm as generally as possible, hence we assume that the optimization problem we handle is:

$$\min_{\mathbf{x}\in I\!R^n} G(\mathbf{x}).$$

However, as we will see later, at some point it will be necessary to particularize the function we are analyzing.

As noted above, we assume that at each step t, each variable follows a unidimensional normal distribution, and has associated the following density function:

$$f_{X_i^t}(x_i) = f_{\mathcal{N}(\mu_i^t,\sigma_i^t)}(x_i) = \frac{1}{\sqrt{2\pi}\sigma_i^t} e^{-(x_i-\mu_i^t)^2/2(\sigma_i^t)^2},$$

where $f_{X_i^t}(x_i)$ denotes the density function of the random variable X_i^t. As we are working with UMDA$_c$ these variables are independent. Hence at each step t we have an n-dimensional random variable $\mathbf{X}^t = (X_1^t, \ldots, X_n^t)$ following the density $f_{\mathbf{X}^t}(\mathbf{x})$ with:

$$f_{\mathbf{X}^t}(\mathbf{x}) = \prod_{i=1}^{n} f_{\mathcal{N}(\mu_i^t,\sigma_i^t)}(x_i).$$

To simplify notation, each density function associated with each variable X_i^t will henceforth be denoted by:

$$f_i^t(x_i) = f_{X_i^t}(x_i) = f_{\mathcal{N}(\mu_i^t,\sigma_i^t)}(x_i), \text{ with } i = 1, 2 \ldots, n.$$

Likewise, its associated distribution function will be denoted by:

$$F_i^t(x_i) = \int_{-\infty}^{x_i} f_i^t(s)ds, \text{ with } i = 1, 2 \ldots, n.$$

We use the usual notation not only in the case of the standard normal density:

$$\phi(x) = f_{\mathcal{N}(0,1)}(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2},$$

but also in the case of its associated distribution function:

$$\Phi(x) = \int_{-\infty}^{x} \phi(s)ds.$$

At each step t the random variable $\mathbf{X}^t_{(1:2)} = \min\{X^t_1, X^t_2\}$ is considered, i.e. the random variable of the best of two variables $\mathbf{X}^t$. Thus:

$$\begin{aligned} \mathsf{E}[\mathbf{X}^t_{(1:2)}] &= \left(\mathsf{E}[X^t_{(1:2),1}], \ldots, \mathsf{E}[X^t_{(1:2),n}]\right) \\ \mathsf{Var}[\mathbf{X}^t_{(1:2)}] &= \left(\mathsf{Var}[X^t_{(1:2),1}], \ldots, \mathsf{Var}[X^t_{(1:2),n}]\right) \end{aligned}$$

after which, the new distributions at time $t+1$ are obtained. As we consider infinite populations, each X^{t+1}_i obeys $\mathcal{N}\left(\mu^{t+1}_i, \sigma^{t+1}_i\right)$, with $\mu^{t+1}_i = \mathsf{E}[X^t_{(1:2),i}]$ and $\sigma^{t+1}_i = \sqrt{\mathsf{Var}[X^t_{(1:2),i}]}$.

As we want to model the behavior of the algorithm, we need to know explicitly the expressions of μ^{t+1}_i and σ^{t+1}_i given μ^t_i, σ^t_i, for $i = 1, \ldots, n$. Then we can use these expressions to analyze the sequences $\{\mu^t_i\}_t$ and $\{\sigma^t_i\}_t$ with $t \in I\!N$, and to study how they evolve when the number of iterates increases. In other words, we study the limits:

$$\begin{aligned} &\lim_{t\to\infty} \mu^t_i, \ i = 1, \ldots, n \\ &\lim_{t\to\infty} \sigma^t_i, \ i = 1, \ldots, n. \end{aligned}$$

In order to calculate μ^{t+1}_i and σ^{t+1}_i, we have to obtain the density function associated with $\mathbf{X}^t_{(1:2)}$ the best individual of each tournament. We denote this density function by $f^t_{(1:2)}(x_1, \ldots, x_n)$. Notice that the previous density will depend on G, the objective function that we are considering.

7.2.1 Calculation of $f^t_{(1:2)}(x_1, \ldots, x_n)$

In order to calculate the density function $f^t_{(1:2)}(x_1, \ldots, x_n)$ we proceed as follows. First we obtain its associated distribution function, $F^t_{(1:2)}(x_1, \ldots, x_n)$. Then we derive this distribution function and obtain the density function $f^t_{(1:2)}(x_1, \ldots, x_n)$.

Let $\mathbf{X}^t_1 = (X^t_{1,1}, \ldots, X^t_{1,n})$ be the random variable associated with the first individual obtained in the tournament at step t, and $\mathbf{X}^t_2 = (X^t_{2,1}, \ldots, X^t_{2,n})$ the random variable corresponding to the second individual. Thus, the distribution function $F^t_{(1:2)}(x_1, \ldots, x_n)$ is:

$$F^t_{(1:2)}(x_1, \ldots, x_n) = \mathsf{P}\left((X^t_{(1:2),1}, \ldots, X^t_{(1:2),n}) \leq (x_1, \ldots, x_n)\right). \tag{7.3}$$

To make the calculus easier, we express the random variable associated with the best individual in each tournament as a sum of random variables:

$$\mathbf{X}^t_{(1:2)} = \mathbf{X}^t_1 \cdot I_{\{G(\mathbf{X}^t_1) \leq G(\mathbf{X}^t_2)\}} + \mathbf{X}^t_2 \cdot I_{\{G(\mathbf{X}^t_1) > G(\mathbf{X}^t_2)\}},$$

where the random variable I_A denotes the characteristic function of the event A. Hence:

$$I_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise.} \end{cases}$$

The event described in (7.3) is written as the following union of events:

$$(X^t_{(1:2),1}, \ldots, X^t_{(1:2),n}) \leq (x_1, \ldots, x_n) = \left\{U^t_1 \cap U^t_2\right\} \cup \left\{V^t_1 \cap V^t_2\right\},$$

where:

$$\begin{aligned} U_1^t &= \{(X_{1,1}^t, \ldots, X_{1,n}^t) \leq (x_1, \ldots, x_n)\} \\ U_2^t &= \{G(X_{1,1}^t, \ldots, X_{1,n}^t) \leq G(X_{2,1}^t, \ldots, X_{2,n}^t)\} \\ V_1^t &= \{(X_{2,1}^t, \ldots, X_{2,n}^t) \leq (x_1, \ldots, x_n)\} \\ V_2^t &= \{G(X_{1,1}^t, \ldots, X_{1,n}^t) > G(X_{2,1}^t, \ldots, X_{2,n}^t)\}. \end{aligned}$$

We denote by U^t the event $U_1^t \cap U_2^t$ and by V^t the event $V_1^t \cap V_2^t$, and since:

$$\{(X_{(1:2),1}^t, \ldots, X_{(1:2),n}^t) \leq (x_1, \ldots, x_n)\} = \{U^t\} \cup \{V^t\}.$$

Hence, given that we have disjoint events, we can state that:

$$F_{(1:2)}^t(x_1, \ldots, x_n) = \mathsf{P}(U^t) + \mathsf{P}(V^t).$$

Taking into account that $\mathsf{P}(U^t) = \mathsf{P}(V^t)$ (G is a continuous function), it is enough to obtain $\mathsf{P}(U^t)$. In order to do so we find the conditional probability $\mathsf{P}(U^t | X_{1,1}^t = x_{1,1}, \ldots, X_{1,n}^t = x_{1,n})$ and then we integrate over the rest of the variables:

$$\begin{aligned} &\mathsf{P}(U^t | X_{1,1}^t = x_{1,1}, \ldots, X_{1,n}^t = x_{1,n}) \\ &\quad = \begin{cases} \mathsf{P}\left(G(X_{2,1}^t, \ldots, X_{2,n}^t) \geq G(x_{1,1}, \ldots, x_{1,n})\right) & \text{if } x_{1,1} \leq x_1, \ldots, x_{1,n} \leq x_n \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

To simplify the notation we write:

$$\mathsf{P}\left(G(X_1^t, \ldots, X_n^t) \geq G(x_1, \ldots, x_n)\right) = A^t\left(G(\mathbf{x})\right)$$

Therefore:

$$\begin{aligned} &\mathsf{P}(U^t) \\ &= \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} \mathsf{P}(U^t | X_{1,1}^t = x_{1,1}, \ldots, X_{1,n}^t = x_{1,n}) f_1^t(x_{1,1}) \cdot \ldots \cdot f_n^t(x_{1,n}) dx_{1,1} \ldots dx_{1,n} \\ &= \int_{-\infty}^{x_1} \ldots \int_{-\infty}^{x_n} A^t\left(G(\mathbf{x}_1)\right) f_1^t(s_1) \cdot \ldots \cdot f_n^t(s_n) ds_1 \ldots ds_n. \end{aligned}$$

Hence, the distribution function of $\mathbf{X}_{(1:2)}^t$ is:

$$\begin{aligned} F_{(1:2)}^t(x_1, \ldots, x_n) &= \mathsf{P}\left((X_{(1:2),1}^t, \ldots, X_{(1:2),n}^t) \leq (x_1, \ldots, x_n)\right) \\ &= 2 \int_{-\infty}^{x_1} \ldots \int_{-\infty}^{x_n} A^t\left(G(\mathbf{x})\right) \cdot f_1^t(s_1) \cdot \ldots \cdot f_n^t(s_n) ds_1 \ldots ds_n. \end{aligned}$$

Deriving the above expression we obtain the density function as:

$$f_{(1:2)}^t(x_1, \ldots, x_n) = 2A^t\left(G(\mathbf{x})\right) \prod_{i=1}^{n} f_i^t(x_i).$$

7.2.2 Calculation of μ_i^{t+1} and σ_i^{t+1}

To obtain μ_i^{t+1} and σ_i^{t+1} we must first calculate each marginal density function $f^t_{(1:2),i}(x_i)$ with $i = 1, \ldots, n$:

$$\begin{aligned} \mu_i^{t+1} &= \int_{-\infty}^{\infty} x_i f^t_{(1:2),i}(x_i) dx_i \\ (\sigma_i^{t+1})^2 &= \int_{-\infty}^{\infty} x_i^2 f^t_{(1:2),i}(x_i) dx_i - (\mu_i^{t+1})^2. \end{aligned}$$

The marginal densities can be expressed as follows:

$$\begin{aligned} & f^t_{(1:2),i}(x_i) \\ & \quad = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} f^t_{(1:2)}(x_1, \ldots, x_n) dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n \\ & \quad = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} 2A^t\left(G(\mathbf{x})\right) \prod_{j=1}^{n} f_j^t(x_j) dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n \\ & \quad = 2 f_i^t(x_i) h_i^t(x_i), \end{aligned} \tag{7.4}$$

where:

$$h_i^t(x_i) = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} A^t\left(G(\mathbf{x})\right) \prod_{\substack{j=1 \\ j \neq i}}^{n} f_j^t(x_j) dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n.$$

As can be seen in (7.4) the calculations of μ_i^{t+1} and σ_i^{t+1} are closely related to the objective function G. For this reason it is necessary to particularize the objective function. Our analysis will now focus on the following two cases:

- The case of linear functions.
- The case of quadratic functions.

7.3 Linear Functions

We shall start by studying the simplest case, where the function $L(\mathbf{x})$ under consideration is:

$$\begin{array}{rccl} L: & I\!R^n & \longrightarrow & I\!R \\ & \mathbf{x} & \longmapsto & a_0 + \sum_{i=1}^n a_i x_i, \text{ for all } i = 1, \ldots, n. \end{array}$$

This will help us to see how the algorithm performs far from the optimum. An optimum (minimum) of this function does not exist in the usual mathematical terms. However, in the sense of a limit process the algorithm (in the case in which the algorithm performs properly) is expected to search for small values of $L(\mathbf{x})$ unboundedly, trying to find the optimum.

7.3.1 Calculation of μ_i^{t+1}

As the calculations are analogous for each component, they will only be given for the first component:

$$\mathsf{E}\left[X^t_{(1:2),1}\right] = \mu_1^{t+1} = \int_{-\infty}^{\infty} x_1 f^t_{(1:2),1}(x_1)dx_1 = \int_{-\infty}^{\infty} 2x_1 f_1^t(x_1)h_1^t(x_1)dx_1.$$

To simplify the notation, in the following calculations the superscript corresponding to the step is left out. It is, however, written in the final expression of μ_i^{t+1} and σ_i^{t+1}.

First we need to know the value of $A\left(L(\mathbf{x})\right)$:

$$A\left(L(\mathbf{x})\right) = A\left(a_0 + \sum_{j=1}^{n} a_j x_j\right) = \mathsf{P}\left(a_0 + \sum_{j=1}^{n} a_j X_j \geq a_0 + \sum_{j=1}^{n} a_j x_j\right).$$

Since each X_i is a random variable with density function $f_{\mathcal{N}(\mu_i,\sigma_i)}(x_i)$, we know that the random variable $T = \sum_{j=1}^{n} a_j X_j$ has density function:

$$\mathcal{N}\left(\sum_{j=1}^{n} a_j\mu_j, \sqrt{\sum_{j=1}^{n} a_j^2\sigma_j^2}\right).$$

Therefore:

$$A\left(L(\mathbf{x})\right) \quad = \quad \mathsf{P}\left(\frac{T - \sum_{j=1}^{n} a_j\mu_j}{\sqrt{\sum_{j=1}^{n} a_j^2\sigma_j^2}} \geq \frac{\sum_{j=1}^{n} a_j(x_j - \mu_j)}{\sqrt{\sum_{j=1}^{n} a_j^2\sigma_j^2}}\right) = 1 - \Phi\left(\frac{\sum_{j=1}^{n} a_j(x_j - \mu_j)}{\sqrt{\sum_{j=1}^{n} a_j^2\sigma_j^2}}\right).$$

Now we can calculate $h_1(x_1)$:

$$h_1(x_1) = \int_{-\infty}^{\infty} f_n(x_n)\ldots\int_{-\infty}^{\infty} f_k(x_k)\ldots\int_{-\infty}^{\infty} f_2(x_2)A\left(L(\mathbf{x})\right)dx_2\ldots dx_k\ldots dx_n$$

using the following notation:

$$g_k(x_1, x_{k+1}, x_{k+2}, \ldots, x_n) = \int_{-\infty}^{\infty} f_k(x_k)\ldots\int_{-\infty}^{\infty} f_2(x_2)A\left(L(\mathbf{x})\right)dx_2\ldots dx_k$$

with $k = 2, \ldots, n$, we know that:

$$h_1(x_1) = g_n(x_1).$$

We are going to prove by induction on k that:

$$g_k(x_1, x_{k+1}, x_{k+2}, \ldots, x_n) = 1 - \Phi\left(\frac{\sum\limits_{\substack{i=1\\ i\neq 2,\ldots,k}}^{n} a_i(x_i - \mu_i)}{\sqrt{\sum_{i=2}^{k} a_i^2\sigma_i^2 + \sum_{i=1}^{n} a_i^2\sigma_i^2}}\right). \tag{7.5}$$

First this is proved when $k = 2$, after which we use as inductive hypothesis case k and demonstrate that (7.5) is fulfilled in case $k + 1$.

Before proving equation (7.5), we must first take into account the following results borrowed from Beyer (2001), which will help us to make the calculations:

$$I_0(a,b) = \int_{-\infty}^{\infty} e^{-s^2/2}\Phi(as+b)ds = \sqrt{2\pi}\Phi\left(\frac{b}{\sqrt{1+a^2}}\right). \tag{7.6}$$

$$I_1(a,b) = \int_{-\infty}^{\infty} se^{-s^2/2}\Phi(as+b)ds = \frac{a}{\sqrt{1+a^2}}\exp\left(\frac{-1}{2}\frac{b^2}{1+a^2}\right). \tag{7.7}$$

Now we verify that (7.5) is satisfied when $k = 2$:

$$g_2(x_1, x_3, x_4, \ldots, x_n) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_2} e^{-(x_2-\mu_2)^2/2\sigma_2^2} \left(1 - \Phi\left(\frac{\sum_{i=1}^n a_i(x_i-\mu_i)}{\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}\right)\right) dx_2.$$

Making the transformation of variable $\frac{x_2-\mu_2}{\sigma_2} = s$ we find that:

$$g_2(x_1, x_3, x_4, \ldots, x_n) = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-s^2/2}\Phi\left(\frac{a_2\sigma_2 s + a_2\mu_2 + \sum_{\substack{i=1\\ i\neq 2}}^n a_i x_i - \sum_{i=1}^n a_i\mu_i}{\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}\right) ds$$

$$= 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-s^2/2}\Phi\left(\frac{a_2\sigma_2 s}{\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}} + \frac{\sum_{\substack{i=1\\ i\neq 2}}^n a_i(x_i-\mu_i)}{\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}\right) ds.$$

Taking into account the result (7.6):

$$g_2(x_1, x_3, x_4, \ldots, x_n) = 1 - \frac{1}{\sqrt{2\pi}} I_0\left(\frac{a_2\sigma_2}{\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}, \frac{\sum_{\substack{i=1\\ i\neq 2}}^n a_i(x_i-\mu_i)}{\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}\right)$$

$$= 1 - \Phi\left(\frac{\sum_{\substack{i=1\\ i\neq 2}}^n a_i(x_i-\mu_i)}{\sqrt{a_2^2\sigma_2^2 + \sum_{i=1}^n a_i^2\sigma_i^2}}\right).$$

By the inductive hypothesis, we assume that (7.5) is true for k. Let us now see whether it is true for $k+1$:

$$g_{k+1}(x_1, x_{k+2}, x_{k+3}, \ldots, x_n) = \int_{-\infty}^{\infty} f_{k+1}(x_{k+1}) g_k(x_1, x_{k+1}, x_{k+2}, \ldots, x_n) dx_{k+1}$$
$$= 1 - \int_{-\infty}^{\infty} f_{k+1}(x_{k+1}) \Phi \left(\frac{\sum_{\substack{i=1 \\ i \neq 2,\ldots,k}}^{n} a_i(x_i - \mu_i)}{\sqrt{\sum_{i=2}^{k} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}} \right) dx_{k+1}.$$

Taking into account the change of variable $\frac{x_{k+1} - \mu_{k+1}}{\sigma_{k+1}} = s$, we find that:

$$g_{k+1}(x_1, x_{k+2}, x_{k+3}, \ldots, x_n)$$
$$= 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-s^2/2} \Phi \left(\frac{a_{k+1}\sigma_{k+1} s + \sum_{\substack{i=1 \\ i \neq 2,\ldots,k+1}}^{n} a_i(x_i - \mu_i)}{\sqrt{\sum_{i=2}^{k} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}} \right) ds.$$

By again using the result (7.6), we find that:

$$g_{k+1}(x_1, x_{k+2}, x_{k+3}, \ldots, x_n)$$
$$= 1 - \frac{1}{\sqrt{2\pi}} I_0 \left(\frac{a_{k+1}\sigma_{k+1}}{\sqrt{\sum_{i=2}^{k} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}}, \frac{\sum_{\substack{i=1 \\ i \neq 2,\ldots,k+1}}^{n} a_i(x_i - \mu_i)}{\sqrt{\sum_{i=2}^{k} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}} \right).$$

Substituting the corresponding value of I_0 at this point:

$$g_{k+1}(x_1, x_{k+2}, x_{k+3}, \ldots, x_n)$$
$$= 1 - \Phi \left(\frac{\frac{\sum_{\substack{i=1 \\ i \neq 2,\ldots,k+1}}^{n} a_i(x_i - \mu_i)}{\sqrt{\sum_{i=2}^{k} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}}}{\sqrt{1 + \frac{a_{k+1}^2 \sigma_{k+1}^2}{\sum_{i=2}^{k} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}}} \right) = 1 - \Phi \left(\frac{\sum_{\substack{i=1 \\ i \neq 2,\ldots,k+1}}^{n} a_i(x_i - \mu_i)}{\sqrt{\sum_{i=2}^{k+1} a_i^2 \sigma_i^2 + \sum_{i=1}^{n} a_i^2 \sigma_i^2}} \right).$$

So we have proven that (7.5) is fulfilled, therefore:

$$h_1(x_1) = g_n(x_1) = 1 - \Phi \left(\frac{a_1(x_1 - \mu_1)}{\sqrt{a_1^2 \sigma_1^2 + 2\sum_{i=2}^{n} a_i^2 \sigma_i^2}} \right).$$

Now we can calculate μ_1^{t+1}:

$$\begin{aligned}\mu_1^{t+1} &= \int_{-\infty}^{\infty} 2x_1 f_1(x_1) h_1(x_1) dx_1 \\ &= \int_{-\infty}^{\infty} 2\frac{1}{\sqrt{2\pi}\sigma_1} x_1 e^{-(x_1-\mu_1)^2/2\sigma_1^2} \left(1 - \Phi\left(\frac{a_1(x_1-\mu_1)}{\sqrt{a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)\right) dx_1.\end{aligned}$$

Using the new variable $s = \frac{x_1-\mu_1}{\sigma_1}$, we obtain:

$$\begin{aligned}&\mu_1^{t+1} \\ &= \frac{2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} (\sigma_1 s + \mu_1) e^{-s^2/2} \left(1 - \Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)\right) ds \\ &= \frac{2}{\sqrt{2\pi}} \Bigg[\sigma_1 \int_{-\infty}^{\infty} s e^{-s^2/2} ds + \mu_1 \int_{-\infty}^{\infty} e^{-s^2/2} ds \\ &\quad -\sigma_1 \int_{-\infty}^{\infty} s e^{-s^2/2} \Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}}\right) ds \\ &\quad -\mu_1 \int_{-\infty}^{\infty} e^{-s^2/2} \Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}}\right) ds\Bigg].\end{aligned}$$

