
**Universidad Politécnica de Madrid**
**Facultad de Informática**
**Departamento de Inteligencia Artificial**

# Regularized Model Learning in EDAs for Continuous and Multi-Objective Optimization

A Thesis
submitted to the Department of Artificial Intelligence at the School of Computer Science of the Technical University of Madrid, as partial fulfillment of the requirements for the degree of Doctor of Philosophy

By
**Hossein Karshenas**

Supervisors
**Pedro Larrañaga and Concha Bielza**

2013

*I want to dedicate this thesis to my family, who has always been a source of love, inspiration and encouragement in my life.*

*And, in loving memory of my dear grandmother.*

# Acknowledgements

The work presented in this thesis would have not been possible without all the help and assistance I received from many individuals. First of all, I would like to thank my supervisors, Pedro Larrañaga and Concha Bielza, for their guidance, support and patience. I have learnt a lot from them during this period of time, and our meetings and talks gave me many new inspirations and motivations for completing this work.

I am very grateful to Roberto Santana who has been like a third supervisor to me. Our meetings, discussions and chats helped me to obtain a better understanding of EDAs and gave me many new ideas for advancing in my research. Thank you Roberto for your help.

I also want to thank Fernando Lobo for receiving me as a visitor in his laboratory in the University of Algarve at Faro, Portugal. I enjoyed very much the calm and friendly working environment with all the people at the laboratory, especially Mosab Bazargani and Hossein Moeinzadeh, who helped me a lot during my stay there.

I like to mention the Consolider Ingenio 2010-CSD-2007-00018 and TIN2010-20900-C04-04 projects, and the financial grant GI111015026 from the Technical University of Madrid, which supported me during this period of time while I was working as a PhD student.

At last but not least, I am grateful to all my friends and colleagues at the Computational Intelligence Group, Rubén Armañanzas, Diego Vidaurre, Hanen Borchani, Pedro Luis López-Cruz, Alfonso Ibáñez, Luis Guerra, Bojan Mihaljevic and Laura Antón-Sánchez, for all of their supports and assistances and for providing an excellent working environment. Thanks to all of you.

# Abstract

*Probabilistic modeling is the defining characteristic of estimation of distribution algorithms (EDAs) which determines their behavior and performance in optimization. Regularization is a well-known statistical technique used for obtaining an improved model by reducing the generalization error of estimation, especially in high-dimensional problems. $\ell_1$-regularization is a type of this technique with the appealing variable selection property which results in sparse model estimations.*

*In this thesis, we study the use of regularization techniques for model learning in EDAs. Several methods for regularized model estimation in continuous domains based on a Gaussian distribution assumption are presented, and analyzed from different aspects when used for optimization in a high-dimensional setting, where the population size of EDA has a logarithmic scale with respect to the number of variables. The optimization results obtained for a number of continuous problems with an increasing number of variables show that the proposed EDA based on regularized model estimation performs a more robust optimization, and is able to achieve significantly better results for larger dimensions than other Gaussian-based EDAs. We also propose a method for learning a marginally factorized Gaussian Markov random field model using regularization techniques and a clustering algorithm. The experimental results show notable optimization performance on continuous additively decomposable problems when using this model estimation method.*

*Our study also covers multi-objective optimization and we propose joint probabilistic modeling of variables and objectives in EDAs based on Bayesian networks, specifically models inspired from multi-dimensional Bayesian network classifiers. It is shown that with this approach to modeling, two new types of relationships are encoded in the estimated models in addition to the variable relationships captured in other EDAs: objective-variable and objective-objective relationships. An extensive experimental study shows the effectiveness of this approach for multi- and many-objective optimization. With the proposed joint variable-objective modeling, in addition to the Pareto set approximation, the algorithm is also able to obtain an estimation of the multi-objective problem structure.*

*Finally, the study of multi-objective optimization based on joint probabilistic modeling is extended to noisy domains, where the noise in objective values is represented by intervals. A new version of the Pareto dominance relation for ordering the solutions in these problems, namely $\alpha$-degree Pareto dominance, is introduced and its properties are analyzed. We show that the ranking methods based on this dominance relation can result in competitive performance of EDAs with respect to the quality of the approximated Pareto sets. This dominance relation is then used together with a method for joint probabilistic modeling based on $\ell_1$-regularization for multi-objective feature subset selection in classification, where six different measures of accuracy are considered as objectives with interval values. The individual assessment of the proposed joint probabilistic modeling*