Equations (7.6) and (7.7) help us to express the above integrals as follows:

$$\begin{aligned}&\mu_1^{t+1} \\ &= \frac{2}{\sqrt{2\pi}} \Bigg[\sigma_1 \cdot 0 + \mu_1 \cdot \sqrt{2\pi} - \sigma_1 \cdot I_1\left(\frac{a_1\sigma_1}{\sqrt{a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}}\ ,\ 0\right) \\ &\quad -\mu_1 \cdot I_0\left(\frac{a_1\sigma_1}{\sqrt{a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}}\ ,\ 0\right)\Bigg].\end{aligned}$$

Using the results (7.6) and (7.7):

$$\begin{aligned}\mu_1^{t+1} &= \frac{2}{\sqrt{2\pi}} \left[\mu_1\sqrt{2\pi} - \sigma_1 \frac{a_1\sigma_1}{\sqrt{2a_1^2\sigma_1^2 + 2\sum_{i=2}^n a_i^2\sigma_i^2}} - \mu_1\frac{\sqrt{2\pi}}{2}\right] \\ &= \mu_1 - \frac{a_1\sigma_1^2}{\sqrt{\pi}\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}.\end{aligned}$$

Summarizing:

$$\mu_1^{t+1} = \mu_1^t - \frac{a_1(\sigma_1^t)^2}{\sqrt{\pi}\sqrt{\sum_{i=1}^n a_i^2(\sigma_i^t)^2}}.$$

The expression for the expectation in any component i is obtained analogously:

$$\mu_i^{t+1} = \mu_i^t - \frac{a_i(\sigma_i^t)^2}{\sqrt{\pi}\sqrt{\sum_{j=1}^n a_j^2(\sigma_j^t)^2}}. \tag{7.8}$$

7.3.2 Calculation of σ_i^{t+1}

As done before in calculating μ_i^{t+1}, we only make the calculations for $i = 1$, after which we generalize the result:

$$(\sigma_1^{t+1})^2 = \mathsf{Var}[X_{(1:2),1}^{t+1}] = \mathsf{E}\left[(X_{(1:2),1}^{t+1})^2\right] - \left(\mathsf{E}[X_{(1:2),1}^{t+1}]\right)^2.$$

We start by obtaining $\mathsf{E}\left[(X_{(1:2),1}^{t+1})^2\right]$

$$\begin{aligned}
&\mathsf{E}\left[(X_{(1:2),1}^{t+1})^2\right] = \int_{-\infty}^{\infty} 2x_1^2 f_1(x_1)h_1(x_1)dx_1 \\
&= \int_{-\infty}^{\infty} 2\frac{1}{\sqrt{2\pi}\sigma_1} x_1^2 e^{-(x_1-\mu_1)^2/2\sigma_1^2}\left(1-\Phi\left(\frac{a_1(x_1-\mu_1)}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)\right)dx_1.
\end{aligned}$$

Making the change of variable $\frac{x_1-\mu_1}{\sigma_1} = s$, we have:

$$\begin{aligned}
&\mathsf{E}\left[(X_{(1:2),1}^{t+1})^2\right] \\
&= \frac{2}{\sqrt{2\pi}}\int_{-\infty}^{\infty}(\sigma_1 s+\mu_1)^2 e^{-s^2/2}\left(1-\Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)\right)ds \\
&= \frac{2}{\sqrt{2\pi}}\left[\sigma_1^2\int_{-\infty}^{\infty}s^2e^{-s^2/2}ds + 2\sigma_1\mu_1\int_{-\infty}^{\infty}se^{-s^2/2}ds + \mu_1^2\int_{-\infty}^{\infty}e^{-s^2/2}ds\right. \\
&\quad -\sigma_1^2\int_{-\infty}^{\infty}s^2e^{-s^2/2}\Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)ds \\
&\quad -2\sigma_1\mu_1\int_{-\infty}^{\infty}se^{-s^2/2}\Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)ds \\
&\quad \left.-\mu_1^2\int_{-\infty}^{\infty}e^{-s^2/2}\Phi\left(\frac{a_1\sigma_1 s}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\right)ds\right].
\end{aligned} \tag{7.9}$$

Taking into account expressions (7.6), (7.7) and:

$$\begin{aligned}
I_2(a,b) &= \int_{-\infty}^{\infty}s^2e^{-s^2/2}\Phi(as+b)ds \\
&= \sqrt{2\pi}\Phi\left(\frac{b}{\sqrt{1+a^2}}\right) - \frac{a^2b}{(1+a^2)\sqrt{1+a^2}}\exp\left(\frac{-1}{2}\frac{b^2}{1+a^2}\right),
\end{aligned} \tag{7.10}$$

the integrals in (7.9) can be written as follows:

$$\begin{aligned}
&\mathsf{E}\left[(X_{(1:2),1}^{t+1})^2\right] \\
&= \frac{2}{\sqrt{2\pi}}\left[\sigma_1^2\cdot\sqrt{2\pi} + 2\sigma_1\mu_1\cdot 0 + \mu_1^2\cdot\sqrt{2\pi} - \sigma_1^2\cdot I_2\left(\frac{a_1\sigma_1}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\ ,\ 0\right)\right. \\
&\quad \left.-2\sigma_1\mu_1\cdot I_1\left(\frac{a_1\sigma_1}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\ ,\ 0\right) - \mu_1^2\cdot I_0\left(\frac{a_1\sigma_1}{\sqrt{a_1^2\sigma_1^2+2\sum_{i=2}^n a_i^2\sigma_i^2}}\ ,\ 0\right)\right].
\end{aligned}$$

Using (7.6), (7.7) and (7.10) we obtain:

$$\begin{aligned}
&\mathsf{E}\left[(X^{t+1}_{(1:2),1})^2\right] \\
&\quad = \frac{2}{\sqrt{2\pi}}\left[\sigma_1^2\sqrt{2\pi} + \mu_1^2\sqrt{2\pi} - \sigma_1^2\frac{\sqrt{2\pi}}{2} - 2\sigma_1\mu_1\frac{a_1\sigma_1}{\sqrt{2\sum_{i=1}^n a_i^2\sigma_i^2}} - \mu_1^2\frac{\sqrt{2\pi}}{2}\right] \\
&\quad = \sigma_1^2 + \mu_1^2 - \frac{2a_1\mu_1\sigma_1^2}{\sqrt{\pi}\sqrt{\sum_{i=1}^n a_i^2\sigma_i^2}}.
\end{aligned}$$

Now the superscripts corresponding to the iteration are written in order to obtain the full expression:

$$\mathsf{E}\left[(X^{t+1}_{(1:2),1})^2\right] = (\sigma_1^t)^2 + (\mu_1^t)^2 - \frac{2a_1\mu_1^t(\sigma_1^t)^2}{\sqrt{\pi}\sqrt{\sum_{i=1}^n a_i^2(\sigma_i^t)^2}}.$$

Finally:

$$\sigma_1^{t+1} = \sigma_1^t\sqrt{1 - \frac{a_1^2(\sigma_1^t)^2}{\pi\sum_{i=1}^n a_i^2(\sigma_i^t)^2}}.$$

An analogous expression for any component i can be given:

$$\sigma_i^{t+1} = \sigma_i^t\sqrt{1 - \frac{a_i^2(\sigma_i^t)^2}{\pi\sum_{j=1}^n a_j^2(\sigma_j^t)^2}}. \tag{7.11}$$

7.3.3 Analyzing the Algorithm's Behavior

Having obtained the expressions of μ_i^{t+1} and σ_i^{t+1}, we now try to predict the algorithm's behavior when t increases. This is done by analyzing each sequence of means $\{\mu_i^t\}_t$ and each sequence of standard deviations $\{\sigma_i^t\}_t$ with $t \in I\!N$.

To prove that the algorithm performs properly we must show that:

$$\begin{aligned}
&\text{when} \quad a_i > 0 \Rightarrow \mu_i^t \longrightarrow -\infty \text{ , as } t \to \infty \\
&\text{when} \quad a_i < 0 \Rightarrow \mu_i^t \longrightarrow +\infty \text{ , as } t \to \infty,
\end{aligned}$$

because if so, the algorithm would improve at each step, minimizing unboundedly the value of the objective function.

Unfortunately means sequences $\{\mu_i^t\}_t$ with $t \in I\!N$ are difficult to study when standard deviations σ_i^t are not equal in each component. However, we can state that the improvement at each step and in each component can be written as follows:

$$|\mu_i^{t+1} - \mu_i^t| = \left|\frac{a_i(\sigma_i^t)^2}{\sqrt{\pi}\sqrt{\sum_{j=1}^n a_j^2(\sigma_j^t)^2}}\right|.$$

Hence, given that sequences $\{\sigma_i^t\}_t$ decrease for all i (see equation (7.11), with $a_i > 0$), the improvement in each component decreases when t increases.

Given the difficulty of analyzing the sequences $\{\mu_i^t\}_t$ and $\{\sigma_i^t\}_t$ with $t \in I\!N$, we are going to study a particular case, in which the function to optimize is:

$$L_1(\mathbf{x}) = \sum_{i=1}^{n} x_i,$$

and the sequence of variances meets the condition:

$$\sigma_i^t = \sigma^t \text{ , with } i = 1, \ldots, n.$$

First of all we study the sequence of variances $\{\sigma^t\}_t$ with $t \in I\!N$. Given that:

$$\sigma^{t+1} = \sigma^t \left(\frac{n\pi - 1}{n\pi} \right)^{1/2},$$

we can write σ^{t+1} as a function of σ^0. Therefore, solving the recurrence, the sequence of standard deviations can be written as follows:

$$\sigma^{t+1} = \sigma^0 \left(\frac{n\pi - 1}{n\pi} \right)^{\frac{t+1}{2}}.$$

The above expression helps us to analyze the means sequence $\{\mu^t\}_t$ with $t \in I\!N$. After substituting this expression in equation (7.8) we obtain:

$$\mu^t = \mu^{t-1} - \frac{\sigma^0}{\sqrt{n\pi}} \left(\frac{n\pi - 1}{n\pi} \right)^{\frac{t-1}{2}}.$$

We can also express μ^t in terms of μ^0 and σ^0:

$$\mu^t = \mu^0 - \frac{\sigma^0}{\sqrt{n\pi}} \left(1 + \left(\frac{n\pi - 1}{n\pi} \right)^{1/2} + \ldots + \left(\frac{n\pi - 1}{n\pi} \right)^{\frac{t-1}{2}} \right)$$

which yields:

$$\mu^t = \mu^0 - \frac{\sigma^0}{\sqrt{n\pi}} \cdot \frac{\left(\frac{n\pi-1}{n\pi}\right)^{t/2} - 1}{\left(\frac{n\pi-1}{n\pi}\right)^{1/2} - 1}.$$

This new form of writing μ^t makes it easier to analyze the means sequence. This sequence decreases and has a finite limit.

$$\begin{aligned} \lim_{t\to\infty} \mu^t &= \lim_{t\to\infty} \left(\mu^0 - \frac{\sigma^0}{\sqrt{n\pi}} \cdot \frac{\left(\frac{n\pi-1}{n\pi}\right)^{\frac{t}{2}} - 1}{\left(\frac{n\pi-1}{n\pi}\right)^{1/2} - 1} \right) \\ &= \mu^0 + \frac{\sigma^0}{\sqrt{n\pi}} \frac{1}{\left(\frac{n\pi-1}{n\pi}\right)^{1/2} - 1} \\ &= \mu^0 + \frac{\sigma^0}{\sqrt{n\pi - 1} - \sqrt{n\pi}}. \end{aligned} \tag{7.12}$$

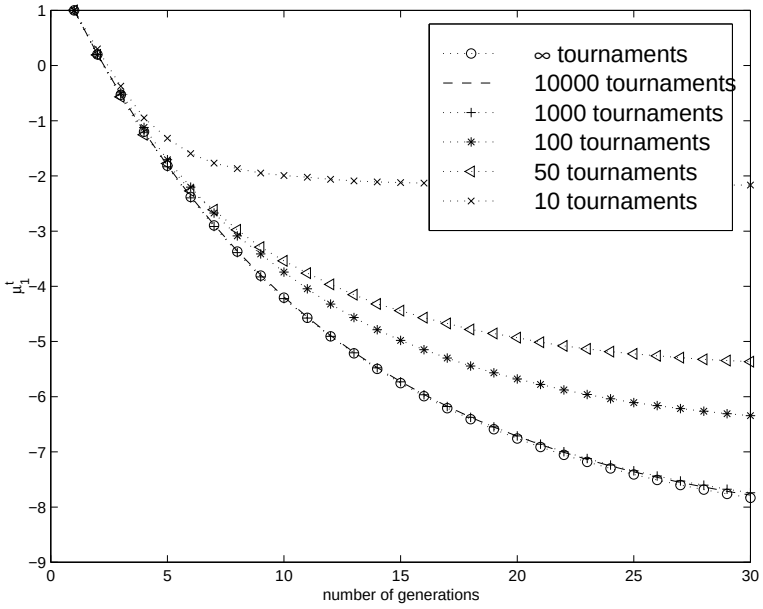


Figure 7.2. Values of μ_1^t for different numbers of tournaments.

Therefore, although the mean values decrease at each step, this decrease is not unbounded. This fact implies poor algorithm performance, leading us to conclude that this algorithm does not work as expected when we are far from the optimum and the number of tournaments at each step is infinite.

To see how the algorithm performs with a finite number of tournaments, we carried out a number of experiments. Having chosen the number of tournaments, we ran the algorithm with the linear function:

$$L_1(\mathbf{x}) = \sum_{i=1}^{2} x_i, \text{ with } \mathbf{x} = (x_1, x_2) \in I\!R^2.$$

The initial density functions used were:

$$f_{X_i^0}(x_i) = f_{\mathcal{N}(1,2)}(x_i), \text{ with } i = 1, 2.$$

We ran the algorithm 50 times for a number of tournaments $(10, 50, 100, 1000, 10000)$. After which we calculated the average value of the mean at each generation. Here we only show the results for the mean values in the first component (the values for the second component are analogous). The results for these finite numbers of tournaments, and also for an infinite number of tournaments – theoretical results obtained in (7.12) – can be seen in Figure 7.2.

The experiments show that a low number of tournaments does not guarantee an unbounded decrease in the mean values. In fact, as can be seen in Figure 7.2, the mean values block for a low number of generations when the number of tournaments is small.

The experiment shows that the smaller the number of tournaments, the worse the algorithm performs.

7.4 Quadratic Function

This section deals with our analysis of the case in which the function considered is

$$\begin{array}{rccc} Q: & I\!R^n & \longrightarrow & I\!R \\ & \mathbf{x} & \longmapsto & \sum_{i=1}^{n} x_i^2. \end{array}$$

This function is used in the literature to study the algorithm's behavior near the optimum.

We attempted to make a similar analysis to the one made in the linear case. However, during the study of this function, problems arose in calculating some integrals. These problems forced us to make certain simplifications in order to obtain as much information as possible concerning the algorithm's behavior. The simplifications consist of assuming that each X_i is distributed as a normal centered at the optimum of $Q(\mathbf{x})$, and that all those variables have the same deviation σ. Such a situation describes what happens when the algorithm is very near the optimum. This fact enables us to study if the optimum is finally reached or not. In the event that it is reached, it is possible to analyze how the speed of convergence is affected as the space dimension increases.

7.4.1 Calculation of μ_i^{t+1}

As in the previous case we make the calculation for the first component:

$$\mu_1^{t+1} = \int_{-\infty}^{\infty} 2x_1 f_1(x_1) h_1(x_1) dx_1.$$

First we need to know the value of $A\left(Q(\mathbf{x})\right)$:

$$A\left(Q(\mathbf{x})\right) = A\left(\sum_{j=1}^{n} x_j^2\right) = \mathsf{P}\left(\sum_{j=1}^{n} (X_j)^2 \geq \sum_{j=1}^{n} x_j^2\right).$$

Since each X_i is a random variable with density function $f_{\mathcal{N}(\mu_i,\sigma_i)}(x_i)$, we know that:

$$\begin{aligned} A\left(Q(\mathbf{x})\right) &= A\left(\sum_{j=1}^{n} x_j^2\right) \\ &= \int \cdots \int_{\mathcal{D}} \left(\frac{1}{\sqrt{2\pi}}\right)^n \frac{1}{\sigma_1 \ldots \sigma_n} e^{-(u_1-\mu_1)^2/2\sigma_1^2} \cdot \ldots \cdot e^{-(u_n-\mu_n)^2/2\sigma_n^2} du_1 \ldots du_n. \end{aligned}$$

where $\mathcal{D} = \{u_1^2 + \ldots + u_n^2 \geq x_1^2 + \ldots + x_n^2\}$.

Here we encounter the first problem: to solve the above integral. To do so we make the following simplification: assuming that each X_i is a random variable distributed as a normal with mean $\mu_i = 0$ and deviation $\sigma_i = \sigma$, in other words with density function $f_{\mathcal{N}(0,\sigma)}(x_i)$. This is the only case where we find the above integrals solvable.

Using this kind of density functions implies studying the algorithm's behavior when the density functions are centered at the optimum, so we are really very near the optimum. But here we wonder: is the optimum actually reached? If the answer is yes, what is the speed of convergence as the dimension of the problem increases?

In order to answer these questions we carry out the following analysis:

Step 1.- To ensure that at each step we do not move away from the optimum, we have to demonstrate that:

$$\mu_i^0 = 0 \;\Rightarrow\; \mu_i^t = 0 \;\forall t$$

Step 2.- To prove that the optimum is reached we have to see whether:

$$\sigma^t \to 0 \text{ as } t \to \infty$$

Step 3.- We study the speed of convergence as dimension increases. This study allows us to compare the difficulty of approaching the optimum as the dimension increases.

7.4.2 Calculation of μ^{t+1} and σ^{t+1}

First of all, as in the linear case, in order to calculate μ^{t+1} and σ^{t+1} we need to find the expression of $A^t\left(Q(\mathbf{x})\right)$:

$$A^t\left(Q(\mathbf{x})\right) = A^t\left(\sum_{j=1}^{n} x_j^2\right) = \int \ldots \int_{\mathcal{D}} \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-(u_1^2+\ldots+u_n^2)/2\sigma^2} du_1 \ldots du_n$$

where $\mathcal{D} = \{u_1^2 + \ldots + u_n^2 \geq x_1^2 + \ldots + x_n^2\}$. Taking into account the change of variable $\frac{u_i}{\sigma} = t_i$, with $i = 1, \ldots, n$, we obtain:

$$A^t\left(Q(\mathbf{x})\right) = \left(\frac{1}{\sqrt{2\pi}}\right)^n \int \ldots \int_{\mathcal{D}^*} e^{-(t_1^2+\ldots+t_n^2)/2} dt_1 \ldots dt_n$$

where $\mathcal{D}^* = \{t_1^2 + \ldots + t_n^2 \geq Q(\mathbf{x})/\sigma^2\}$. This integral can be solved thanks to the generalization to n dimensions of the spherical change of variable in dimension three. This change can be seen in detail in Appendix A. Here we give the essential information:

- The variables $(t_1, \ldots, t_n)$ are changed to the variables $(\rho, \alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_2, \alpha_1)$.

- The range of variation of each new variable is:

$$\begin{aligned} 0 \leq \rho \leq \infty & \\ -\pi/2 \leq \alpha_j \leq \pi/2, & \quad \text{for } j = 2, \ldots, n-1 \\ 0 \leq \alpha_1 \leq 2\pi. & \end{aligned}$$

- The Jacobian of the transformation is :

$$|J_n| = \rho^{n-1} \cdot \cos\alpha_2 \cdot \cos^2\alpha_3 \cdot \ldots \cdot \cos^{n-3}\alpha_{n-2} \cdot \cos^{n-2}\alpha_{n-1}.$$

Using this change the integral is modified to:

$$
\begin{aligned}
& A^t\left(Q(\mathbf{x})\right) \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^n \int_0^{2\pi} d\alpha_1 \int_{-\pi/2}^{\pi/2} \cos\alpha_2 d\alpha_2 \int_{-\pi/2}^{\pi/2} \cos^2\alpha_3 d\alpha_3 \cdot \ldots \\
&\quad \ldots \cdot \int_{-\pi/2}^{\pi/2} \cos^{n-3}\alpha_{n-2} d\alpha_{n-2} \int_{-\pi/2}^{\pi/2} \cos^{n-2}\alpha_{n-1} d\alpha_{n-1} \int_{\sqrt{Q(\mathbf{x})}/\sigma}^{\infty} \rho^{n-1} e^{-\rho^2/2} d\rho \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^n \underbrace{2\pi \prod_{i=1}^{n-2} \int_{-\pi/2}^{\pi/2} \cos^i\beta d\beta}_{S(n)} \int_{\sqrt{Q(\mathbf{x})}/\sigma}^{\infty} \rho^{n-1} e^{-\rho^2/2} d\rho,
\end{aligned}
$$

where $S(n) = \frac{2\pi(\sqrt{\pi})^{n-2}}{\Gamma\left(\frac{n}{2}\right)}$ is the constant associated with the spherical change of variable in n dimensions (see Appendix A). Therefore, substituting the value of $S(n)$ above:

$$
A^t\left(Q(\mathbf{x})\right) = \left(\frac{1}{\sqrt{2\pi}}\right)^n \frac{2\pi(\sqrt{\pi})^{n-2}}{\Gamma\left(\frac{n}{2}\right)} \int_{\sqrt{Q(\mathbf{x})}/\sigma}^{\infty} \rho^{n-1} e^{-\rho^2/2} d\rho.
$$

Let I_n denote the indefinite integral $\int \rho^{n-1} e^{-\rho^2/2} d\rho$, and $I_n(u,v)$ denote definite integral $I_n\big|_u^v$. The integral $I_n(\sqrt{Q(\mathbf{x})}/\sigma, \infty)$ has different values when n is odd or even. When n is odd $I_1(\sqrt{Q(\mathbf{x})}/\sigma, \infty)$ is an incomplete Gamma function (it has no explicit expression), meaning that from here we only work with even dimension. We emphasize this fact writing $A^t_{2n}\left(Q(\mathbf{x})\right)$. Therefore:

$$
\begin{aligned}
A^t_{2n}\left(Q(\mathbf{x})\right) &= \left(\frac{1}{\sqrt{2\pi}}\right)^{2n} \frac{2\pi(\sqrt{\pi})^{2n-2}}{(n-1)!} \int_{\sqrt{Q(\mathbf{x})}/\sigma}^{\infty} \rho^{2n-1} e^{-\rho^2/2} d\rho \\
&= \frac{1}{2^{n-1}(n-1)!} \cdot I_{2n}(\sqrt{Q(\mathbf{x})}/\sigma, \infty).
\end{aligned}
$$

In order to solve integral I_{2n} we write:

$$
\begin{aligned}
u = \rho^{2n-2} &\quad \Rightarrow \quad du = (2n-2)\rho^{2n-3} d\rho \\
dv = \rho e^{-\rho^2/2} d\rho &\quad \Rightarrow \quad v = -e^{-\rho^2/2}
\end{aligned}
$$

so that:

$$
\begin{aligned}
I_{2n} &= -\rho^{2n-2} e^{-\rho^2/2} + (2n-2) \int \rho^{2n-3} e^{-\rho^2/2} d\rho \\
&= -\rho^{2n-2} e^{-\rho^2/2} + (2n-2) \cdot I_{2n-2}.
\end{aligned}
$$

Substituting the expressions of I_{2n-j}, with $j = 2, \ldots, 2n-2$:

$$
\begin{aligned}
& I_{2n} \\
&= -\rho^{2n-2} e^{-\rho^2/2} - (2n-2)\rho^{2n-4} e^{-\rho^2/2} - (2n-2)(2n-4)\rho^{2n-6} e^{-\rho^2/2} \\
&\quad - \ldots - (2n-2)(2n-4) \cdot \ldots \cdot 2 \cdot 1 e^{-\rho^2/2} \\
&= -e^{-\rho^2/2} \left[\rho^{2n-2} + \sum_{j=2}^{n} \rho^{2n-2j} (2n-2)(2n-4) \cdot \ldots \cdot (2n-2(j-1)) \right] \\
&= -e^{-\rho^2/2} \left[\rho^{2n-2} + \sum_{j=2}^{n} \rho^{2(n-j)} \cdot 2^{j-1} \cdot \frac{n!}{n(n-j)!} \right]. \qquad (7.13)
\end{aligned}
$$

Taking into account equation (7.13), integral $A_{2n}^{t}\left(Q(\mathbf{x})\right)$ can be expressed as follows:

$$\begin{aligned}
&A_{2n}^{t}\left(Q(\mathbf{x})\right)\\
&= \frac{1}{2^{n-1}(n-1)!}\cdot\left[-e^{-\rho^2/2}\left(\rho^{2n-2}+\sum_{j=2}^{n}\rho^{2(n-j)}\cdot 2^{j-1}\cdot\frac{n!}{n(n-j)!}\right)\right]\Bigg|_{\sqrt{Q(\mathbf{x})}/\sigma}^{\infty}\\
&= \frac{2^{1-n}}{(n-1)!}\cdot e^{-Q(\mathbf{x})/2\sigma^2}\left[\left(\frac{Q(\mathbf{x})}{\sigma^2}\right)^{n-1}+\sum_{j=2}^{n}\left(\frac{Q(\mathbf{x})}{\sigma^2}\right)^{n-j}\cdot 2^{j-1}\cdot\frac{n!}{n(n-j)!}\right]\\
&= \frac{2^{1-n}}{(n-1)!}\cdot\frac{e^{-Q(\mathbf{x})/2\sigma^2}}{\sigma^{2(n-1)}}\Bigg[\left(Q(\mathbf{x})\right)^{n-1}\\
&+ \sum_{j=2}^{n}\left(Q(\mathbf{x})\right)^{n-j}\cdot\sigma^{2(j-1)}\cdot 2^{j-1}\cdot\frac{n!}{n(n-j)!}\Bigg].
\end{aligned}$$

The next step is to calculate integral $h_1(x_1)$:

$$\begin{aligned}
&h_1(x_1)\\
&= \int\cdots\int_{I\!R^{2n-1}}\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{2n-1}e^{-(x_2^2+\ldots+x_{2n}^2)/2\sigma^2}A_{2n}^{t}\left(Q(\mathbf{x})\right)dx_2\ldots dx_{2n}\\
&= \int\cdots\int_{I\!R^{2n-1}}\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{2n-1}e^{-(x_2^2+\ldots+x_{2n}^2)/2\sigma^2}\cdot\\
&\quad\cdot\frac{2^{1-n}}{(n-1)!}\cdot\frac{e^{-(x_1^2+x_2^2+\ldots+x_{2n}^2)/2\sigma^2}}{\sigma^{2(n-1)}}\cdot\\
&\quad\cdot\left[\left(Q(\mathbf{x})\right)^{n-1}+\sum_{j=2}^{n}\left(Q(\mathbf{x})\right)^{n-j}\cdot\sigma^{2(j-1)}\cdot 2^{j-1}\cdot\frac{n!}{n(n-j)!}\right]dx_2\ldots dx_{2n}.
\end{aligned}$$

Changing the variable, $\frac{x_i}{\sigma}=t_i$, and taking into account that $\left(Q(\mathbf{x})\right)^k=\left(x_1^2+\sigma^2(t_2^2+\ldots+t_{2n}^2)\right)^k$, the expression of $h_1(x_1)$ can be written as:

$$\begin{aligned}
&h_1(x_1)\\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^{2n-1}\cdot\frac{2^{1-n}}{\sigma^{2(n-1)}(n-1)!}\int\cdots\int_{I\!R^{2n-1}}e^{-(t_2^2+\ldots+t_{2n}^2)}\cdot e^{-x_1^2/2\sigma^2}\cdot\\
&\quad\cdot\Bigg[\left((x_1^2+\sigma^2(t_2^2+\ldots+t_{2n}^2)\right)^{n-1}\\
&\quad+\sum_{j=2}^{n}\left((x_1^2+\sigma^2(t_2^2+\ldots+t_{2n}^2)\right)^{n-j}\cdot\sigma^{2(j-1)}\cdot 2^{j-1}\cdot\frac{n!}{n(n-j!)}\Bigg]dt_2\ldots dt_{2n}.
\end{aligned}$$

Making again the generalization to $2n-1$ dimensions of the spherical change of variable in dimension three:

$$
\begin{aligned}
& h_1(x_1) \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^{2n-1} \cdot \frac{2^{1-n}}{\sigma^{2(n-1)}(n-1)!} \frac{2\pi(\sqrt{\pi})^{2n-3}}{\Gamma\left(\frac{2n-1}{2}\right)} \cdot \\
&\quad \cdot \int_0^\infty \rho^{2n-2} e^{-\rho^2} e^{-x_1^2/2\sigma^2} \left[\left(x_1^2 + \sigma^2\rho^2\right)^{n-1} \right. \\
&\quad \left. + \sum_{j=2}^{n} \left(x_1^2 + \sigma^2\rho^2\right)^{n-j} \cdot \sigma^{2(j-1)} \cdot 2^{j-1} \cdot \frac{n!}{n(n-j)!} \right] d\rho \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^{2n-1} \cdot \frac{2^{1-n}}{\sigma^{2(n-1)}(n-1)!} \frac{2\pi(\sqrt{\pi})^{2n-3}}{\Gamma\left(\frac{2n-1}{2}\right)} \cdot \\
&\quad \cdot e^{-x_1^2/2\sigma^2} \left[\int_0^\infty \rho^{2n-2} e^{-\rho^2} \left(x_1^2 + \sigma^2\rho^2\right)^{n-1} \right. \\
&\quad \left. + \sum_{j=2}^{n} \sigma^{2(j-1)} \cdot 2^{j-1} \cdot \frac{n!}{n(n-j)!} \int_0^\infty \rho^{2n-2} e^{-\rho^2} \left(x_1^2 + \sigma^2\rho^2\right)^{n-j} d\rho \right]
\end{aligned}
$$

Unfortunately to compute integrals $\int_0^\infty \rho^{2n-2} e^{-\rho^2} \left(x_1^2 + \sigma^2\rho^2\right)^{n-j} d\rho$, with $j = 1, \ldots, 2n$ is not an easy task, hence to find an explicit general expression for $h_1(x_1)$ is difficult. Therefore we have solved this integral for some finite cases (by parts). This allows us to gain a general idea of the algorithm's performance as the problem dimension increases.

7.4.3 Analyzing the Algorithm's Behavior

As explained above we solve $h_1(x_1)$ in some finite cases. Although our analysis will not be so general as in the linear case, we do obtain some information about the algorithm's behavior near the optimum.

The finite cases are:

$$
\begin{aligned}
2n &= 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 40, \\
&\quad 50, 60, 70, 80, 90, 100, 150, 200, 300, 400, 500, 600.
\end{aligned}
$$

After obtaining the values for $h_1(x_1)$, for the above cases, we substitute them in the expression of μ^{t+1}, obtaining $\mu^{t+1} = 0$ for every t. This indicates that for the studied dimensions, the algorithm remains near the optimum. The results of substituting these values in the expression of σ^{t+1} are summarized in Table 7.1.

As can be seen in Table 7.1, for these finite cases we can write $\sigma^{t+1} = a_{2n}\sigma^t$. The factor of decrease a_{2n} is represented in Figure 7.3.

Having the $\{a_{2n}\}$ data sequence we need now to find a formula that approximates it, in other words, to fit a curve through the points in $\{a_{2n}\}$ sequence. This allows us to estimate the speed of convergence. We find the following least squares fit to data:

$$g(n) = 1 - \frac{0.4}{\sqrt{n}}.$$

Figure 7.4 shows that $g(n)$ fit properly the $\{a_{2n}\}$ data.

$2n$	σ^{t+1}	$2n$	σ^{t+1}
2	$.7071\sigma^t$	30	$.9249\sigma^t$
4	$.7906\sigma^t$	40	$.9352\sigma^t$
6	$.8291\sigma^t$	50	$.9422\sigma^t$
8	$.8524\sigma^t$	60	$.9473\sigma^t$
10	$.8683\sigma^t$	70	$.9513\sigma^t$
12	$.8800\sigma^t$	80	$.9545\sigma^t$
14	$.8891\sigma^t$	90	$.9571\sigma^t$
16	$.8964\sigma^t$	100	$.9594\sigma^t$
18	$.9025\sigma^t$	150	$.9669\sigma^t$
20	$.9076\sigma^t$	200	$.9714\sigma^t$
22	$.9120\sigma^t$	300	$.9767\sigma^t$
24	$.9159\sigma^t$	400	$.9799\sigma^t$
26	$.9192\sigma^t$	500	$.9820\sigma^t$
28	$.9223\sigma^t$	600	$.9836\sigma^t$

Table 7.1. Values of σ^{t+1} for some finite cases.

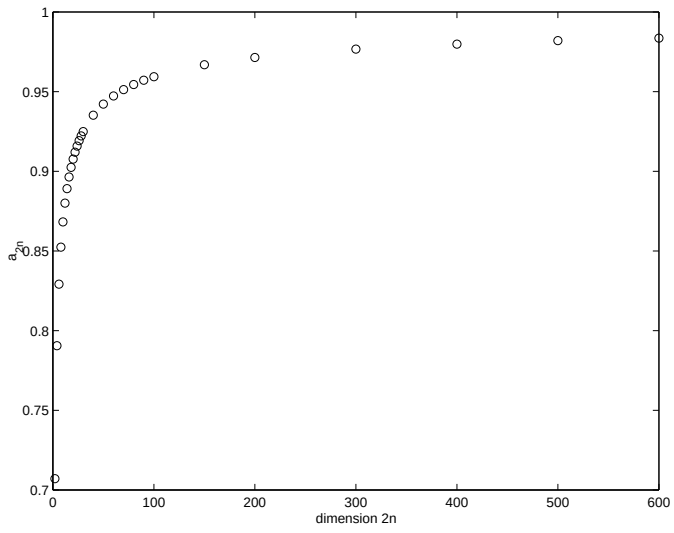


Figure 7.3. Factor of decrease of σ^{t+1}.

The results indicate that:

1 The value of $\sigma^t \to 0$ as $t \to \infty$ in the analyzed dimensions, therefore the algorithm reaches the optimum.

2 Since $g(n)$ seems to fit properly the points in $\{a_{2n}\}$ data, the speed of convergence decreases with the dimension as $O\left(\frac{1}{\sqrt{n}}\right)$.

Therefore we can conclude that in the finite cases studied, the algorithm reaches the optimum, but the speed of convergence decreases as the dimension of the problem increases.

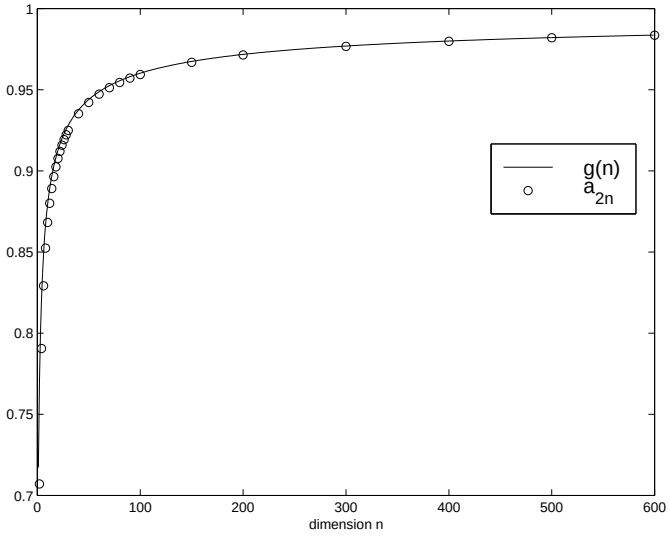


Figure 7.4. Fitting $\{a_{2n}\}$ values.

7.5 Summary

We have modeled the UMDA$_c$ algorithm with tournament selection applied to linear and quadratic functions when an infinite number of tournaments is performed. Linear functions are used to model the algorithm when far from the optimum, while quadratic function is used to analyze the algorithm when near the optimum.

Based on this modeling we have analyzed its behavior in n-dimensional linear functions and in an n-dimensional quadratic function. In the case of linear functions we conclude that the algorithm does not work as expected in linear function $L_1(\mathbf{x}) = \sum_{i=1}^{n} x_i$, with $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. After making certain assumptions in the case of quadratic function $Q(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$, we have proved for some finite dimensions that the algorithm reaches the optimum. Moreover, the speed of convergence gets slower as dimension increases.

The results obtained are closely related to the chosen distributions (unidimensional normals). It will be helpful to study the behavior of the algorithm when different distributions are used.

Part III

Results concerning Time Complexity

Chapter 8

Results concerning Time Complexity

The third part of this dissertation deals with the analysis of time complexity of EDAs. Time complexity studies the expected waiting time until a global optimum is encountered for the first time in terms of individual size. This is an important measure of performance for any optimization algorithm.

The chapter is organized in four sections. Section 8.1 introduces the chapter and also reviews previous work on time complexity of EAs.

Section 8.2 analyzes the time complexity of EDAs and offers a result concerning worst-case first hitting time for EDAs based on Markov chains: the highest expected first hitting time for UMDA, MIMIC, TREE and EBNA$_{BIC}$, is exponential in individual size, when they are used to maximize any pseudo-boolean injective function. Finally, we discuss how this result influences the calculus of bounds of average-case hitting times for EDAs. The analysis for UMDA algorithm appears for the first time in González et al. (2004).

Section 8.3 presents a study based on empirical results of the average first hitting time of EDAs, and it reports the analysis carried out in González et al. (2005). The algorithms are applied to one example of linear, pseudo-modular, and unimax functions. The average first hitting time is analyzed and compared. This section also addresses the following issues in EDAs: (i) the relationship between the complexity of the probabilistic model used by the algorithm and its efficiency, and (ii) the matching between this model and the relationship among the variables of the objective function.

To finalize, conclusions are drawn in Section 8.4

8.1 Introduction

In working with EAs, it is important to calculate the first hitting time for a given objective function, and to know what is the best solution the algorithm can provide for a given computational time. Time complexity is a particularly crucial issue in the analysis of EAs. It shows how efficient an algorithm is for a large problem. However, for most algorithms and functions, time complexity is relatively unknown (except for some cases), and few results are available (Eiben and Rudolph, 1999; He and Yao, 2001; Rudolph, 1998; Wolpert and Macready, 1997). This fact makes theoretical comparisons between EAs and other optimization algorithms difficult.

Time complexity has been theoretically analyzed mainly for $(1+1)$ algorithms. The first results were made available by Rudolph (1997), who studied the application of $(1+1)$ EAs to $OneMax$, linear, and general unimodal problems.

Droste et al. (1998) offer a rigorous complexity analysis of (1+1) EAs for linear functions with boolean inputs. In a later work Droste et al. (2002) prove that linear functions are optimized in expected time $O(n \ln n)$, when the mutation rate is $1/n$ (where n is the size of the individual), while the optimization of some degree 2 polynomials, and a unimodal function need exponential time.

He and Yao (2001) considered EAs which use a population size greater than 1, crossover, mutation and selection. In their work, using drift analysis, conditions under which an EA will take no more than polynomial time (in problem size) to solve a problem and conditions under which an EA will take at least exponential time (in problem size) are obtained. This work was extended (He and Yao, 2002) to carry out a comparison between the first hitting time between (1+1) EAs and $(N + N)$ EAs. A further extension of He and Yao (2001) can be found in He and Yao (2004), where a general classification of easy and hard problems for EAs is added. He and Yao (2003) have also proposed a general framework based on Markov chains for analyzing first hitting times for EAs.

8.2 Worst-case First Hitting Time Analysis for EDAs

This section is intended as a first step toward furthering the theoretical study of time complexity of EDAs. Based on the work of He and Yao (2003), we have modeled EDAs by means of an absorbing Markov chain. This model enables us to show the following result: the UMDA, MIMIC, TREE and EBNA$_{BIC}$ algorithms need in the worst-case, exponential expected time (in individual size) for their first hitting time in any pseudo-boolean injective function.

It is clear that the calculation of the mean first hitting time of an algorithm is highly important to its analysis. But surely people reading this dissertation will wonder why it is interesting to calculate the algorithm's worst-case first hitting time. The reason is the following: our calculation of the worst-case first hitting time for some EDAs has major implications for the derivation of bounds of the expected first hitting time for EDAs, and helps to better understand how these algorithms work. Moreover, our result provides clues for recognizing situations in which the worst-case hitting time is exponential: when there exists a population of selected individuals such that the probability for the EDA to visit populations with better individuals is very small.

We want to point out that for the EDAs used in this section we take into account the following assumptions:

Assumption 8.1. *The estimated probability distribution assigns positive probability to each point of the search space at each step of the algorithm (this can be guaranteed using estimations based on the Laplace correction (Cestnik, 1990)).*

Assumption 8.2. *The population of selected individuals at time t, D_t^{Se} is obtained as follows: D_t is obtained sampling $M-1$ times $p(\mathbf{x}|D_{t-1}^{Se})$, and adding the best individual of the previous selected population, D_{t-1}^{Se}. After that the N best individuals of D_t are selected to obtain the population D_t^{Se}.*

Assumption 8.3. *All the variables are binary.*

8.2.1 Modeling EDAs by Means of an Absorbing Markov Chain

This section is devoted to modeling EDAs using Markov chains. Here we give the general framework necessary to obtain the result (adapted from He and Yao (2003)) that characterizes those EDAs whose first hitting time is exponential in the worst case.

Let us introduce some notation. The pseudo-boolean combinatorial optimization problem to solve will be:

$$\max_{\mathbf{x}\in\Omega} f(\mathbf{x}), \tag{8.1}$$

where $f : \Omega \to I\!R$ is the objective function and $\Omega = \{0,1\}^n$ denotes the search space.