*and solution ranking methods on datasets with small-medium dimensionality, when using two different Bayesian classifiers, shows that comparable or better Pareto sets of feature subsets are approximated in comparison to standard methods.*

# Resumen

*La modelización probabilista es la característica definitoria de los algoritmos de estimación de distribuciones (EDAs, en inglés) que determina su comportamiento y rendimiento en problemas de optimización. La regularización es una técnica estadística bien conocida utilizada para obtener mejores modelos al reducir el error de generalización de la estimación, especialmente en problemas de alta dimensionalidad. La regularización $\ell_1$ tiene la atractiva propiedad de seleccionar variables, lo que permite estimaciones de modelos dispersos.*

*En esta tesis, estudiamos el uso de técnicas de regularización para el aprendizaje de modelos en EDAs. Se presentan y analizan dos aproximaciones para la estimación del modelo regularizado en dominios continuos basado en la asunción de distribución Gaussiana desde diferentes aspectos en situaciones de alta dimensionalidad, donde el tamaño de la población del EDA es logarítmico en el número de variables. Los resultados de optimización obtenidos para algunos problemas continuos, con un número creciente de variables, muestran que el EDA propuesto, basado en la estimación del modelo regularizado, realiza una optimización más robusta y es capaz de lograr resultados significativamente mejores para dimensiones más grandes en comparación con otros EDAs basados en la asunción de distribución Gaussiana. También proponemos un método para aprender un modelo de campo aleatorio de Markov Gaussiano que está marginalmente factorizado utilizando técnicas de regularización y un algoritmo de clustering. Los resultados experimentales muestran un rendimiento notable en optimización de problemas continuos aditivamente descomponibles cuando se utiliza este método de estimación del modelo.*

*Nuestro estudio también cubre optimización multi-objetivo y proponemos una modelización probabilística conjunta de variables y objetivos en EDAs basada en redes Bayesianas, específicamente redes Bayesianas multidimensionales. Se demuestra que con este enfoque de modelización, en los modelos estimados se codifican dos nuevos tipos de relaciones además de las relaciones de variables capturadas en otros EDAs: relaciones de objetivo-variable y de objetivo-objetivo. Un amplio estudio experimental muestra la efectividad de este enfoque para optimización multi-objetivo y para muchos objetivos. Con la modelización propuesta, conjunta para variables y objetivos, además de la aproximación del conjunto de Pareto, el algoritmo es también capaz de obtener una estimación de la estructura del problema multi-objetivo.*

*Por último, el estudio de optimización multi-objetivo basado en modelización probabilística conjunta se extiende a dominios con ruido, donde el ruido en los valores de los objetivos se representa por intervalos. Se introduce una nueva versión de la relación de dominancia de Pareto para ordenar las soluciones en estos problemas, denominada dominancia de Pareto de grado $\alpha$, y se analizan sus propiedades. Mostramos que los métodos de ordenación basados en esta relación de dominancia pueden resultar en EDAs*

*con un rendimiento competitivo con respecto a la calidad de la aproximación del conjunto dePareto. Esta relación de dominancia se utiliza después junto con un método de modelización probabilística conjunta basado en regularización $\ell_1$ para selección multi-objetivo de subconjuntos de características en clasificación, donde se consideran seis medidas diferentes de precisión como los objetivos con valores de intervalo. La evaluación individual de estas dos propuestas de modelación probabilística conjunta y los métodos de ordenación en conjuntos de datos de pequeña-mediana dimensionalidad, cuando se utilizan dos clasificadores Bayesianos diferentes, demuestra que se aproximan conjuntos de Pareto de subconjuntos de características que comparables o mejores que con métodos estándar.*

# Contents

# Preface

In the everyday life we face different optimization problems, like the fastest way to reach the workplace considering traffic conditions, proper scheduling of our daily tasks so that it can be done in the shortest time with minimum effort, or even when we are selecting the food menu in a restaurant. From this point of view, humankind can be considered as a general-purpose problem-solver. Similarly, many of the computer-based intelligent systems used nowadays need to find solutions to real-world problems, and therefore optimization algorithms have become an indispensable part of these systems.