Let $\mathbf{x} \in \Omega$ be a solution of an EDA, which is represented by an individual. A population D_t is a set of M individuals (in the multiset sense, it could contain repeated individuals). The selected population at step t is denoted by D_t^{Se}, has N individuals and:

$$D_t^{Se} \in \{(\mathbf{x}_1, \ldots, \mathbf{x}_N) | \mathbf{x}_i \in \Omega\}.$$

The Markov chain used here to model EDAs (Markov model III) is sightly different than the two used previously (Markov model I and Markov model II). In this case and given that the selected population at time t, D_t^{Se}, only depends on the selected population at time $t-1$, D_{t-1}^{Se}, we can model EDAs by means of the following Markov chain:

$$(D_t^{Se} : t = 0, 1, \ldots) \tag{8.2}$$

whose state space E consists of all possible selected populations (of size N). Let r be the number of different populations of size N that could be formed, then:

$$E = \{1, \ldots, r\},$$

where i denotes the i^{th} population of size N. If no self-adaptation is used (as in our case) the Markov chain is homogeneous.

Let Ω_{opt} be the set of solutions of problem (8.1). We divide the set of selected populations into two disjointed subsets. On the one hand, we have the set of all selected populations that contain at least an optimal solution:

$$H = \{i \in E \mid \exists \mathbf{x}^* \in \Omega_{opt} \text{ such that } \mathbf{x}^* \in i\}. \tag{8.3}$$

On the other, the set of selected populations that does not contain any optima:

$$T = \{i \in E \mid \nexists \mathbf{x}^* \in \Omega_{opt} \text{ such that } \mathbf{x}^* \in i\}. \tag{8.4}$$

The cardinality of H and T will be denoted by $|H|$ and $|T|$ respectively. We also order the states as follows: the first states are the states in H, while the states in T are in the last places. Therefore the transition matrix associated with the Markov chain is:

$$\bar{\mathbf{P}} = \begin{bmatrix} \mathbf{H} & \mathbf{0} \\ \mathbf{R} & \mathbf{T} \end{bmatrix},$$

where $\mathbf{H}$ is a matrix whose dimension is $|H| \times |H|$ and each entry of $\mathbf{H}$, P_{ij}, represents the probability of going from the selected population i that contains at least an optimum to the selected population j that contains at least an optimum. $\mathbf{T}$ has dimension $|T| \times |T|$ and its entries P_{ij} represent the probability of going from the selected population i that does not contain any optima to the selected population j that does not contain any optima. Matrix $\mathbf{0}$ is the null matrix of dimension $|H| \times |T|$. Finally we describe matrix $\mathbf{R}$, which is a matrix of dimension $|T| \times |H|$ and contains the probabilities of going from a selected population that does not contain any optima to a selected population that contains at least one optimum.

When set H is reached the algorithm is trapped in it and it is impossible to leave this set (we consider Assumption 2). We are only interested in the first hitting time, and do not care about the behavior after that. Therefore we can substitute matrix $\mathbf{H}$ for the identity matrix $\mathbf{I}_{|H|}$. Now it is sufficient to analyze the Markov chain associated whith the following transition matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{|H|} & \mathbf{0} \\ \mathbf{R} & \mathbf{T} \end{bmatrix} .$$

The absorbing states in this chain are the states belonging to set H. Given that the states in H can be reached from any other state, the chain is absorbing.

Once we have modeled the algorithm by means of the above absorbing Markov chain, we are interested in the first time in which an optimum is reached. As defined in Section 2.3.2, the random variable that indicates the first hitting time to the set H – the set of populations that contain at least an optimal point – when starting from state i, is:

$$\tau_i = \min\{t; t \geq 0, D_t^{Se} \in H | D_0^{Se} = i\}.$$

We are interested in the expectation of the random variable τ_i:

$$m_i = \mathsf{E}[\tau_i; \tau_i < \infty] .$$

The vector whose components represent the expected first hitting time depending on the initial state i, is $\mathbf{m} = [m_i]_{i \in E}$. It can be proven that this vector can be calculated as follows:

$$\mathbf{m} = (\mathbf{I}_{|T|} - \mathbf{T})^{-1}\mathbf{1} , \tag{8.5}$$

where $\mathbf{1}$ denotes the $|T|$-dimensional vector $(1, \ldots, 1)^t$.

For vector $\mathbf{m}$ we can define different norms. Two special norms are:

$$\begin{aligned} \|\mathbf{m}\|_0 &= \frac{1}{|E|}\sum_{i \in E} m_i . \\ \|\mathbf{m}\|_\infty &= \max_{i \in E} m_i . \end{aligned}$$

The above norms represent two different performance measures of the expected first hitting time. Norm $\|\mathbf{m}\|_0$ is related to the average-case analysis (when the initial distribution of the states is uniform), while norm $\|\mathbf{m}\|_\infty$ is related to the worst-case analysis.

The calculation of both quantities, $\|\mathbf{m}\|_\infty$ and $\|\mathbf{m}\|_0$ is interesting for the analysis of any EDA. Here we have focused our efforts on the calculation of $\|\mathbf{m}\|_\infty$. We also see how the calculation of $\|\mathbf{m}\|_\infty$ can help to bound the quantity $\|\mathbf{m}\|_0$.

Now we give a result that characterizes those EDAs whose first hitting time, in the worst-case analysis, is exponential in the individual size. This result is adapted from He and Yao (2003). The proof is based on the calculation of the formula (8.5), which requires handling matrix $\mathbf{P}$. Since the number of entries in the transition matrix is often huge (exponential in individual and population size), we decompose the state space E into $L+1$ disjointed subsets $E_0, E_1, \ldots, E_L$, where E_l, $l = 0, \ldots, L$, contains as best individual the $(l+1)^{th}$ best individual of the search space.

Theorem 8.1. *For the Markov chain associated with an EDA such that it fulfills Assumption 8.1 and Assumption 8.2, $\|\mathbf{m}\|_\infty$ is exponential in n if and only if, for some l, $0 \leq l \leq L$ and for some population i with $i \in E_l$,*

$$\frac{1}{\sum_{k=0}^{l-1}\sum_{j \in E_k} P(j \mid i)} \tag{8.6}$$

is exponential in n.

The condition associated with expression (8.6) means that starting from some state (population i), in a subset of populations E_l, the probability for the EDA to move to upper subsets with higher fitness is very small.

8.2.2 Worst Case Analysis for Some EDAs

This section is concerned with proving the main result of this chapter: UMDA, MIMIC, TREE and EBNA$_{BIC}$ algorithms fulfill the condition of Theorem 8.1, when they are used to maximize any pseudo-boolean injective function. Therefore we must show that in the worst-case analysis, the first hitting time for these algorithms is exponential in n.

It is important to remember that the algorithms we are analyzing satisfy Assumption 1, Assumption 2 and Assumption 3.

Our aim is to prove that Theorem 8.1 is true for the different algorithms. To this end we will demonstrate that the population composed of the second best individual and $N-1$ copies of the complement of the best individual (which we denote by i^*) fulfills the condition of Theorem 1.

Given that population i^* belongs to subset E_1 (contains as best individual the second best individual), we must write the expression (8.6) for $l = 1$:

$$\frac{1}{\sum_{j \in E_0} \mathsf{P}(j \mid i^*)} \ . \tag{8.7}$$

If we want to see that i^* meets the condition of Theorem 8.1 for these algorithms, we must show that (8.7) is exponential in n in all cases.

Without loss of generality we prove the condition in a situation in which the optimum is the point $(1, \ldots, 1)$ and the second best individual is $(1, \ldots, 1, 0)$. Notice that which are the best and the second best individuals is irrelevant for the proof. In this case the population of selected individuals that we analyze is $i^* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_N^*\}$, where:

$$\begin{aligned} \mathbf{x}_1^* &= (0, \ldots, 0, 0) \\ &\ldots \\ \mathbf{x}_{N-1}^* &= (0, \ldots, 0, 0) \\ \mathbf{x}_N^* &= (1, \ldots, 1, 0) \ . \end{aligned}$$

The next step is to calculate $\mathsf{P}(j \mid i^*)$, the probability of going from population i^* to a population $j \in E_0$, i.e. to a population that contains the individual $(1, \ldots, 1)$. In order to do so we have to estimate the joint probability distribution associated with the population of the selected individuals i^* for each algorithm. This estimation depends on the probabilistic model used by the algorithm. This is the reason why from this point we analyze the problem separately for UMDA, MIMIC, TREE and EBNA$_{BIC}$ algorithms.

8.2.2.1 Worst Case Analysis for UMDA

The UMDA algorithm uses the simplest model to estimate the joint probability distribution of the selected individuals, supposing that all the variables are independent. Here we use a version of the algorithm that fulfills Assumptions 1, 2 and 3 mentioned above. In order to satisfy the first Assumption and taking into account that the variables are binary, we use estimations based on the Laplace correction as follows:

$$\mathsf{P}(X_i = 1 \mid D_{t-1}^{Se}) = \frac{N_i + 1}{N + 2} \ , \tag{8.8}$$

where N_i is the number of times variable X_i takes value 1 in D_{t-1}^{Se}.

Taking into account that the population of selected individuals is i^* and using (8.8) to calculate the univariate marginal distributions, the probability distribution associated with selected individuals can be written as a vector:

$$\mathbf{p} = \left(\frac{2}{N+2}, \frac{2}{N+2}, \ldots, \frac{2}{N+2}, \frac{1}{N+2} \right) \ ,$$

where the i^{th} component of vector $\mathbf{p}$, p_i is the probability of obtaining a one in the i^{th} position, i.e. $p_i = \mathsf{P}(X_i = 1)$.

Therefore at the next step of the algorithm, the following population is obtained sampling the probability vector $\mathbf{p}$. Thus the denominator of quotient (8.7) for population i^* can be expressed as:

$$\sum_{j \in E_0} \mathsf{P}(j|i^*) = \sum_{j \in E_0} \mathsf{P}(j|\mathbf{p}) \ ,$$

where $\mathsf{P}(j|\mathbf{p})$ is the probability of obtaining population j after sampling the probability vector $\mathbf{p}$. Thus:

$$\sum_{j \in E_0} \mathsf{P}(j|\mathbf{p}) \quad = \quad 1 - \mathsf{P}(j \notin E_0|\mathbf{p}) = 1 - \mathsf{P}((1,1,\ldots,1) \notin j|\mathbf{p}) \ .$$

Expression $\mathsf{P}((1,1,\ldots,1) \notin j|\mathbf{p})$ denotes the probability after sampling $M-1$ times $\mathbf{p}$, of not obtaining individual $(1,1,\ldots,1)$. Given that the probability of obtaining $(1,1,\ldots,1)$ after sampling $\mathbf{p}$ once is:

$$\left(\frac{2}{N+2} \right)^{n-1} \cdot \frac{1}{N+2} \ ,$$

the probability of not obtaining $(1,1,\ldots,1)$ after sampling $\mathbf{p}$ once is:

$$1 - \left(\frac{2}{N+2} \right)^{n-1} \cdot \frac{1}{N+2} \ .$$

Thus, the probability of not obtaining $(1,1,\ldots,1)$ after sampling $\mathbf{p}$ $M-1$ times is given by:

$$\mathsf{P}((1,1,\ldots,1) \notin j|\mathbf{p}) = \left(1 - \left(\frac{2}{N+2} \right)^{n-1} \cdot \frac{1}{N+2} \right)^{M-1} \ .$$

Finally, we have:

$$\sum_{j \in E_0} \mathsf{P}(j|\mathbf{p}) = 1 - \left(1 - \left(\frac{2}{N+2} \right)^{n-1} \cdot \frac{1}{N+2} \right)^{M-1} \ . \tag{8.9}$$

If we set M and N such that they depend linearly on n, we find that (8.9) decreases exponentially in n, with the result that quotient (8.7) is exponential in n.

8.2.2.2 Worst Case Analysis for MIMIC

The probabilistic model used by MIMIC takes into account dependencies between pairs of variables. Denoting by $\pi = (i_1, \ldots, i_n)$ a permutation of the indexes $1, 2, \ldots, n$, the probabilistic model used by MIMIC can be written as:

$$p_t^{\pi}(\mathbf{x}) = p_t(x_{i_1}|x_{i_2}) \cdot p_t(x_{i_2}|x_{i_3}) \cdot \ldots \cdot p_t(x_{i_{n-1}}|x_{i_n}) \cdot p_t(x_{i_n}).$$

In order to learn the structure, a greedy algorithm is used. At each generation the objective is to find the best permutation π among the variables such that it minimizes the Kullback-Leibler divergence between $p_t^{\pi}(\mathbf{x})$ and the empirical distribution of the set of selected points.

Therefore we have to find the order of the variables (a permutation π^* in the indexes), using the algorithm in Figure 3.7. The first step consists of calculating the empirical Shannon entropy of all the variables (taking into account that the population of selected individuals is i^*) and choosing the minimum of them. Given that the first $n-1$ variables have the same values, we find that for $i = 1, \ldots, n-1$:

$$\begin{aligned} \hat{H}(X_i) &= -\sum_x \mathsf{P}(X_i = x) \log \mathsf{P}(X_i = x) \\ &= -\left[\mathsf{P}(X_i = 0) \log \mathsf{P}(X_i = 0) + \mathsf{P}(X_i = 1) \log \mathsf{P}(X_i = 1)\right] \ . \end{aligned}$$

Given the set of selected individuals the parameters are estimated using the Laplace correction with the following expressions:

$$\mathsf{P}(X_i = x_i | X_j = x_j) = \frac{N_{ij} + 1}{N_j + 2} \ , \tag{8.10}$$

$$\mathsf{P}(X_i = x_i) = \frac{N_i + 1}{N + 2} \ , \tag{8.11}$$

where N_{ij} is the number of times variable X_i takes value x_i and variable X_j takes value x_j in i^*, and N_i (resp. N_j) is the number of times variable X_i (resp. X_j) takes value x_i (resp. x_j) in i^*.

Therefore using expression (8.11) for the estimation of the parameters we obtain:

$$\begin{aligned} \hat{H}(X_i) &= -\left[\left(\frac{N}{N+2}\right) \log \left(\frac{N}{N+2}\right) + \left(\frac{2}{N+2}\right) \log \left(\frac{2}{N+2}\right)\right] \\ &= -\left[\log\left(\frac{1}{N+2}\right) + \frac{N \log N + 2 \log 2}{N+2}\right] \ . \end{aligned}$$

The empirical entropy for the variable X_n is:

$$\begin{aligned} \hat{H}(X_n) &= -\left[\left(\frac{N+1}{N+2}\right) \log \left(\frac{N+1}{N+2}\right) + \left(\frac{1}{N+2}\right) \log \left(\frac{1}{N+2}\right)\right] \\ &= -\left[\log\left(\frac{1}{N+2}\right) + \frac{(N+1)\log(N+1)}{N+2}\right] \ . \end{aligned}$$

It is easy to see that for $N \geq 2$, $\hat{H}(X_i) > \hat{H}(X_n)$ with $i = 1, \ldots, n-1$. For this reason variable X_n is selected at step 1 of the algorithm in Figure 3.7. Therefore, given that $\hat{H}(X_i|X_n)$ is equal for all $i = 1, \ldots, n-1$, one possible permutation for the variables is $\pi^* = (i_1^* = 1, i_2^* = 2, \ldots, i_{n-1}^* = n-1, i_n^* = n)$ (the choice of this permutation does not affect the final result). The probabilistic model used by MIMIC in this case is:

$$p^{\pi^*}(\mathbf{x}) = p(x_1|x_2) \cdot p(x_2|x_3) \cdot \ldots \cdot p(x_{n-1}|x_n) \cdot p(x_n).$$

Thus the denominator of quotient (8.7) for population i^* can be expressed as:

$$\sum_{j \in E_0} \mathsf{P}(j|i^*) = \sum_{j \in E_0} \mathsf{P}(j|p^{\pi^*}(\mathbf{x})) \ ,$$

where $\mathsf{P}(D^{Se}|p^{\pi^*}(\mathbf{x}))$ is the probability of obtaining population j after sampling the joint probability distribution $p^{\pi^*}(\mathbf{x})$. Hence, arguing as in the case of UMDA algorithm, we find that:

$$\sum_{j \in E_0} \mathsf{P}(j|p^{\pi^*}(\mathbf{x})) = 1 - \mathsf{P}((1,1,\ldots,1) \notin j|p^{\pi^*}(\mathbf{x})) \ .$$

The probability of obtaining one $(1,1,\ldots,1)$ after sampling $p^{\pi^*}(\mathbf{x})$ once is:

$$\begin{aligned} &\mathsf{P}((1,1,\ldots,1)|p^{\pi^*}(\mathbf{x})) \\ &\quad = \mathsf{P}(X_1 = 1|X_2 = 1) \cdot \mathsf{P}(X_2 = 1|X_3 = 1) \cdot \ldots \cdot \mathsf{P}(X_{n-1} = 1|X_n = 1) \cdot \mathsf{P}(X_n = 1) \ . \end{aligned}$$

Taking into account that the parameters are estimated by means of expressions (8.10) and (8.11):

$$\mathsf{P}((1,1,\ldots,1)|p^{\pi^*}(\mathbf{x})) = \left(\frac{2}{3}\right)^{n-2} \cdot \frac{1}{2} \cdot \frac{1}{N+2} \ ,$$

which implies that:

$$\sum_{j \in E_0} \mathsf{P}(j|p^{\pi^*}(\mathbf{x})) = 1 - \left(1 - \left(\left(\frac{2}{3}\right)^{n-2} \cdot \frac{1}{2} \cdot \frac{1}{N+2}\right)\right)^{M-1} \ . \tag{8.12}$$

Finally, taking M and N linear in n we clearly find that (8.12) decreases exponentially in n. This proves that quotient (8.7) is exponential in n in the case of MIMIC algorithm.

8.2.2.3 Worst Case Analysis for TREE

In this algorithm the dependency structure among the variables forms a tree. Estimation of the tree structure of the probability distribution of the selected individuals is carried out using the Chow and Liu algorithm (see Figure 3.8). Furthermore, in this case the parameters, given the structure, are calculated using the Laplace correction by means of the following expressions:

$$\mathsf{P}(X_i = x_i, X_j = x_j) = \frac{N_{ij} + 1}{N + 4} \ , \tag{8.13}$$

$$\mathsf{P}(X_i = x_i|X_j = x_j) = \frac{N_{ij} + 1}{N_j + 2} \ , \tag{8.14}$$

$$\mathsf{P}(X_i = x_i) = \frac{N_i + 1}{N + 2} \ , \tag{8.15}$$

where N_{ij} is the number of times variable X_i takes value x_i and variable X_j takes value x_j in D_{t-1}^{Se} (which is i^*), and N_i (resp. N_j) is the number of times variable X_i (resp. X_j) takes value x_i (resp. x_j) in D_{t-1}^{Se} (which is i^*).

Hence, using the Chow Liu (1968) algorithm in order to find the tree structure, first we must calculate the mutual information for all variable pairs. Because the mutual information between each pair of the first $n-1$ variables takes the same value, as well as the mutual information between variable X_n and any of the $n-1$ first variables, it is sufficient to calculate $I(X_i, X_{i+1})$ for some $i = 1, \ldots, n-2$ and $I(X_i, X_n)$ for some $i = 1, \ldots, n-1$.

For $i = 1, \ldots, n-2$:

$$I(X_i, X_{i+1}) = \sum_{x_i, x_{i+1}} \mathsf{P}(X_i = x_i, X_{i+1} = x_{i+1}) \log \left(\frac{\mathsf{P}(X_i = x_i, X_{i+1} = x_{i+1})}{\mathsf{P}(X_i = x_i)\mathsf{P}(X_{i+1} = x_{i+1})} \right) \ .$$

Given that the parameters are estimated using expressions (8.13) and (8.15):

$$\begin{aligned} & I(X_i, X_{i+1}) \\ & = \frac{N}{N+4} \log \left(\frac{\frac{N}{N+4}}{\frac{N}{N+2}\frac{N}{N+2}} \right) + \frac{2}{N+4} \log \left(\frac{\frac{1}{N+4}}{\frac{2}{N+2}\frac{N}{N+2}} \right) + \frac{2}{N+4} \log \left(\frac{\frac{2}{N+4}}{\frac{2}{N+2}\frac{2}{N+2}} \right). \end{aligned}$$

By carrying out some basic calculations, we obtain:

$$I(X_i, X_{i+1}) = \log \left(\frac{(N+2)^2}{N+4} \right) - \frac{(N+2)\log N + 4\log 2}{N+4} \ . \tag{8.16}$$

For $i = 1, \ldots, n-1$, and estimating the parameters as above, we find that:

$$\begin{aligned} & I(X_i, X_n) \\ & = \frac{N}{N+4} \log \left(\frac{\frac{N}{N+4}}{\frac{N+1}{N+2}\frac{N}{N+2}} \right) + \frac{1}{N+4} \log \left(\frac{\frac{1}{N+4}}{\frac{N}{N+2}\frac{1}{N+2}} \right) \\ & + \frac{2}{N+4} \log \left(\frac{\frac{2}{N+4}}{\frac{2}{N+2}\frac{N+1}{N+2}} \right) + \frac{1}{N+4} \log \left(\frac{\frac{1}{N+4}}{\frac{1}{N+2}\frac{2}{N+2}} \right) \ . \end{aligned}$$

Further calculations yield the following:

$$I(X_i, X_n) = \log \left(\frac{(N+2)^2}{N+4} \right) - \frac{(N+2)\log(N+1) + \log N + \log 2}{N+4}. \tag{8.17}$$

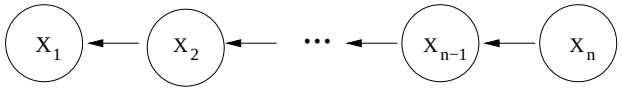
It can be seen that for $N \geq 2$, expression (8.16) is greater than expression (8.17). Taking into account that expression (8.16) is equal for each $i = 1, \ldots, n-2$, and that expression (8.17) is equal for $i = 1, \ldots, n-1$, there are many possible tree structures that can be the result of the Chow and Liu algorithm. All of them can be listed following these steps:

1 Find one of the possible trees that can be formed with the first $n-1$ variables.

2 Connect variable X_n to one variable of the tree formed in the previous step.

Consequently, we can choose any of the possible structures formed as above. We have chosen the following:

Given that the above structure is the same as in the case of MIMIC algorithm, from this point on the demonstration is equal to the demonstration in the MIMIC case. It is important to note that the resulting tree does not affect the final result.



8.2.2.4 Worst Case Analysis for EBNA$_{BIC}$

In the EBNA$_{BIC}$ algorithm, learning the probabilistic model means learning a Bayesian network (whose structure is denoted by S) from the selected individuals, in our case from $i^* = \{\mathbf{x}_1^*, \dots \mathbf{x}_N^*\}$. The structure is searched using Algorithm B, which starts with an arc-less structure, and at each step, adds the arc with the maximum improvement in the BIC score (based on penalized maximum likelihood), which evaluates how well the Bayesian network represents the probability distribution of a database. Given a database D and a Bayesian network with structure S and set of parameters $\boldsymbol{\theta}$, $(S, \boldsymbol{\theta})$, a general formula for a penalized maximum likelihood score can be written as follows:

$$\log \mathsf{P}(D|S, \boldsymbol{\theta}) - f(N)dim(S) \ ,$$

where $\mathsf{P}(D|S, \boldsymbol{\theta})$ denotes the probability of obtaining the data D sampling the Bayesian network $(S, \boldsymbol{\theta})$, $dim(S)$ is the dimension – number of parameters needed to specify the model – of the Bayesian network with a structure given by S, and $f(N)$ is a non negative penalization function. Thus:

$$dim(S) = \sum_{i=1}^{n} q_i(r_i - 1) \ ,$$

where q_i denotes the number of possible different instances of the parent variables of X_i (if X_i has no parent, $q_i = 1$). The Jeffreys-Schwarz criterion, sometimes called the Bayesian Information Criterion (BIC) (Schwarz, 1978) establishes $f(N) = \frac{1}{2} \log N$. Thus the BIC score can be written as follows:

$$BIC(S, D) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} \mathsf{P}(X_i = x_{w,i}|\mathbf{pa}_i^S, \boldsymbol{\theta}_i) - \frac{1}{2} \log N \sum_{i=1}^{n} q_i(r_i - 1) \ .$$

The parameters of the Bayesian network are calculated using the Laplace correction:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + 1}{N_{ij} + 2} \ . \tag{8.18}$$

Note that in this case N_{ij} does not represent the same thing as in the previous cases. Here N_{ij} is the number of individuals in j_{t-1} in which variables $\boldsymbol{Pa}_i^S$ take their j^{th} value and N_{ijk} is the number of individuals in j_{t-1} in which variable X_i takes its k^{th} value and variables $\boldsymbol{Pa}_i^S$ take their j^{th} value.

At each step of Algorithm B, there is not only one arc with the maximum improvement in the BIC score. This is the reason why there are a number of final structures resulting from the application of Algorithm B. Any of those structures can be chosen and it does not influence our reasoning.

In Appendix B we prove in detail why one possible final structure after $n-1$ steps of Algorithm B – denoted by S_{n-1} – is the structure that appears in Figure 8.1.

Taking into account the structure in Figure 8.1, the probabilistic model used by EBNA$_{BIC}$ in this case is:

$$p^*(\mathbf{x}) = p(x_1|x_2) \cdot p(x_2|x_3) \cdot \ldots \cdot p(x_{n-2}|x_{n-1}) \cdot p(x_{n-1}) \cdot p(x_n)$$

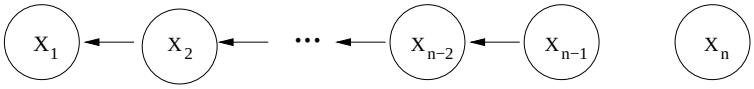


Figure 8.1. One possible final structure S_{n-1}.

Therefore we can write the denominator of quotient (8.7) for population i^*:

$$\sum_{j \in E_0} \mathsf{P}(j|i^*) = \sum_{j \in E_0} \mathsf{P}(j|p^*(\mathbf{x})) \ ,$$

where $\mathsf{P}(j|p^*(\mathbf{x}))$ is the probability of obtaining population j after sampling the joint probability distribution $p^*(\mathbf{x})$. Therefore, using the same arguments as for the previous algorithms:

$$\sum_{j \in E_0} \mathsf{P}(j|p^*(\mathbf{x})) = 1 - \mathsf{P}((1,1,\ldots,1) \notin j|p^*(\mathbf{x})) \ .$$

The probability of obtaining one $(1,1,\ldots,1)$ after sampling $p^*(\mathbf{x})$ once is:

$$\begin{aligned} &\mathsf{P}((1,1,\ldots,1)|p^*(\mathbf{x})) \\ &\quad = \ \mathsf{P}(X_1 = 1|X_2 = 1) \cdot \ldots \cdot \mathsf{P}(X_{n-2} = 1|X_{n-1} = 1) \cdot \mathsf{P}(X_{n-1} = 1) \cdot \mathsf{P}(X_n = 1). \end{aligned}$$

Now we estimate the parameters:

$$\mathsf{P}((1,1,\ldots,1)|p^*(\mathbf{x})) = \left(\frac{2}{3}\right)^{n-2} \cdot \frac{2}{N+2} \cdot \frac{1}{N+2} \ .$$

Finally we find that:

$$\sum_{j \in E_0} \mathsf{P}(j|p^*(\mathbf{x})) = 1 - \left(1 - \left(\left(\frac{2}{3}\right)^{n-2} \cdot \frac{2}{(N+2)^2}\right)\right)^{M-1} \ . \tag{8.19}$$

Assuming that M and N depend linearly on n it can be seen that (8.19) decreases exponentially in n, meaning that quotient (8.7) is exponential in n in the case of EBNA$_{BIC}$ algorithm.

8.2.3 The Average-case Analysis

In the previous section we showed for some instances of EDAs that $\|\mathbf{m}\|_\infty$ is exponential in n. This fact does not imply that the first hitting time is exponential in the average-case for these EDAs. In fact we will prove (empirically) in the next section that some problems – the optimization of a linear function, a pseudo-modular one, and other unimax function – can be solved in polynomial time in the average-case.

The result found for the worst-case analysis for EDAs helps us in the study of the average-case analysis. Here we explain how the bounding of $\|\mathbf{m}\|_0$ is influenced by the fact of $\|\mathbf{m}\|_\infty$ being exponential in n.

Remember that:

$$\|\mathbf{m}\|_0 \;=\; \frac{1}{|E|}\sum_{i\in E} m_i \;.$$

Quantity $\|\mathbf{m}\|_0$ is a quotient whose denominator is the number of different populations $|E|$ which is given by the different ways to place $|\Omega|$ balls into $N+|\Omega|-1$ boxes. Therefore, taking into account that $|\Omega| = 2^n$:

$$|E| = \binom{N+2^n-1}{2^n-1} \;.$$

Based on the expected first hitting time we can divide the states in E into two different classes:

- $E(P)$ is the class of the states $i \in E$ such that m_i is polynomial in n.
- $E(NP)$ is the class of the states $i \in E$ such that m_i is exponential in n.

Taking this division into account we can write $\|\mathbf{m}\|_0$ as follows:

$$\|\mathbf{m}\|_0 \;=\; \frac{1}{|E|}\left(\sum_{i\in E(P)} m_i + \sum_{i\in E(NP)} m_i\right).$$

As we explain above, in the next section we will experimentally prove for some functions that $\|\mathbf{m}\|_0$ can be bounded by a polynomial. Therefore the first idea for bounding $\|\mathbf{m}\|_0$ consists of finding a general polynomial bound for all m_i, but the existence of some populations in $E(NP)$ keep us from finding it. This is the reason why, if we want to find a polynomial bound for $\|\mathbf{m}\|_0$, we must first find a polynomial bound for $|E(NP)|$ (which is really difficult).

In particular, for those cases in which $|E(NP)|/|E|$ is an inverse polynomial of n, $\|\mathbf{m}\|_0$ is exponential in n.

8.3 Average Time Complexity of EDAs

The previous section provides an idea of how hard it can be to find (theoretically) a bound for the average time complexity of an EDA. Nevertheless, in order to give some idea of the average time complexity of these algorithms, a good starting point would seem to be an empirical analysis of their first hitting time.

In this section we present empirical results of the average first hitting time for some instances of EDAs: UMDA, TREE and EBNA$_{BIC}$ applied to one example of linear, pseudo-modular and unimax functions. The particular algorithms used in this Section have been chosen to provide one instance of each level of complexity of the probabilistic model. Our aim is to give information about the average first hitting time of the EDA approaches studied, and to compare them.

We also want to address some important questions related to the order of convergence that have been raised in recent research on EDAs (Mühlenbein and Mahnig, 1999; Pelikan et al., 2002b; Mühlenbein and Mahnig, 2002b): Does the average first hitting time decrease when the complexity of the probabilistic model used by the algorithm increases? Is the algorithm's capability of detecting and exploiting the (in)dependencies of the objective function related to efficiency?

8.3.1 The Functions Used

To carry out the experiments, we have chosen one particular function of each of the following classes of pseudo-boolean functions: linear, pseudo-modular and unimax functions. Section 6.1.3 in a previous chapter introduces in detail those classes of functions.

The **linear function** analyzed in this work is:

$$f(\mathbf{x}) = \sum_{i=1}^{n} i \cdot x_i, \quad x_i \in \{0, 1\} \tag{8.20}$$

It is clear that $(1, \ldots, 1)$ is the only global maximum for function (8.20), and the value of $f(\mathbf{x})$ at this point is $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$.

It should be stressed that function (8.20) can be optimized individually, variable to variable. Taking this fact into account a suitable probabilistic model that corresponds to the relationship between the variables of function (8.20) is one that supposes that all the variables are independent. Therefore UMDA seems a good candidate to optimize (8.20).

The **pseudo-modular function** we have used is:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \quad x_j \in \{0, 1\} \tag{8.21}$$

For fitness function (8.21) the only optimal solution is $(1, \ldots, 1)$, and the value of $f(\mathbf{x})$ at this point is n.

The variables of function (8.21) present a chain of pairwise dependencies, where x_i depends on x_{i-1}, $i = 1, \ldots, n-1$. This fact suggests MIMIC as a suitable model to optimize (8.21).

The **unimax function** chosen to carry out the experiments was the well-known long path problem (see Section 6.1.3). It is important to remember that this function only has sense for odd values of n.

The optimal point in the long path problem is $(1, 1, 0, \ldots, 0)$ and the function value is 0.

The long path problem has been designed to be very difficult to optimize for a hill climbing algorithm. Particularly the expected first hitting time for a hill climbing algorithm in this problem is exponential in the dimension of the problem n (the algorithm has to go from one point on the path to the next point on the path and the path has exponential length).

In long path problems the relationship between the variables of the problem are not so evident, and EBNA algorithms therefore seem to be the most adequate to optimize these functions.

8.3.2 Experimental Results

In order to find the average first hitting time of the EDAs we are studying, we have carried out experiments using those algorithms to maximize the different functions in Section 8.3.1. The empirical results for each algorithm and each objective function have been fitted to curves (in the least squares sense) in terms of problem size, using the "*Mathematica*" program. We have measured the "goodness" of each fit with the coefficient of determination and the mean squared error. After comparing those quantities for various fits we have chosen the fit with the highest coefficient of determination and lowest mean squared error. Cotta and Moscatto (2003) was an interesting reference on how a fitting must be done, in this work empirical data from EAs is fitted to severeal complexity curves. Finally, all this information has been graphically presented.

8.3.2.1 General Considerations

In this section we will set up the parameters and decisions that are common to all EDAs used in the experiments. These characteristics are: (i) size of the population, (ii) stopping condition, (iii) selection method, (iv) number of selected individuals and (v) type of elitism.

The population size, M, was fixed to avoid dependency of the order of convergence on this parameter. Accordingly we fixed M to the individual size n, i.e. the problem dimension. The stopping condition taken into account is the same for each EDA approach: the algorithm is stopped when it finds the optimum for the first time. Truncation selection is the selection method of choice, i.e. the best individuals are selected.

Finally we would like to illustrate the way in which the new population is created in order to set up the number of individuals selected and the type of elitism used. Once the population D_{t-1} is created, the $n/2$ best individuals of D_{t-1} are used to estimate the joint probability distribution $p_t(\mathbf{x})$. After that, $n-1$ individuals are sampled from $p_t(\mathbf{x})$, obtaining D_t^*. Finally the new population D_t is created by selecting the n best individuals from $D_{t-1} \cup D_t^*$. In this way we make sure that the best individual in population D_{t-1} will never be lost.

Remember that if we want to calculate the order of convergence of an algorithm we need to ensure that the algorithm passes through the optimum. This can be done in EDAs (see Chapter 4) by slightly modifying the maximum likelihood estimation of the parameters of the probability distributions at each step. For instance, the Laplace correction (which keeps the parameter from taking a value of 0 or 1) is the estimation used in the experiments:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + 1}{N_{ij} + 2}$$

Notice that all the probabilistic models used for the estimation of $p_l(\mathbf{x})$ can be represented as instances of Bayesian networks. This is the reason why, for those algorithms that do not use Bayesian networks, similar formulas can be given.

A pseudocode for a general algorithm used in the experiments can be seen in Figure 8.2.

We run each algorithm 1,000 times for each objective function and each problem dimension, each time recording the generation in which the optimum was reached for the first time. Due to the computational cost associated with the learning of a Bayesian network at each iteration, the problem dimension of the EBNA algorithm is lower than in the rest. Table 8.1 shows the problem dimension for linear, pseudo-modular and unimax objective functions for each algorithm.

	Linear	**Pseudo-modular**	**Unimax**
UMDA	5 to 300	5 to 150	5 to 65
TREE	5 to 300	5 to 150	5 to 65
EBNA$_{BIC}$	5 to 150	5 to 150	5 to 65

Table 8.1. Problem dimensions used in the experiments with linear, pseudo-modular and unimax functions.

8.3.2.2 Summarizing the Results

After obtaining the results for each algorithm and each objective function, we have found a simple formula that approximates these numerical results, fitting a curve through the data obtained. Furthermore, we can offer a general idea concerning the expected time complexity for each algorithm. We provide here four figures for each function, the last of them being a table. The first three figures

EDA

$D_0 \leftarrow$ Generate $M = n$, individuals (initial population) randomly, with $5 \leq n \leq 300$.
Define: $f_{max} = \max\{f(\mathbf{x}) : \mathbf{x} \in \{0,1\}^n\}$

Repeat for $t = 1, 2, \ldots$ until $f_{D_t} = f_{max}$

$D_{t-1}^{Se} \leftarrow$ Select the $n/2$ best individuals from D_{t-1}

$p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) \leftarrow$ Estimate the joint probability distribution of an individual being among the selected individuals, using the Laplace correction in the estimation of the parameters

$D_t^* \leftarrow$ Sample $n-1$ individuals from $p_t(\mathbf{x})$

$D_t \leftarrow$ Select the n best individuals from $D_{t-1} \cup D_t^*$

$f_{D_t} \leftarrow \max\{f(\mathbf{x}) : \mathbf{x} \in D_t\}$

Figure 8.2. Pseudocode for the EDA approach used in the experiments.

show the results obtained for an EDA when optimizing the objective function. The table entries are: (i) the curve that fits the results obtained for each EDA and (ii) the order of the fitting curve.

Linear function: The results in the linear function can be seen in Figures 8.3 to 8.6. They show that UMDA has the best behavior ($O(n^{\varepsilon})$, $0.6 < \varepsilon < 1$). In this case, the probabilistic model of UMDA, which seemed to be the one that best corresponds with the relationship between the variables of the problem, obtains the best average first hitting time.

Pseudo-modular function: The results for the pseudo-modular function are shown in Figures 8.7 to 8.10. As can be seen in Figure 8.10, all the algorithms have the same linear ($O(n)$) behavior. If we take into account the fitting curves, UMDA's results appear slightly better than the others. Here the correspondence between the probabilistic model used by the algorithm and the relationship between the variables of the problem does not result in a better average first hitting time.

Unimax function: The results for the unimax function are given in Figures 8.11 to 8.14. As can be seen in Figure 8.14 all the algorithms have the same quadratic behavior ($O(n^2)$). Taking into account the fitting curves, TREE seems to have better behavior than the others. Again, the correspondence between the probabilistic model used by the algorithm and the relationship between the variables of the problem does not imply a better average first hitting time.

The results enable us to conclude that:(i) the average optimization time is not related to the complexity of the probabilistic model, (ii) an algorithm whose probabilistic model mimics the structure of the objective function does not guarantee a greater efficiency.

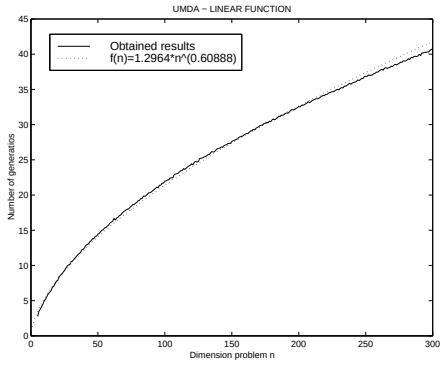


Figure 8.3. UMDA - Linear F.

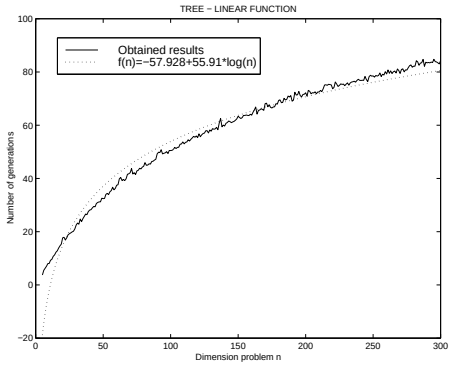


Figure 8.4. TREE - Linear F.

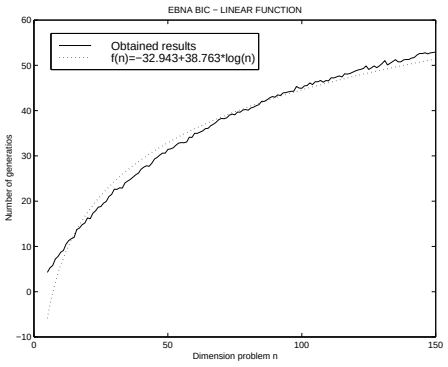


Figure 8.5. EBNA$_{BIC}$ - Linear F.

	Fitting Curve	Order
UMDA	$1.2964 \cdot n^{0.60888}$	$O(n^{\varepsilon})$ $0.6<\varepsilon<1$
TREE	$-57.928+55.91 \log(n)$	$O(\log n)$
EBNA$_{BIC}$	$-32.943+38.763 \log n$	$O(\log n)$

Figure 8.6. Fitting curves for the different EDAs in the optimization of the linear function and their order.

8.4 Summary

In this chapter we have analyzed the time complexity of EDAs from a theoretical stand point. The most important goal of this analysis lies in demonstrating for some instances of EDAs – UMDA, MIMIC, TREE and EBNA$_{BIC}$ – that their worst-case first hitting time is exponential (in individual size) when they are used to maximize any pseudo-boolean injective function. The proof of this result is based on a framework built on the absorbing Markov chain model of EDAs.

The worst-case analysis carried out is useful for discussing how to find bounds in the average-case analysis.

Secondly, based on empirical results, this chapter has provided new information on the average time complexity of some EDAs applied to a number of objective functions.

Our experiments lead to the following conclusions:

i) The average optimization time is not related to the complexity of the probabilistic model used by the algorithm. Greater complexity of the probabilistic model of the EDA does not imply greater efficiency (low order of the first hitting time).

ii) EDAs whose probabilistic model reflects the relationship between the variables of the problem do not obtain a better order of the first hitting time than those that do not.

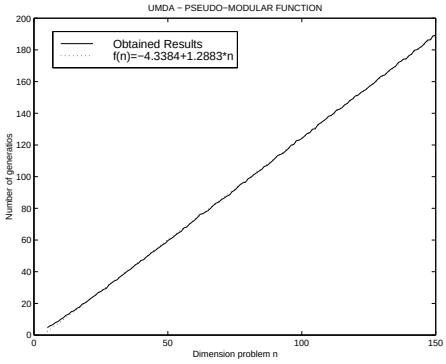


Figure 8.7. UMDA - Pse. Mod. F.

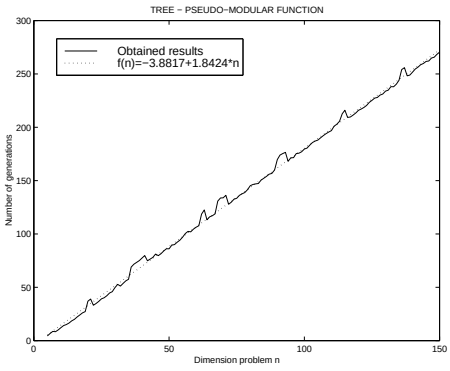


Figure 8.8. TREE - Pse. Mod. F.