Evolutionary algorithms (EAs) are a type of meta-heuristic search methods, developed in the field of artificial intelligence, which have been successfully applied to find satisfactory solutions for problems with complicated and huge search spaces. These algorithms perform a rapid stochastic exploration of the search space guided by the survival of the fittest principle in evolution. As a relatively new class of EAs, EDAs employ a systematic approach to account for the uncertainty in the exploration of the search space by incorporating probabilistic modeling into the evolutionary framework for optimization.

Learning a probabilistic model which allows an effective search for the solutions of complex optimization problems remains a challenging task in the development of EDAs. As a step in this direction, in this thesis, we use model estimation based on *regularization* in EDAs, and study how it influences these optimization algorithms. Regularization is a technique used in machine learning and statistics for improving model estimation from a limited dataset of samples. This technique is especially effective for model learning in high-dimensional domains, where many parameters should be estimated from a small number of samples. This property of regularized model estimation is very interesting for EDAs, since the size of the population used for model estimation is always a point of concern in these algorithms.

Another important motivation for using regularization is the promising properties of a special type of this technique, known as $\ell_1$-regularization. In model estimation based on $\ell_1$-regularization, some of the model parameters become exactly zero, ruling out the relationships between the corresponding problem variables. This, in effect, permits EDAs to perform an implicit linkage learning when using $\ell_1$-regularized model estimation. We also study this effect of regularization in the thesis, mainly considering continuous problem optimization.

Multi-objective optimization is another major part of consideration in this thesis, since many of the real-world problems involve several conflicting criteria. So far, the probabilistic modeling used in multi-objective EDAs for search space exploration only comprises variables. In order to explicitly exploit the quality information of the solutions, provided by objective functions, in the generation of new solutions, and moreover, to model the relationships between variables and objectives, we propose and study *joint*

*variable-objective modeling* in EDAs for multi-objective optimization. We are especially interested in the performance of this approach as the number of problem objectives increases.

In some of the real-world problems, the objective values of the solutions are not exact and can change in different evaluations of the objective functions. The classification accuracy measures used for feature subset selection is an example of these problems, where the estimated quality of a feature subset depends on the dataset used for its evaluation. Therefore, usually techniques like $k$-fold cross-validation are used to obtain better estimation of the feature subsets quality by averaging over the quality values estimated in different folds. However, another possibility is to consider the confidence interval of the real quality, based on the quality values estimated in different folds. In this thesis, we introduce $\alpha$-degree Pareto dominance relation for ordering the solutions in multi-objective optimization when the values of objective functions are given as intervals. Using ranking methods based on this dominance relation, we study the performance of joint probabilistic modeling for multi-objective optimization of this type of problems, and specifically multi-objective feature subset selection.

The thesis is organized in four parts. The first part gives an introduction on probabilistic modeling in EDAs and some of their applications, and consists of three chapters. Chapter 1 briefly introduces probabilistic graphical models. It presents the related terms and concepts and reviews some of the properties of the probabilistic models used later in the thesis together with their learning and sampling methods. Chapter 2 reviews probabilistic modeling in different EDAs proposed in the literature and discusses some of the implications and characteristics of this approach to optimization. In Chapter 3, we show that one of the interesting application domains of EAs is the optimization problems defined in the learning and inference of Bayesian networks, which are one of the probabilistic models frequently used in EDAs.

The second part, which is composed of four chapters, presents regularized model estimation and its use for continuous optimization. Regularization is introduced in Chapter 4 and some of the employed regularization techniques are explained. In Chapter 5, we present two main approaches to regularized model estimation in continuous EDAs based on Gaussian distributions, and analyze both from several aspects like model accuracy and time complexity. Chapter 6 introduces RegEDA, an EDA based on regularized model estimation, and studies its performance in continuous optimization under a high-dimensional setting. The statistical analysis of the difference in the optimization results compared with several state-of-the-art Gaussian-based EDAs is also presented. Chapter 7 proposes a method for learning marginally factorized models using regularization techniques in an EDA based on Gaussian Markov random fields, and applies it for the optimization of continuous deceptive function and protein folding prediction problems.