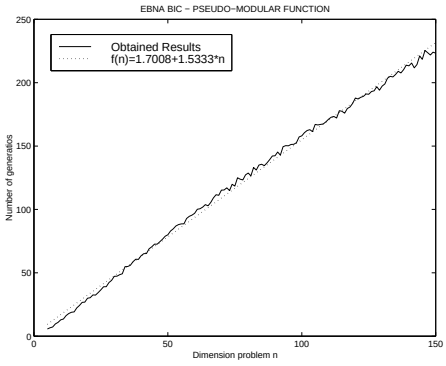


Figure 8.9. EBNA$_{BIC}$ - Pse. Mod. F.

	Fitting Curve	**Order**
UMDA	$-4.3384+1.2883 \cdot n$	$O(n)$
TREE	$-3.8817+1.8424 \cdot n$	$O(n)$
EBNA$_{BIC}$	$1.7008+1.5333 \cdot n$	$O(n)$

Figure 8.10. Fitting curves for the different EDAs in the optimization of the pseudo-modular function and their order.

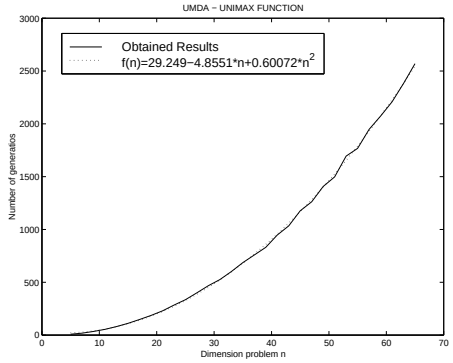


Figure 8.11. UMDA - Unimax F.

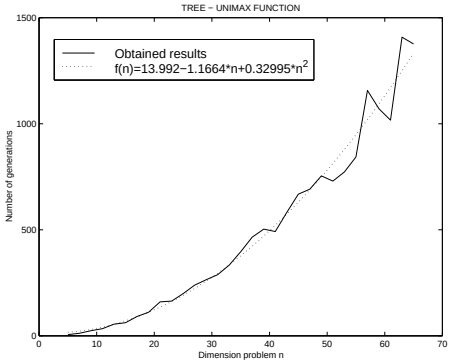


Figure 8.12. TREE - Unimax F.

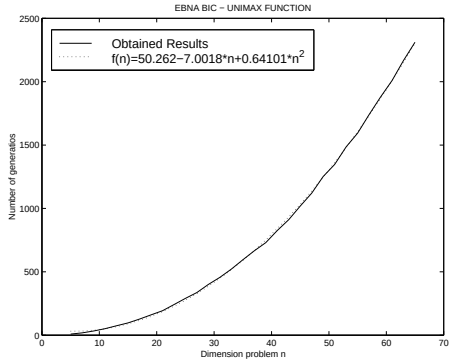


Figure 8.13. EBNA$_{BIC}$ - Unimax F.

	Fitting Curve	**Order**
UMDA	$29.249-4.8551 \cdot n+0.67002 \cdot n^2$	$O(n^2)$
TREE	$13.999-1.1664 \cdot n+0.32995 \cdot n^2$	$O(n^2)$
EBNA$_{BIC}$	$50.262-7.0018 \cdot n+0.64101 \cdot n^2$	$O(n^2)$

Figure 8.14. Fitting curves for the different EDAs in the optimization of the unimax function and their order.

Part IV

Conclusions

Chapter 9

Conclusions

This part summarizes the main contributions and conclusions of this dissertation regarding the theoretical study of EDAs, and points to areas where further research is required.

9.1 Contributions

This thesis has been concerned with helping to close the gap between the vast amounts of empirical observations on EDAs and the sparse number of theoretical results from their study reported in the literature.

Specifically, this dissertation has presented the following items and conclusions as novel contributions to the theoretical study of EDAs:

- Building of a general analytic framework based on Markov chains for analyzing the convergence of a general EDA. Proving of a new general convergence theorem for EDAs using Markov chains. Analysis of the most common discrete EDAs, by means of the application of this theorem, resulting in convergence and non-convergence algorithms. For those algorithms that do not converge, some conditions have been imposed on the parameters of their probability distributions to guarantee convergence.

- Analysis of PBIL based on Markov chains making it possible to study the behavior of PBIL applied to the $OneMax$ function when $M = 2$ (population size) and N=1 (number of selected individuals). This enables us to prove that:

$$P\left(\lim_{t\to\infty} \mathbf{p}_t = (a,b)\right) \longrightarrow 1$$

  when $\alpha \to 1$, $\mathbf{p}_0 \to (a,b)$, and $(a,b) \in \{0,1\}^2$. This fact implies a strong dependence of PBIL convergence on the initial vector $\mathbf{p}_0$ and on the α parameter value. It is also important to note that although when the value of α is small the experimental results carried out seem to be more stable, it can not be concluded that PBIL converges to the optimum.

- Analysis of PBIL based on discrete dynamical systems. A discrete dynamical system is associated with the PBIL algorithm. It is proved that the behavior of the PBIL algorithm follows the iterates of the discrete dynamical system for a long time when the parameter α is near zero. It is proved – performing a stability analysis – that all the points of the search space are fixed points of the

dynamical system, and that the local optimum points for the function to optimize coincide with the stable fixed points. Hence it can be deduced that the PBIL algorithm converges to the global optimum in unimodal functions.

- Analysis of the behavior of UMDA in some pseudo-boolean functions based on Markov chains. Calculation of the absorption probability to the optimum and the expected absorption times for UMDA on the maximization of linear, pseudo-modular, unimax and almost positive functions, for different values of selected population size N and individual length n. This information is used to provide some insights into how the absorption probability to the optimum and the expected absorption times evolve when the size of population increases. The results show the different behaviors of the algorithm in the analyzed functions and the behavior of the algorithm when the complexity of the function increases: the absorption probability decreases while the expected absorption time increases. Even in the almost positive function the absorption probability to the optimum approaches zero as N increases.

- Modeling of UMDA with infinite population and proportionate selection using discrete dynamical systems, arriving at the same expression as obtained by Mühlenbein and Mahnig (2002a).

- Analysis and modeling of UMDA$_c$ with tournament selection applied to n-dimensional linear and quadratic functions when an infinite number of tournaments is performed. Linear functions are used to model the algorithm when far from the optimum, while quadratic function is used to analyze the algorithm when near the optimum. In the case of linear functions it is concluded that the algorithm does not work correctly in linear function $L_1(\mathbf{x}) = \sum_{i=1}^{n} x_i$, with $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. After making certain assumptions in the case of quadratic function $Q(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$, it is proved for some finite dimensions that the algorithm reaches the optimum. Moreover, the speed of convergence becomes slower as dimension increases.

- Analysis of time complexity of EDAs based on a framework built on the absorbing Markov chain model of EDAs. For some instances of EDAs – UMDA, MIMIC, TREE and EBNA$_{BIC}$ – it is proved that their worst-case first hitting time is exponential (in individual size) when they are used to maximize any pseudo-boolean injective function. The worst-case analysis carried out is useful for discussing how to find bounds in the average-case analysis.

- Based on empirical results, new information is provided on average time complexity of some EDAs applied to a number of objective functions. The main conclusions after the experiments are: (i) The average optimization time is not related to the complexity of the probabilistic model used by the algorithm. Greater complexity of the probabilistic model of the EDA does not imply greater efficiency (low order of the first hitting time); (ii) EDAs whose probabilistic model reflects the relationship between the variables of the problem do not obtain a better order of the first hitting time than those that do not.

9.2 Future Work

EDAs are being used more and more in a wide variety of applications. Sometimes they are very successful, and sometimes they are not. We still have no clear understanding of why they work when they do and why they sometimes fail. This is why theoretical research is so important, so that we can begin to understand what it is we are doing and how we can do it better. However, while it is relatively easy to invent yet another variant of an EDA, it can be much harder to analyze it. But

unless we have a formal understanding of these algorithms, their development is largely going to be based on guesswork.

There is much further work to be done in the theoretical analysis of EDAs. Here we propose some lines of further research.

As a first step, if we want to have a formal way of comparing the efficiency of EDAs with and between other algorithms, research on the time complexity of EDAs is of special importance, in particular in the first hitting time in the average-case. The contributions presented here concerning the study of the worst-case first hitting time of EDAs give a possible route for future investigations. As we explain in Chapter 8, to find a polynomial bound for the first hitting time in the average-case of some EDAs it is necessary to find a polynomial bound for the number of those populations from which the first hitting time is exponential. This is an exceedingly difficult task because in principle the entire information about the evolutionary process in an EDA is implicitly contained in the joint probability distribution of the selected individuals, and to extract sufficient and appropriate information out of it in order to predict finite time behavior is hard work.

In order to contribute to the question of whether EDAs which exploit the structure of the objective function perform better than those that do not, another line of approach could be to find mathematical models which enable us to analyze the relationship between the probabilistic models that EDAs use and the characteristics of the function to optimize.

An ambitious and interesting task could be to find a general and unified analytic framework which makes it possible to analyze different EDAs for different problems and for convergence and computation time. This would be a step towards a systematic comparative study among different EDAs. Such a framework should also serve to find a classification of easy and hard problems for EDAs. A future challenge to gain knowledge would be to be able to rank the different classes of problems according to how hard they are to be optimized by each EDA. In other words, we would want to categorize the problems in groups $P_1, \ldots, P_l$, in such a way that given an instance of EDA, A, the average-time complexity of A applied to problem P_i would be τ_i, with $i = 1, \ldots, l$.

Appendix A
Spherical Change of Variable in Dimension n

This appendix explains in detail the generalization to n dimensions of the spherical change of variable in dimension three.

The first step is to solve the problem for $n = 4$. In spherical coordinates the position of a point $P(x_1, x_2, x_3, x_4)$ in the space is determined by four numbers ρ, α_1, α_2, α_3, where:

- ρ is the distance from point P to the origin.
- α_3 is the angle formed by the vector $\overline{OP}$ and its projection (denoted by $\overline{r}_1$) upon the plane $\overline{OX_1X_2X_3}$.
- α_2 is the angle formed by the projection of $\overline{r}_1$ (denoted by $\overline{r}_2$) upon the plain $\overline{OX_1X_2}$.
- α_1 is the angle formed by axis X_1 and $\overline{r}_2$.

Taking these facts into account, we can write the old coordinates depending on the new ones:

$$x_4 = \rho \sin \alpha_3 \tag{A.1}$$
$$x_3 = \rho \cos \alpha_3 \sin \alpha_2 \tag{A.2}$$
$$x_2 = \rho \cos \alpha_3 \cos \alpha_2 \sin \alpha_1 \tag{A.3}$$
$$x_1 = \rho \cos \alpha_3 \cos \alpha_2 \cos \alpha_1 \tag{A.4}$$
$$\tag{A.5}$$

For any point $P(x_1, x_2, x_3, x_4) \in I\!R^4$, each new variable varies in:

$$0 \leq \rho \leq \infty \tag{A.6}$$
$$-\pi/2 \leq \alpha_2, \alpha_3 \leq \pi/2 \tag{A.7}$$
$$0 \leq \alpha_1 \leq 2\pi \ . \tag{A.8}$$

Jacobian of the change J_4 is:

$$J_4 = \rho \cos \alpha_2 \cos^2 \alpha_3. \tag{A.9}$$

Therefore after making this change of variable in the integral:

$$\int\int\int\int_{I\!R^4} f(x_1, x_2, x_3, x_4) dx_1 dx_2 dx_3 dx_4, \tag{A.10}$$

we obtain:

$$\int_0^{2\pi} \int_{-\pi/2}^{\pi/2} \int_{-\pi/2}^{\pi/2} \int_0^{\infty} f(\rho, \alpha_3, \alpha_2, \alpha_1) \cdot \rho \cos \alpha_2 \cos^2 \alpha_3 \cdot d\alpha_1 d\alpha_2 d\alpha_3 d\rho. \tag{A.11}$$

Taking into account the above arguments, we can say that using the generalization to n dimensions of the spherical change of variable in dimension four, the integral

$$I_n^S = \int \dots \int_{I\!R^n} f(x_1, \dots, x_n) dx_1 \dots dx_n, \tag{A.12}$$

changes to:

$$\begin{aligned} & I_n^S \\ & = \int_0^{2\pi} \int_{-\pi/2}^{\pi/2} \dots \int_{-\pi/2}^{\pi/2} \int_0^{\infty} f(\rho, \alpha_{n-1}, \dots, \alpha_1) \rho^{n-1} \cos\alpha_2 \cdot \cos^2\alpha_3 \cdot \\ & \quad \cdot \cos^{n-3}\alpha_{n-2} \cdot \dots \cdot \cos^{n-2}\alpha_{n-1} d\alpha_1 d\alpha_2 d\alpha_3 \dots d\alpha_{n-2} d\alpha_{n-3} d\alpha_{n-1} d\rho. \end{aligned}$$

In the integrals we have to solve in this dissertation using this change of variable, function f only depends on ρ ($f(\rho)$). Therefore, the above integral can be written as follows:

$$\begin{aligned} I_n^S = & \int_0^{2\pi} d\alpha_1 \int_{-\pi/2}^{\pi/2} \cos\alpha_2 d\alpha_2 \int_{-\pi/2}^{\pi/2} \cos^2\alpha_3 d\alpha_3 \cdot \dots \\ & \dots \cdot \int_{-\pi/2}^{\pi/2} \cos^{n-3}\alpha_{n-2} d\alpha_{n-2} \int_{-\pi/2}^{\pi/2} \cos^{n-2}\alpha_{n-1} d\alpha_{n-1} \int_0^{\infty} \rho^{n-1} f(\rho) d\rho \\ = & \ 2\pi \prod_{i=1}^{n-2} \int_{-\pi/2}^{\pi/2} \cos^i\beta d\beta \int_0^{\infty} \rho^{n-1} f(\rho) d\rho \ . \end{aligned} \tag{A.13}$$

To solve this integral it will be useful to calculate the constant:

$$S(n) = 2\pi \prod_{i=1}^{n-2} \int_{-\pi/2}^{\pi/2} \cos^i\beta d\beta = 2\pi \prod_{i=1}^{n-2} \sqrt{\pi} \frac{\Gamma\left(\frac{i+1}{2}\right)}{\Gamma\left(1+\frac{i}{2}\right)} \ , \tag{A.14}$$

where Γ is the Gamma function. Simplifying:

$$S(n) = \frac{2\pi(\sqrt{\pi})^{n-2}}{\Gamma\left(\frac{n}{2}\right)} \ . \tag{A.15}$$

Appendix B
One possible final structure after $n-1$ steps of Algorithm B

Our aim here is to prove that one possible final structure after $n-1$ steps of Algorithm B – denoted by S_{n-1} – is as follows:

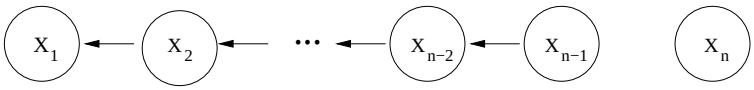


Figure B.1. One possible final structure S_{n-1}.

In order to prove that we carry out the schema:

i) First we prove by induction on k that at the k-th step of Algorithm B, with $k = 1, \ldots, n-2$, the arc that goes from variable X_{k+1} to variable X_k is added.

ii) After this we prove that at the $n-1$-th step of Algorithm B no arc is added.

Throughout the demonstration it will be necessary to compare the BIC scores associated with the different structures we have to choose from. Because these BIC scores depend on two parameters, N and n, in order to make this comparison easy at the moment of the choice, we will assume that N depends linearly on n.

First, part (**i**) is proved for $k = 1$. To this end we have to calculate the score associated with the arc-less structure, which we denote by S_0, as well as the score associated with the structure with an arc and choose the case with the maximum associated score. Given that:

$$BIC(S_0, i^*) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} \mathsf{P}(X_i = x_{w,i}^* | \mathbf{pa}_i^{S_0}, \boldsymbol{\theta}_i) - \frac{1}{2} \log N \sum_{i=1}^{n} q_i (r_i - 1) \ ,$$

and taking into account that no variable has a parent we find that:

$$BIC(S_0, i^*) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} \mathsf{P}(X_i = x_{w,i}^*) - \frac{1}{2} n \log N \ .$$

The expression for the BIC score in this case is:

$$BIC(S_0, i^*)$$
$$= \log \left[\left[(\mathsf{P}(X_1 = 0))^{N-1} \cdot \mathsf{P}(X_1 = 1) \right] \cdot \ldots \cdot \left[(\mathsf{P}(X_{n-1} = 0))^{N-1} \cdot \mathsf{P}(X_{n-1} = 1) \right] (\mathsf{P}(X_n = 0))^N \right] - \frac{1}{2} n \log N.$$

Given that the parameters of the Bayesian network are calculated using expression (8.18), we find that:

$$BIC(S_0, i^*)=\log\left[\left(\frac{N}{N+2}\right)^{(N-1)(n-1)}\cdot\left(\frac{2}{N+2}\right)^{(n-1)}\cdot\left(\frac{N+1}{N+2}\right)^{N}\cdot\left(\frac{1}{N^{n/2}}\right)\right].$$

Now we have to calculate the score for the structure when an arc is added (we denote this kind of structure by S_1). There are three kinds of arcs to add (classified by the associated BIC score):

1 Arcs that go from variable X_i to variable X_j, with $i, j = 1, \ldots, n-1$ and $i \neq j$. The score of this kind of structures will be denoted by $BIC(S_1^1, i^*)$.

2 Arcs that go from variable X_n to variable X_i, with $i = 1, \ldots, n-1$. The score of this kind of structures will be denoted by $BIC(S_1^2, i^*)$.

3 Arcs that go from variable X_i, with $i = 1, \ldots, n-1$ to variable X_n. The score of this kind of structures will be denoted by $BIC(S_1^3, i^*)$.

The quantity $BIC(S_1^1, i^*)$ is equal for the $(n-1)(n-2)$ possible arcs of kind 1, so it is sufficient to study the situation in which the structure S_1^1 is the structure resulting from adding the arc that goes from variable X_2 to variable X_1. Given that only variable X_1 has as parent variable X_2, and that no other variables have a parent, we find that:

$$\begin{aligned}
BIC(S_1^1, i^*) &= \log\left[\prod_{w=1}^{N}\mathsf{P}(X_1 = x_{w,1}^*|\mathbf{pa}_1^{S_1^1}, \boldsymbol{\theta}_1)\prod_{w=1}^{N}\prod_{i=2}^{n}\mathsf{P}(X_i = x_{w,i}^*)\right]-\frac{1}{2}(n+1)\log N \\
&= \log\left[\left[(\mathsf{P}(X_1 = 0|X_2 = 0))^{N-1}\cdot\mathsf{P}(X_1 = 1|X_2 = 1)\right]\cdot\right. \\
&\cdot\left[(\mathsf{P}(X_2 = 0))^{N-1}\cdot\mathsf{P}(X_2 = 1)\right]\cdot\ldots\cdot\left[(\mathsf{P}(X_{n-1} = 0))^{N-1}\cdot\mathsf{P}(X_{n-1} = 1)\right]\cdot(\mathsf{P}(X_n = 0))^{N}\Big]-\frac{1}{2}(n+1)\log N.
\end{aligned}$$

We carry out the estimation of probabilities as before, which yields that the BIC score for structure S_1^1 can be written as:

$$BIC(S_1^1, i^*) = \log\left[\left(\frac{N}{N+1}\right)^{(N-1)}\cdot\frac{2}{3}\cdot\left(\frac{N}{N+2}\right)^{(N-1)(n-2)}\cdot\left(\frac{2}{N+2}\right)^{n-2}\cdot\left(\frac{N+1}{N+2}\right)^{N}\cdot\frac{1}{N^{(n+1)/2}}\right].$$

Next we calculate $BIC(S_1^2, i^*)$. As in the previous case, we calculate this quantity only for one of the $n-1$ possible cases. Here we study the case in which S_1^2 is the structure resulting from adding the arc that goes from variable X_n to variable X_{n-1}. Because only variable X_{n-1} has as parent variable X_n, and no other variable has a parent we find that:

$$\begin{aligned}
&BIC(S_1^2, i^*) \\
&= \log\left[\prod_{w=1}^{N}\prod_{i=1}^{n-2}\mathsf{P}(X_i = x_{w,i}^*)\prod_{w=1}^{N}\mathsf{P}(X_{n-1} = x_{w,n-1}^*|\mathbf{pa}_{n-1}^{S_1^2}, \boldsymbol{\theta}_{n-1})\prod_{w=1}^{n}\mathsf{P}(X_n = x_{w,n}^*)\right]-\frac{1}{2}(n+1)\log N \\
&= \log\left[\left[(\mathsf{P}(X_1 = 0))^{N-1}\cdot\mathsf{P}(X_1 = 1)\right]\cdot\ldots\cdot\left[(\mathsf{P}(X_{n-2} = 0))^{N-1}\cdot\mathsf{P}(X_{n-2} = 1)\right]\cdot\right. \\
&\cdot\quad\left.\left[(\mathsf{P}(X_{n-1} = 0|X_n = 0))^{N-1}\cdot\mathsf{P}(X_{n-1} = 1|X_n = 0)\right]\cdot(\mathsf{P}(X_n = 0))^{N}\right]-\frac{1}{2}(n+1)\log N\ .
\end{aligned}$$

Estimating the probabilities as above, the BIC score for structure S_1^2 can be written as:

$$BIC(S_1^2, i^*) = \log\left[\left(\frac{N}{N+2}\right)^{(N-1)(n-1)} \cdot \left(\frac{2}{N+2}\right)^{(n-1)} \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \left(\frac{1}{N^{(n+1)/2}}\right)\right] .$$

Finally, for the calculation of $BIC(S_1^3, i^*)$, we use the structure S_1^3 resulting from the addition of the arc that goes from variable X_{n-1} to variable X_n. Taking into account that only variable X_n has as parent variable X_{n-1}, and that no other variable has a parent, we see that:

$$\begin{aligned} & BIC(S_1^3, i^*) \\ &= \log\left[\prod_{w=1}^{N}\prod_{i=1}^{n-1} \mathsf{P}(X_i = x_{w,i}^*) \prod_{w=1}^{N} \mathsf{P}(X_n = x_{w,n}^* | \mathbf{pa}_n^{S_1^3}, \boldsymbol{\theta}_n)\right] - \frac{1}{2}(n+1)\log N \\ &= \log\Big[\big[(\mathsf{P}(X_1=0))^{N-1}\cdot\mathsf{P}(X_1=1)\big]\cdot\ldots\cdot\big[(\mathsf{P}(X_{n-1}=0))^{N-1}\cdot\mathsf{P}(X_{n-1}=1)\big]\cdot \\ &\cdot \quad \big[(\mathsf{P}(X_n=0|X_{n-1}=0))^{N-1}\cdot\mathsf{P}(X_n=0|X_{n-1}=1)\big]\Big] - \frac{1}{2}(n+1)\log N \ . \end{aligned}$$

Now we estimate the probabilities as before, which yields that the BIC score for structure S_1^3 can be written as:

$$BIC(S_1^3, i^*) = \log\left[\left(\frac{N}{N+2}\right)^{(N-1)(n-1)} \cdot \left(\frac{2}{N+2}\right)^{(n-1)} \cdot \left(\frac{N}{N+1}\right)^{(N-1)} \cdot \frac{2}{3} \cdot \left(\frac{1}{N^{(n+1)/2}}\right)\right].$$

To carry out the comparison between the scores associated with structures S_0, S_1^1, S_1^2 and S_1^3 we take into account that N depends linearly on n. It can be seen that $BIC(S_1^1, i^*) > BIC(S_0, i^*), BIC(S_1^2, i^*), BIC(S_1^3, i^*)$. Therefore we have proved part (**i**) for $k = 1$.