In the third part of the thesis, we describe joint probabilistic modeling for multi-objective optimization. This part comprises five chapters. Chapter 8 gives an introduction to multi-objective optimization and reviews EDAs proposed for this purpose in the literature. Joint variable-objective modeling is proposed in Chapter 9, and the corresponding methods for learning and sampling such a model in an EDA based on Gaussian Bayesian networks is explained. The performance of the proposed algorithm is then evaluated on a set of difficult multi-objective problems in comparison with other algorithms. In Chapter 10, this approach is extended and studied in more detail by using a specific

probabilistic model inspired from multi-dimensional Bayesian network classifiers for joint variable-objective modeling. The resulting MBN-EDA is then applied for many-objective optimization and compared with several competitive multi-objective EAs. Moreover, an analysis of the joint probabilistic modeling in the proposed algorithm is presented. Chapter 11 extends the application of MBN-EDA to noisy domains by introducing $\alpha$-degree Pareto dominance relation for dealing with the noisy objective values given as intervals. This idea is then used in Chapter 12 to apply MBN-EDA for multi-objective feature subset selection by employing a specific joint modeling scheme based on $\ell_1$-regularization.

Finally, the thesis is concluded in Chapter 13, the sole chapter of the fourth part, where the conclusions and some perspectives of future works for extending the research conducted in this thesis are given.

# Part I

# Introduction

# Chapter 1

# Probabilistic Graphical Models

## 1.1 Introduction

Probability theory has provided a sound basis for many of the scientific and engineering tasks. Artificial intelligence, and more specifically machine learning, is one of the fields that has exploited probability theory to develop new algorithms and theorems. Probabilistic graphical models (PGMs) combine the concepts in probability and graph theories to provide a more comprehensible representation of the joint probability distribution over a vector of random variables. The main characteristic of this type of models is that they consist of a graphical structure and a set of parameters that together encode a joint probability distribution for the random variables.

A popular class of PGMs, Bayesian networks (BNs), first introduced in [Pearl, 1985], has been highly favored and extensively used in many machine learning applications. This tool can point out useful modularities in the underlying problem and help to accomplish the reasoning and decision making tasks especially in uncertain domains. The application of BNs has been further improved by the development of different methods proposed for inference (reasoning) [Lauritzen and Spiegelhalter, 1988] and automatic induction [Cooper and Herskovits, 1992] from a set of samples.

Different types of PGMs have been introduced in the literature so far: Markov networks, Bayesian networks, dependency networks [Heckerman et al., 2001], chain graphs [Frydenberg, 1990]. In this chapter, some of the important concepts related to probabilistic graphical modeling, mainly in the context of Bayesian and Markov networks, which will be later referenced and used throughout the thesis, are reviewed. For more detailed information on PGMs and their use, see specific references related to the topic, e.g. [Koller and Friedman, 2009] and [Larrañaga and Moral, 2011].

## 1.2 Probability-Related Notations

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a vector of random variables and $\boldsymbol{x} = (x_1, \ldots, x_n)$ a possible value-setting for these variables. $x_i$ denotes a possible value of $X_i$, the $i^{th}$ component of $\boldsymbol{X}$, and $\boldsymbol{y}$ denotes a possible value-setting for the sub-vector $\boldsymbol{Y} = (X_{J_1}, \ldots, X_{J_k})$, $J = \{J_1, \ldots, J_k\} \subseteq \{1, \ldots, n\}$.

If all variables in vector $\boldsymbol{X}$ are discrete, $P(\boldsymbol{X} = \boldsymbol{x})$ (or simply $P(\boldsymbol{x})$) is used to denote

the *joint probability mass* of a specific value-setting $\boldsymbol{x}$ for the variables. The *conditional probability mass* of a specific value $x_i$ of variable $X_i$ given that $X_j = x_j$ is denoted by $P(X_i = x_i \mid X_j = x_j)$ (or simply $P(x_i \mid x_j)$). Similarly, for continuous variables, the *joint density function* will be denoted as $p(\boldsymbol{x})$ and the *conditional density function* by $p(x_i \mid x_j)$. When the nature of variables in $\boldsymbol{X}$ is irrelevant or $\boldsymbol{X}$ consists of both discrete and continuous variables, $\rho(\boldsymbol{x})$ will be used to represent the generalized joint probability distribution.

Let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sub-vectors of variables. Then, $\boldsymbol{Y}$ is said to be *conditionally independent* of $\boldsymbol{Z}$ given $\boldsymbol{W}$ (denoted by $I(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{W})$), iff $\rho(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{w}) = \rho(\boldsymbol{y} \mid \boldsymbol{w})$, for all $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{w}$.