By the inductive hypothesis, we assume that part (**i**) is true for k, $k = 1, \ldots, n-3$. Let us now see whether it is true for $k+1$.

By the inductive hypothesis we know that at the k-th step of Algorithm B, with $k = 1, \ldots, n-3$, the arc that goes from variable X_{k+1} to variable X_k has been added (we denote by S_k^1 the resulting structure). Hence the biggest score in this case is the score associated with structure S_k^1. Since variables $X_1, \ldots, X_k$ have one parent and the remaining variables have no parents, we find that:

$$BIC(S_k^1, i^*) = \log\left[\prod_{w=1}^{N}\prod_{i=1}^{k} \mathsf{P}(X_i = x_{w,i}^* | \mathbf{pa}_i^{S_k^1}, \boldsymbol{\theta}_i) \prod_{w=1}^{N}\prod_{i=k+1}^{n} \mathsf{P}(X_i = x_{w,i}^*)\right] - \frac{1}{2}(n+k)\log N.$$

Estimating the probabilities as before:

$$BIC(S_k^1, i^*) = \log\left[\left[\left(\frac{N}{N+1}\right)^{(N-1)} \cdot \frac{2}{3}\right]^{k} \cdot \left[\left(\frac{N}{N+2}\right)^{(N-1)} \cdot \frac{2}{N+2}\right]^{n-k-1} \cdot \left(\frac{N+1}{N+2}\right)^{N}\right] - \frac{1}{2}(n+k)\log N.$$

Therefore, the BIC score for structure S_k^1 can be written as:

$$BIC(S_k^1, i^*) = \log\left[\left(\frac{N}{N+1}\right)^{k(N-1)} \cdot \left(\frac{2}{3}\right)^{k} \cdot \left(\frac{N}{N+2}\right)^{(N-1)(n-k-1)} \cdot \left(\frac{2}{N+2}\right)^{n-k-1} \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \frac{1}{N^{(n+k)/2}}\right].$$

Now he have to calculate the score for the structure after $k+1$ steps of Algorithm B (this structure is denoted by S_{k+1}), and choose the bigger one between $BIC(S_k^1, i^*)$, and $BIC(S_{k+1}, i^*)$. At this step of the algorithm there are four kinds of arcs to add (classified by the associated BIC score):

1 Arcs that go from variable X_i to variable X_j, with $i = 1, \ldots, n-1$, $j = k+1, \ldots, n-1$, and $i \neq j$. The score of this kind of structures will be denoted by $BIC(S^1_{k+1}, i^*)$.

2 Arcs that go from variable X_n to variable X_i, with $i = k+1, \ldots, n-1$. The score of this kind of structures will be denoted by $BIC(S^2_{k+1}, i^*)$.

3 Arcs that go from variable X_i, with $i = 1, \ldots, n-1$ to variable X_n. The score of this kind of structures will be denoted by $BIC(S^3_{k+1}, i^*)$.

4 Arcs that go from variable X_i to variable X_j, with $i = k+1, \ldots, n$ and $j = 1, \ldots, k$. The score of this kind of structures will be denoted by $BIC(S^4_{k+1}, i^*)$.

Given that the quantity $BIC(S^1_{k+1}, i^*)$ is equal for the $(n-k-1)(n-2)$ possible arcs of kind 1, it is sufficient to calculate this quantity when the structure S^1_{k+1} is the structure resulting from adding the arc that goes from variable X_{k+2} to variable X_{k+1}. Taking into account that the first $k+1$ variables have one parent, and the remaining variables have no parents, we see that:

$$BIC(S^1_{k+1}, i^*) = log\left[\prod_{w=1}^{N}\prod_{i=1}^{k+1} \mathsf{P}(X_i = x^*_{w,i}|\mathbf{pa}_i^{S^1_{k+1}}, \boldsymbol{\theta}_i) \prod_{w=1}^{N}\prod_{i=k+2}^{n} \mathsf{P}(X_i = x^*_{w,i})\right] - \frac{1}{2}(n+k+1)\log N.$$

If we estimate the parameters as in previous cases, we find that the BIC score for structure S^1_{k+1} can be written as:

$$\begin{aligned} &BIC(S^1_{k+1}, i^*) \\ &= \log\left[\left(\frac{N}{N+1}\right)^{(N-1)(k+1)} \cdot \left(\frac{2}{3}\right)^{k+1} \cdot \left(\frac{N}{N+2}\right)^{(N-1)(n-k-2)} \cdot \left(\frac{2}{N+2}\right)^{n-k-2} \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \frac{1}{N^{(n+k+1)/2}}\right]. \end{aligned} \tag{B.1}$$

Next we calculate $BIC(S^2_{k+1}, i^*)$. Given that the score is equal for the $n-k-1$ possible structures of kind 2 we calculate it only when S^2_{k+1} is the structure resulting from adding the arc that goes from variable X_n to variable X_{n-1}. Because variable $X_1, \ldots, X_k$ and variable X_{n-1} have one parent and the remaining variables have no parents, we find that:

$$\begin{aligned} &BIC(S^2_{k+1}, i^*) \\ &= \log\left[\prod_{w=1}^{N}\prod_{i=1}^{k} \mathsf{P}(X_i = x^*_{w,i}|\mathbf{pa}_i^{S^2_{k+1}}, \boldsymbol{\theta}_i) \prod_{w=1}^{N}\prod_{i=k+1}^{n-2} \mathsf{P}(X_i = x^*_{w,i}) \right. \\ &\quad \left. \prod_{w=1}^{N} \mathsf{P}(X_{n-1} = x^*_{w,n-1}|\mathbf{pa}_{n-1}^{S^2_{k+1}}, \boldsymbol{\theta}_{n-1}) \prod_{w=1}^{N} \mathsf{P}(X_n = x^*_{w,n})\right] - \frac{1}{2}(n+k+1)\log N \ . \end{aligned}$$

Taking into account that the parameters are estimated as above, the BIC score for structure S^2_{k+1} can be expressed as:

$$\begin{aligned} &BIC(S^2_{k+1}, i^*) \\ &= \log\left[\left(\frac{N}{N+1}\right)^{(N-1)k} \cdot \left(\frac{2}{3}\right)^{k} \cdot \left(\frac{N}{N+2}\right)^{(N-1)(n-k-1)} \cdot \left(\frac{2}{N+2}\right)^{n-k-1} \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \left(\frac{1}{N^{(n+k+1)/2}}\right)\right]. \end{aligned} \tag{B.2}$$

The calculation of $BIC(S^3_{k+1}, i^*)$ is carried out using the structure S^3_{k+1} where the arc that goes from variable X_{n-1} to variable X_n is added. In this case variables $X_1, X_2, \ldots, X_k$ have one parent, and the rest have none, which yields that:

$$BIC(S^3_{k+1}, i^*)$$
$$= \log\left[\prod_{w=1}^{N}\prod_{i=1}^{k}\mathsf{P}(X_i = x^*_{w,i}|\mathbf{pa}_i^{S^3_{k+1}}, \boldsymbol{\theta}_i)\prod_{w=1}^{N}\prod_{i=k+1}^{n-1}\mathsf{P}(X_i = x^*_{w,i})\prod_{w=1}^{N}\mathsf{P}(X_n = x^*_{w,n}|\mathbf{pa}_n^{S^3_{k+1}}, \boldsymbol{\theta}_n)\right] - \frac{1}{2}(n+1)\log N.$$

Given that the parameters are estimated as in the previous cases:

$$\begin{aligned} BIC(S^3_{k+1}, i^*) &= \log\left[\left(\frac{N}{N+1}\right)^{(N-1)k}\cdot\left(\frac{2}{3}\right)^{k}\cdot\left(\frac{N}{N+2}\right)^{(N-1)(n-k-1)}\cdot\left(\frac{2}{N+2}\right)^{(n-k-1)}\cdot\right. \\ &\cdot \left.\left(\frac{N}{N+1}\right)^{(N-1)}\cdot\frac{2}{3}\cdot\left(\frac{1}{N^{(n+k+1)/2}}\right)\right]. \end{aligned} \tag{B.3}$$

For the calculation of $BIC(S^4_{k+1}, i^*)$, we use the structure S^4_{k+1} resulting from adding the arc that goes from variable X_{k+2} to variable X_1. It follows from variables $X_1, X_2, \ldots, X_k$ having one parent and the rest having none, that:

$$BIC(S^4_{k+1}, i^*) = \log\left[\prod_{w=1}^{N}\prod_{i=1}^{k}\mathsf{P}(X_i = x^*_{w,i}|\mathbf{pa}_i^{S^4_{k+1}}, \boldsymbol{\theta}_i)\prod_{w=1}^{N}\prod_{i=k+1}^{n}\mathsf{P}(X_i = x^*_{w,i})\right] - \frac{1}{2}(n+1)\log N\ .$$

Consequently, estimating the probabilities as above, the BIC score for structure S^4_{k+1} can be expressed as:

$$\begin{aligned} &BIC(S^4_{k+1}, i^*) \\ &= \log\left[\left(\frac{N}{N+1}\right)^{(N-1)k}\cdot\left(\frac{2}{3}\right)^{k}\cdot\left(\frac{N}{N+2}\right)^{(N-1)(n-k-1)}\cdot\left(\frac{2}{N+2}\right)^{n-k-1}\left(\frac{N+1}{N+2}\right)^{N}\cdot\left(\frac{1}{N^{(n+k)/2}}\right)\right]. \end{aligned} \tag{B.4}$$

Taking N linear in n makes it easier to compare the different scores. It can be seen that $BIC(S^1_{k+1}, i^*) > BIC(S_k, i^*)$, $BIC(S^2_{k+1}, i^*)$, $BIC(S^3_{k+1}, i^*)$, $BIC(S^4_{k+1}, i^*)$. Therefore we have proven part (**i**).

To finalize we have to demonstrate part (**ii**). In order to do so it is necessary to compare the score associated with structure S^1_{n-2} (note that (**i**) is true) and the score associated with the structure S_{n-1}.

The calculation of $BIC(S^1_{n-2}, i^*)$ can be carried out as follows:

$$BIC(S^1_{n-2}, i^*) = BIC(S^1_{k+1}, i^*),\ \text{for } k = n-3,$$

therefore, using (B.1) for $k = n-3$, we see that:

$$BIC(S^1_{n-2}, i^*) = \log\left[\left(\frac{N}{N+1}\right)^{(N-1)(n-2)}\cdot\left(\frac{2}{3}\right)^{n-2}\cdot\left(\frac{N}{N+2}\right)^{(N-1)}\cdot\left(\frac{2}{N+2}\right)\cdot\left(\frac{N+1}{N+2}\right)^{N}\cdot\frac{1}{N^{n-1}}\right].$$

Following the score for the structure after $n-1$ steps, S_{n-1} is calculated. After that the bigger of $BIC(S^1_{n-2}, i^*)$, and $BIC(S_{n-1}, i^*)$ is chosen. At this step of the algorithm there are again four kinds of arcs to add (classified by the associated BIC score):

1 Arcs that go from variable X_i, with $i = 1, \ldots, n-2$ to variable X_{n-1}. The score of this kind of structures will be denoted by $BIC(S^1_{n-1}, i^*)$.

2 The arc that goes from variable X_n to variable X_{n-1}. The score of this structure will be denoted by $BIC(S^2_{n-1}, i^*)$.

3 Arcs that go from variable X_i, with $i = 1, \ldots, n-1$ to variable X_n. The score of this kind of structures will be denoted by $BIC(S^3_{n-1}, i^*)$.

4 Arcs that go from variable X_i, with $i = n-1, n$ to variable X_j, with $j = 1, \ldots, n-2$. The score of this kind of structures will be denoted by $BIC(S^4_{n-1}, i^*)$.

The score associated with S^1_{n-1} can be calculated as follows:

$$BIC(S^1_{n-1}, i^*) = BIC(S^1_{k+1}, i^*), \text{ for } k = n-2. \tag{B.5}$$

Thus, taking into account expression (B.1) for $k = n-2$, we see that:

$$BIC(S^1_{n-1}, i^*) = \log\left[\left(\frac{N}{N+1}\right)^{(N-1)(n-1)} \cdot \left(\frac{2}{3}\right)^{n-1} \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \left(\frac{1}{N^{(2n-1)/2}}\right)\right].$$

Among the $n-2$ possible structures of kind S^1_{k+1} we choose the structure resulting from adding the arc that goes from variable X_1 to variable X_{n-1}.

To obtain the score associated with S^2_{n-1} we can use:

$$BIC(S^2_{n-1}, i^*) = BIC(S^2_{k+1}, i^*), \text{ for } k = n-2,$$

hence, after using (B.2), for $k = n-2$:

$$\begin{aligned} &BIC(S^2_{n-1}, i^*) \\ &= \log\left[\left(\frac{N}{N+1}\right)^{(N-1)(n-2)} \cdot \left(\frac{2}{3}\right)^{(n-2)} \cdot \left(\frac{N}{N+2}\right)^{(N-1)} \cdot \left(\frac{2}{N+2}\right) \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \left(\frac{1}{N^{(2n-1)/2}}\right)\right]. \end{aligned}$$

The calculation of the score associated with S^3_{n-1} can be carried out using:

$$BIC(S^3_{n-1}, i^*) = BIC(S^3_{k+1}, i^*), \text{ for } k = n-2.$$

Therefore, taking $k = n-2$ in expression (B.3) gives:

$$\begin{aligned} &BIC(S^3_{n-1}, i^*) \\ &= \log\left[\left(\frac{N}{N+1}\right)^{(N-1)(n-2)} \cdot \left(\frac{2}{3}\right)^{(n-2)} \cdot \left(\frac{N}{N+2}\right)^{(N-1)} \cdot \left(\frac{2}{N+2}\right) \cdot \left(\frac{N}{N+1}\right)^{(N-1)} \cdot \frac{2}{3} \cdot \left(\frac{1}{N^{(2n-1)/2}}\right)\right]. \end{aligned}$$

Finally, to calculate S^4_{n-1} we can use:

$$BIC(S^4_{n-1}, i^*) = BIC(S^4_{k+1}, i^*), \text{ for } k = n-2.$$

If we write expression (B.4) for $k = n-2$:

$$BIC(S^4_{n-1}, i^*) = \log\left[\left(\frac{N}{N+1}\right)^{(N-1)(n-2)} \cdot \left(\frac{2}{3}\right)^{(n-2)} \cdot \left(\frac{N}{N+2}\right)^{(N-1)} \cdot \left(\frac{2}{N+2}\right) \cdot \left(\frac{N+1}{N+2}\right)^{N} \cdot \left(\frac{1}{N^{(n-1)}}\right)\right].$$

As in the previous cases, to make the comparison among the scores easier, we assume that N depends linearly on n. Hence, it can be seen that $BIC(S^1_{n-1}, i^*) > BIC(S^2_{n-2}, i^*)$, $BIC(S^2_{n-1}, i^*)$, $BIC(S^3_{n-1}, i^*)$, $BIC(S^4_{n-1}, i^*)$. Therefore we have proved part (**ii**), i.e. one possible structure that can be found after $n-1$ steps of Algorithm B is the structure that can be seen in Figure B.1.

References

Ahn, C. W., Ramakrishna, R. S., and Goldberg, D. E. (2004). Real-Coded Bayesian Optimization Algorithm: Bringing the Strength of BOA into the Continuous World. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 840–851. Springer.

Ali, W. and Topchy, A. P. (2004). Memetic Optimization of Video Chain Designs. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 869–882. Springer.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

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.

Baluja, S. and Caruana, R. (1995). Removing the Genetics from the Standard Genetic Algorithm. In Prieditis, A. and Russell, S., editors, *Proceedings of the International Conference on Machine Learning*, pages 38–46. Morgan Kaufmann.

Baluja, S. and Davies, S. (1997a). Combining Multiple Optimization Runs with Optimal Dependency Trees. Technical Report CMU-CS-97-157, Carnegie Mellon University.

Baluja, S. and Davies, S. (1997b). Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of the Search Space. In Fisher, D. H., editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 30–38. Morgan Kaufmann.

Baluja, S. and Davies, S. (1998). Fast Probabilistic Modeling for Combinatorial Optimization. In *AAAI-98*, pages 469–476.

Bandyopadhyay, S., Kargupta, H., and Wang, G. Revisiting the GEMGA: Scalable Evolutionary Optimization Through Linkage Learning. In *Proceedings of the Fifth IEEE International Conference on Evolutionary Computation*.

Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A., and Boeres, C. (2002). Learning and Simulation of Bayesian Networks Applied to Inexact Graph Matching. *Pattern Recognition*, 35(12):2867–2880.

Berny, A. (2000a). An Adaptive Scheme for Real Function Optimization Acting as a Selection Operator. In Yao, X., editor, *Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 140–149.

Berny, A. (2000b). Selection and Reinforcement Learning for Combinatorial Optimization. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Proceedings*, volume 1917 of *Lecture Notes in Computer Science*, pages 601–610. Springer.

Beyer, H. G. (2001). *The Theory of Evolution Strategies*. Springer.

Billingsley, P. (1986). *Probability and Measure*. John Willey and Sons.

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.

Bosman, P. A. N. and de Jong, E. D. (2004). Learning Probabilistic Tree Grammars for Genetic Programming. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 192–201. Springer.

Bosman, P. A. N. and Thierens, D. (1999). Linkage Information Processing in Distribution Estimation Algorithms. In Banzhaf, W., Daida, J. M., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M. J., and Smith, R. E., editors, *Genetic and Evolutionary Computation - GECCO 1999, Proceedings*, pages 60–67. Morgan Kaufmann.

Bosman, P. A. N. and Thierens, D. (2000). Continuous Iterated Density Estimation Evolutionary Algorithms within the IDEA Framework. In Wu, A. S., editor, *Workshop Program at the Genetic and Evolutionary Computation Conference - GECCO 2000, Proceedings*, pages 197–200.

Bosman, P. A. N. and Thierens, D. (2002). Permutation Optimization by Iterated Estimation of Random Keys Marginal Product Factorizations. In Merelo, J. J., Adamidis, P., Beyer, H.-G., Martín, J. L. F.-V., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 331–340. Springer.

Buntine, W. (1991). Theory Refinement in Bayesian Networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann.

Cantú-Paz, E. (2003). Pruning Neural Networks with Distribution Estimation Algorithms. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part I*, volume 2723 of *Lecture Notes in Computer Science*, pages 790–800. Springer.

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

Cesar, R. M., Bengoetxea, E., Bloch, I., and Larrañaga, P. (2005). Inexact Graph Matching for Model-Based Recognition: Evaluation and Comparison of Optimization Algorithms. *Pattern Recognition*, 38(11):2099–2113.

Cestnik, B. (1990). Estimating Probabilities: A Crucial Task in Machine Learning. In *Proceedings of the European Conference in Artificial Intelligence*, pages 147–149.

Cho, D.-Y. and Zhang, B.-T. (2004). Evolutionary Continuous Optimization by Distribution Estimation with Variational Bayesian Independent Component Analyzers Mixture Model. In Yao,

X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 212–221. Springer.

Chow, C. and Liu, C. (1968). Approximating Discrete Probability Distributions with Dependency Trees. *IEEE Transactions on Information Theory*, (14):462–467.

Cooper, G. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347.

Cotta, C. and Moscato, P. (2003). A Mixed-Evolutionary Statistical Analysis of an Algorithm's Complexity. *Applied Mathematics Letters*, 16(1):41–47.

Crama, Y. (1999). Recognition Problems for Special Classes of Polynomials in 0-1 Variables. *Mathematical Programming*, (44):139–155.

Crama, Y., Hansen, P., and Jaumard, B. (1990). The Basic Algorithm for Pseudo-boolean Programming Revisited. *Discrete Applied Mathematics*, (29):171–185.

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

David, H. A. (1970). *Order Statistics*. New York: Wiley.

de Bonet, J. S., Isbell, C. L., and Viola, P. (1997). MIMIC: Finding Optima by Estimating Probability Densities. *Advances in Neural Information Processing Systems*, 9:424.

De Jong, K. A., Spears, W. M., and Gordon, D. F. (1995). Using Markov Chains to Analyze GAFOs. In Whitley, L. D. and Vose, M., editors, *Proceedings of the 3rd Workshop on Foundations of Genetic Algorithms FOGA 3*, pages 115–135. Morgan Kaufmann.

de la Ossa, L., Gámez, J. A., and Puerta, J. M. (2004). Migration of Probability Models Instead of Individuals: An Alternative when Applying the Island Model to EDAs. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 242–252. Springer.

DeGroot, M. H. (1987). *Probability and Statistics*. Addison-Wesley.

Droste, S., Jansen, T., and Wegener, I. (1998). A Rigorous Complexity Analysis of the (1+1) Evolutionary Algorithm for Linear Functions for Boolean Inputs. *Evolutionary Computation*, 6(2):185–196.

Droste, S., Jansen, T., and Wegener, I. (2002). On the Analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science*, (276):51–81.

Eiben, A. E. and Rudolph, G. (1999). Theory of Evolutionary Algorithms: A Bird's Eye View. *Theoretical Computer Science*, 1(229):3–9.

Etxeberria, R. and Larrañaga, P. (1999). Global Optimization with Bayesian Networks. In *Proceedings of II Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization*, pages 332–339.

Fogel, D. B. (1994). An Introduction to Evolutionary Computation. *IEEE Transactions on Neural Networks*, 5(1):3–14.

Fogel, D. B. (1998). *Evolutionary Computation. The Fossil Record*. IEEE Press.

Fogel, L. J. (1962). Autonomous Automata. *Industrial Research*, 4:14–19.

Fogel, L. J. (1964). *On the Organization of Intellect*. Doctoral Dissertation. University of California, Los Angeles, CA.

Gallagher, M. and Frean, M. (2005). Population-Based Continuous Optimization, Probabilistic Modelling and Mean Shift. *Evolutionary Computation*, 13(1):29–42.

Gao, Y. and Culberson, J. (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.

Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. R. (1993). Rapid Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64. Morgan Kaufmann.

González, C., Lozano, J. A., and Larrañaga, P. (1999). Algoritmo PBIL: Un Análisis Teórico Preliminar. In *Proceedings of VIII Conferencia de la Asociación Española para la Inteligencia Artificial*, pages 81–88.

González, C., Lozano, J. A., and Larrañaga, P. (2000). Analyzing the Population Based Incremental Learning Algorithm by Means of Discrete Dynamical Systems. *Complex Systems*, 12(4):465–479.

González, C., Lozano, J. A., and Larrañaga, P. (2001). The Convergence Behavior of the PBIL Algorithm: A Preliminary Approach. In Kurková, V., Steele, N. C., Neruda, R., and Kárný, M., editors, *Proceedings of the Fifth International Conference on Artificial Neural Networks and Genetic Algorithms, ICANNGA 2001*, pages 228–231.

González, C., Lozano, J. A., and Larrañaga, P. (2002a). Mathematical Modelling of Discrete Estimation of Distribution Algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 147–163. Kluwer Academic Publishers.

González, C., Lozano, J. A., and Larrañaga, P. (2002b). Mathematical Modelling of UMDA$_c$ Algorithm with Tournament Selection. Behaviour on Linear and Quadratic Functions. *International Journal of Approximate Reasoning*, 31:313–340.

González, C., Lozano, J. A., and Larrañaga, P. (2002c). Modelado Matemático del Algoritmo UMDA$_c$ con Selección por Torneo Aplicado a Funciones Lineales. In *Proceedings of Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados, AEB*, pages 437–444.

González, C., Lozano, J. A., and Larrañaga, P. (2004). análisis del Peor Caso para el Tiempo Esperado por UMDA$_c$ en Alcanzar el óptimo por Primera Vez. In *Proceedings of Tercer Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB*, pages 262–269.

González, C., Ramírez, A., Lozano, J. A., and Larrañaga, P. (2005). Average Time Complexity of Estimation of Distribution Algorithms. In Cabestany, J., Prieto, A., and Hernández, F. S., editors, *IWANN*, volume 3512 of *Lecture Notes in Computer Science*, pages 42–49. Springer.

González, C., Rodríguez, J. D., Lozano, J. A., and Larrañaga, P. (2003). Analysis of the Univariate Marginal Distribution Algorithm Modeled by Markov Chains. In Mira, J. and Álvarez, J. R., editors, *IWANN (1)*, volume 2686 of *Lecture Notes in Computer Science*, pages 510–517. Springer.

Grahl, J. and Rothlauf, F. (2004). PolyEDA: Combining Estimation of Distribution Algorithms and Linear Inequality Constraints. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 1174–1185. Springer.

Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128.

Handa, H. (2003). Hybridization of Estimation of Distribution Algorithms with a Repair Method for Solving Constraint Satisfaction Problems. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part I*, volume 2723 of *Lecture Notes in Computer Science*, pages 991–1002. Springer.

Handa, H. (2004). Mutation Can Improve the Search Capability of Estimation of Distribution Algorithms. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 396–397. Springer.

Harik, G. (1999). Linkage Learning via Probabilistic Modeling in the EcGA. Technical Report 99010, IlliGAL Technical Report.

Harik, G., Lobo, F. G., and Golberg, D. E. (1998). The Compact Genetic Algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 523–528.

He, J. and Yao, X. (2001). Drift Analysis and average Time Complexity of Evolutionary Algorithms. *Artificial Intelligence*, 127:57–85.

He, J. and Yao, X. (2002). From an Individual to a Population: An Analysis of the First Hitting Time of Population-Based Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):495–511.

He, J. and Yao, X. (2003). Towards an Analytic Framework for analyzing the Computation Time of Evolutionary Algorithms. *Artificial Intelligence*, 145(1-2):59–97.

He, J. and Yao, X. (2004). A Study of Drift Analysis for Estimating Computation Time of Evolutionary Algorithms. *Natural Computing*, 3(1):21–35.

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. *Uncertainty in Artificial Intelligence*, 2:149–163.

Hiroyasu, T., Miki, M., Sano, M., Shimosaka, H., Tsutsui, S., and Dongarra, J. (2003). Distributed Probabilistic Model-Building Genetic Algorithm. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part I*, volume 2723 of *Lecture Notes in Computer Science*, pages 1015–1028. Springer.

Höhfeld, M. and Rudolph, G. (1997). Towards a Theory of Population-Based Incremental Learning. In *Proceedings of the Fourth IEEE International Conference on Evolutionary Computation*, pages 1–5. IEEE Press.

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

Horn, J., Goldberg, D. E., and Deb, K. (1994). Long Path Problems. In Davidor, Y., Schwefel, H. P., and Männer, R., editors, *Parallel Problem Solving from Nature - PPSN III, 3rd International Conference, Proceedings*, volume 866 of *Lecture Notes in Computer Science*, pages 149–158. Springer.

Iosifescu, M. (1980). *Finite Markov Processes and their Applications*. Wiley, Chichester.

Isaacson, D. L. and Madsen, R. W. (1985). *Markov Chains Theory and Applications*. Robert E. Krieger Publishing Company, INC.

Kargupta, H. (1996). The Gene Expression Messy Genetic Algorithm. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 631–636. IEEE Press.

Kargupta, H. and Goldberg, D. E. (1996). SEARCH, Blackbox Optimization and Sample Complexity. In *Proceedings of the 4th Workshop on Foundations of Genetic Algorithms FOGA 4*, pages 291–324. Morgan Kaufmann.

Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (1999). Optimization by Learning and Simulation of Bayesian and Gaussian Networks. Technical Report KZZA-IK-4-99, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (2000a). Combinatorial Optimization by Learning and Simulation of Bayesian Networks. In Boutilier, C. and Goldszmidt, M., editors, *Proceedings of Uncertainty in Artificial Intelligence, UAI-2000*, pages 343–352. Morgan Kaufmann.

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, *Workshop Program at the Genetic and Evolutionary Computation Conference - GECCO 2000, Proceedings*, pages 201–204.

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

Laumanns, M. and Ocenasek, J. (2002). Bayesian Optimization Algorithms for Multi-Objective Optimization. In Merelo, J. J., Adamidis, P., Beyer, H.-G., Martín, J. L. F.-V., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 298–307. Springer.

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

Li, J. and Aickelin, U. (2004). The Application of Bayesian Optimization and Classifier Systems in Nurse Scheduling. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 581–590. Springer.

Llorà, X. and Goldberg, D. E. (2003). Wise Breeding GA via Machine Learning Techniques for Function Optimization. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1172–1183. Springer.

Lobo, F. G., Deb, K., Harik, G. R., and Wang, L. (1998). Compressed Introns in a Linkage Learning Genetic Algorithm. In *Proceedings of the Third Annual Conference on Genetic Programming*, pages 551–558. Morgan-Kaufmann.

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

Lu, Q. and Yao, X. (2005). Clustering and Learning Gaussian Distribution for Continuous Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 35(2):195–204.

Mendiburu, A., Lozano, J., and Miguel-Alonso, J. (2005). Parallel Implementation of EDAs Based on Probabilistic Graphical Models. *IEEE Transactions On Evolutionary Computation*, 9(4):406–423.

Miquélez, T., Bengoetxea, E., and Larrañaga, P. (2004). Evolutionary Computation Based on Bayesian Classifiers. *International Journal of Applied Mathematics and Computer Science*, 14(3):335–349.

Mühlenbein, H. (1998). The Equation for Response to Selection and its Use for Prediction. *Evolutionary Computation*, 5:303–346.

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 Mahnig, T. (1999). FDA - A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions. *Evolutionary Computation*, 7(4):353–376.

Mühlenbein, H. and Mahnig, T. (2002a). Evolutionary Computation and Wright's Equation. *Theoretical Computer Science*, 287(1):145–165.

Mühlenbein, H. and Mahnig, T. (2002b). Evolutionary Optimization and the Estimation of Search Distributions with Applications to Graph Bipartitioning. *International Journal of Approximate Reasoning*, 31(3):157–192.

Mühlenbein, H., Mahnig, T., and Ochoa, A. (1999). Schemata, Distributions and Graphical Models in Evolutionary Optimization. *Journal of Heuristics*, 5:215–247.

Mühlenbein, H. and Paaβ, 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, 4th International Conference, Proceedings*, volume 1141 of *Lecture Notes in Computer Science*, pages 178–187. Springer.

Munetomo, M., Murao, N., and Akama, K. (2004). Empirical Investigations on Parallelized Linkage Identification. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 322–331. Springer.

Ocenasek, J., Kern, S., Hansen, N., and Koumoutsakos, P. (2004). A Mixed Bayesian Optimization Algorithm with Variance Adaptation. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 352–361. Springer.

Ocenasek, J., Schwarz, J., and Pelikan, M. (2003). Design of Multithreaded Estimation of Distribution Algorithms. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1247–1258. Springer.

Ochoa, A., Mühlenbein, H., and Soto, M. (2000a). A Factorized Distribution Algorithm Using Single Connected Bayesian Networks. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Proceedings*, volume 1917 of *Lecture Notes in Computer Science*, pages 787–796. Springer.

Ochoa, A., Mühlenbein, H., and Soto, M. (2000b). Factorized Distribution Algorithm Using Bayesian Networks. In Wu, A. S., editor, *Workshop Program at the Genetic and Evolutionary Computation Conference - GECCO 2000, Proceedings*, pages 212–215.

Ochoa, A., Soto, M., Santana, R., Madera, J., and Jorge, N. (1999). The Factorized Distribution Algorithm and the Junction Tree: A Learning Perspective. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 1999*, pages 368–377.

Paul, T. K. and Iba, H. (2003). Reinforcement Learning Estimation of Distribution Algorithm. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1259–1270. Springer.

Paul, T. K. and Iba, H. (2004). Identification of Informative Genes for Molecular Classification Using Probabilistic Model Building Genetic Algorithm. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 414–425. Springer.

Pelikan, M. and Goldberg, D. E. (2000a). Hierarchical Problem Solving and the Bayesian Optimization Algorithm. In Whitley, D., Goldberg, D. E., Cantú-Paz, E., Spector, L., Parmee, I., and Beyer, H. G., editors, *Genetic and Evolutionary Computation - GECCO 2000, Proceedings*, pages 267–274. Morgan Kaufmann.

Pelikan, M. and Goldberg, D. E. (2000b). Research on the Bayesian Optimization Algorithm. In Wu, A. S., editor, *Workshop Program at the Genetic and Evolutionary Computation Conference - GECCO 2000, Proceedings*, pages 212–215.

Pelikan, M. and Goldberg, D. E. (2003). Hierarchical BOA Solves Ising Spin Glasses and MAXSAT. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1271–1282. Springer.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian Optimization Algorithm. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Genetic and Evolutionary Computation - GECCO 1999, Proceedings*, volume 1, pages 525–532. Morgan Kaufmann.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (2000a). Bayesian Optimization Algorithm, Population Sizing, and Time to Convergence. In Whitley, D., Goldberg, D., Cantú-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Genetic and Evolutionary Computation - GECCO 2000, Proceedings*, pages 275–282. Morgan Kaufmann.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (2000b). Linkage Problem, Distribution Estimation and Bayesian Networks. *Evolutionary Computation*, 8(3):311–340.

Pelikan, M., Goldberg, D. E., and Lobo, F. (2002a). A Survey of Optimization by Building and Using Probabilistic Models. *Computational Optimization and Applications*, 21(1):5–20. Also IlliGAL Report No. 99018.

Pelikan, M., Goldberg, D. E., and Sastry, K. (2000c). Bayesian Optimization Algorithm, decision Graphs, and Occam's Razor. Technical Report IlliGAL, 200020, Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M. and Lin, T.-K. (2004). Parameter-Less Hierarchical BOA. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman,

M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 24–35. Springer.

Pelikan, M. and Mühlenbein, H. (1999). The Bivariate Marginal Distribution Algorithm. *Advances in Soft Computing-Engineering Design and Manufacturing*, pages 521–535.

Pelikan, M., Sastry, K., and Goldberg, D. E. (2002b). Scalability of the Bayesian Optimization Algorithm. *International Journal of Approximate Reasoning*, 31:221–258.

Peña, J. M., Lozano, J. A., and Larrañaga, P. (2005). Globally Multimodal Problem Optimization Via an Estimation of Distribution Algorithm Based on Unsupervised Learning of Bayesian Networks. *Evolutionary Computation*, 13(1):43–66.

Posik, P. (2004). Distribution Tree-Building Real-Valued Evolutionary Algorithm. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo, J. J., Bullinaria, J. A., Rowe, J. E., Tiño, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*, pages 372–381. Springer.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimieriung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Fromman-Holzboog, Stuttgart.

Rissanen, J. (1978). Modeling by Shortest Data Description. *Automatica*, pages 465–471.

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.

Ross, S. M. (2003). *Introduction to Probability Models*. Academic Press.

Rudlof, S. and Köppen, M. (1996). Stochastic Hill Climbing by Vectors of Normal Distributions. In *Proceedings of the First Online Workshop on Soft Computing (WSC1)*.

Rudolph, G. (1997). *Convergence Properties of Evolutionary Algorithms*. Verlag Kovač Hamburg.

Rudolph, G. (1998). Finite Markov Chains Results in Evolutionary Computation: A Tour d'Horizon. *Fundamenta Informaticae*, 35(1-4):67–89.

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

Sastry, K. and Goldberg, D. E. (2004). Designing Competent Mutation Operators Via Probabilistic Model Building of Neighborhoods. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 114–125. Springer.

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

Schwarz, J. and Ocenasek, J. L. (1999). Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Algorithms BMDA and BOA. In *Proceedings of the Fifth International Conference on Soft Computing*, pages 124–130.

Schwefel, H.-P. (1973). *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. John Wiley & Sons, Inc.

Sebag, M. and Ducoulombier, A. (1998). Extending Population-Based Incremental Learning to Continuos Search Spaces. In Eiben, . E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN V, 5th International Conference, Proceedings*, volume 1498 of *Lecture Notes in Computer Science*, pages 418–427. Springer.

Servet, I., Trave-Massuyes, L., and Stern, D. (1997). Telephone Network Traffic Overloading Diagnosis and Evolutionary Techniques. In *Proceedings of the Third European Conference on Artificial Evolution, AE 1997*, pages 137–144.

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

Shan, Y., McKay, R. I., Abbass, H. A., and Essam, D. (2003). Program Evolution with Explicit Learning. In Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003*, pages 1639–1646. IEEE Press.

Shapiro, J. L. (2003). The Sensitivity of PBIL to its Learning Rate, and How Detailed Balance Can Remove it. In *Proceedings of Foundations of Genetic Algorithms FOGA 7*, pages 115–132. Morgan Kaufmann.

Shapiro, J. L. (2005). Drift and Scaling in Estimation of Distribution Algorithms. *Evolutionary Computation*, 13(1):99–123.

Sheinerman, E. R. (1996). *Invitation to Dynamical Systems*. Prentice-Hall.

Shimosaka, H., Hiroyasu, T., and Miki, M. (2003). Comparison of Pulling Back and Penalty Methods for Constraints in BGA. In Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003*, pages 1941–1948. IEEE Press.

Smith, P. W. F. and Whittaker, J. (1998). Edge Exclusion Tests for Graphical Gaussian Models. In *Learning in Graphical Models*, pages 555–574. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Soto, M., Ochoa, A., Acid, S., and de Campos, L. M. (1999). Introducing the Polytree Aproximation of Distribution Algorithm. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 1999*, pages 360–367.

Spirtes, P., Glymour, C., and Scheines, R. (1991). An Algorithm for Fast Recovery of Sparse Causal Graphs. *Social Science Computing Reviews*, 9:62–72.

Taylor, H. M. and Karlin, S. (1994). *An Introduction to Stochastic Modeling*. Academic Press.

Toussaint, M. (2003). The Structure of Evolutionary Exploration: On Crossover, Buildings Blocks, and Estimation of Distribution Algorithms. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R. K., Kendall, G., Wilson, S. W., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J. F., editors, *Genetic and Evolutionary Computation - GECCO 2003, Proceedings, Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 1444–1456. Springer.

Tsuji, M., Munetomo, M., and Akama, K. (2004). Modeling Dependencies of Loci with String Classification According to Fitness Differences. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 246–257. Springer.

Tsutsui, S. (2002). Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram. In Merelo, J. J., Adamidis, P., Beyer, H.-G., Martín, J. L. F.-V., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 224–233. Springer.

Tsutsui, S., Hiroyasu, T., and Miki, M. (2003). The Effect of Introducing a Tag Node in Solving Scheduling problems Using Edge Histogram Based Sampling Algorithms. In Sarker, R., Reynolds,

R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003*, pages 22–29, Canberra. IEEE Press.

van Kemenade, C. H. M. (1998). Building Block Filtering and Mixing. In *Proceedings of the Fifth IEEE International Conference on Evolutionary Computation*. IEEE Press.

Vose, M. D. (1999a). Random Heuristic Search. *Theoretical Computer Science*, 1-2(229):103–142.

Vose, M. D. (1999b). *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. John Willey & Sons.

Wolpert, D. H. and Macready, W. G. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wright, A. H., Poli, R., Stephens, C. R., Langdon, W. B., and Pulavarty, S. (2004). An Estimation of Distribution Algorithm Based on Maximum Entropy. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E. K., Darwen, P. J., Dasgupta, D., Floreano, D., Foster, J. A., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A. M., editors, *Genetic and Evolutionary Computation - GECCO 2004, Proceedings, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 343–354. Springer.

Yanai, K. and Iba, H. (2003). Estimation of Distribution Programming Based on Bayesian Network. In Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003*, pages 1618–1625. IEEE Press.

Yuan, B. and Gallagher, M. (2003). Playing in Continuous Spaces: Some Analysis and Extension of Population-Based Incremental Learning. In Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., and Gedeon, T., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003*, pages 443–450. IEEE Press.

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.

Zlochin, M. and Dorigo, M. (2002). Model-Based Search for Combinatorial Optimization: A Comparative Study. In Merelo, J. J., Adamidis, P., Beyer, H.-G., Martín, J. L. F.-V., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 651–664. Springer.