## 1.3 Bayesian Networks

A BN $\mathcal{B}(\mathcal{S}, \Theta)$ for a vector of variables $\boldsymbol{X} = (X_1, \ldots, X_n)$ consists of two components:

- A structure $\mathcal{S}$ represented by a directed acyclic graph (DAG), expressing a set of conditional independencies between variables [Dawid, 1979].
- A set of local parameters $\Theta$ representing the conditional probability distributions for the values of each variable given different value-settings of their parents according to the structure $\mathcal{S}$.

Figure 1.1(a) shows an example of a BN structure for a problem with six variables. Let $\boldsymbol{Nd}_i$ represent the set of non-descendants of variable $X_i$, i.e. all of the variables except $X_i$, its children and grandchildren to any level in the structure $\mathcal{S}$. Also, let $\boldsymbol{Pa}_i$ denote the set of parents of variable $X_i$, i.e. the set of variables with a direct link outgoing to variable $X_i$ in the structure $\mathcal{S}$. Then, for each variable $X_i$, $i = 1, \ldots, n$, structure $\mathcal{S}$ represents the assertion that $X_i$ and its non-descendants excluding its parents are conditionally independent given its parents:

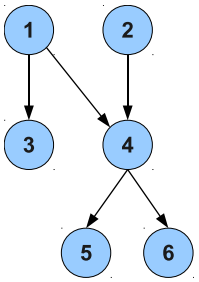
$$I(X_i, \{\boldsymbol{Nd}_i \setminus \boldsymbol{Pa}_i\} \mid \boldsymbol{Pa}_i).$$

This property is known as the Markov condition of BNs. Because of this condition, it can be shown that a BN encodes a factorization for the joint probability distribution of the variables in $\boldsymbol{X}$

$$\rho(\boldsymbol{x}) = \rho(x_1, \ldots, x_n) = \prod_{i=1}^{n} \rho_{\mathcal{B}}(x_i \mid \boldsymbol{pa}_i), \tag{1.1}$$

where $\boldsymbol{pa}_i$ denotes a possible value-setting for the parents $\boldsymbol{Pa}_i$. Equation (1.1) states that the joint probability distribution of the variables represented by a BN can be computed as the product of univariate conditional probability distributions of the variables given their parents. These conditional probability distributions are encoded as local parameters $\boldsymbol{\theta}_i$ in the BN.

A related notion in BNs is the so-called *Markov blanket* (MB) [Pearl, 1988] of the variables. The MB of a variable in a BN consists of its parents, its children and the parents of its children (spouses). The important property of this subset is that a variable in the BN is only influenced by its MB. In other words, given its MB, a variable is




(a) Structure

| $X_1$ | 1 | 1 | 1 | 2 | 2 | 2 |
|---|---|---|---|---|---|---|
| $X_2$ | 1 | 2 | 3 | 1 | 2 | 3 |
| $P(X_4=1 \mid X_1,X_2)$ | $\theta_{411}$ | $\theta_{421}$ | $\theta_{431}$ | $\theta_{441}$ | $\theta_{451}$ | $\theta_{461}$ |
| $P(X_4=2 \mid X_1,X_2)$ | $\theta_{412}$ | $\theta_{422}$ | $\theta_{432}$ | $\theta_{442}$ | $\theta_{452}$ | $\theta_{462}$ |
| $P(X_4=3 \mid X_1,X_2)$ | $\theta_{413}$ | $\theta_{423}$ | $\theta_{433}$ | $\theta_{443}$ | $\theta_{453}$ | $\theta_{463}$ |
| $P(X_4=4 \mid X_1,X_2)$ | $\theta_{414}$ | $\theta_{424}$ | $\theta_{434}$ | $\theta_{444}$ | $\theta_{454}$ | $\theta_{464}$ |
| $P(X_4=5 \mid X_1,X_2)$ | $\theta_{415}$ | $\theta_{425}$ | $\theta_{435}$ | $\theta_{445}$ | $\theta_{455}$ | $\theta_{465}$ |

(b) Conditional probability table

| | $X_1$ | $X_2$ | $X_3$ | $X_5$ | $X_6$ |
|---|---|---|---|---|---|
| $\mu_4$ $\sigma_4$ | $b_{41}$ | $b_{42}$ | 0 | 0 | 0 |

(c) GBN parameter table

| $X_2$ | | | $X_1$ | $X_3$ | $X_5$ | $X_6$ |
|---|---|---|---|---|---|---|
| 1 | $\mu_{41}$ | $\sigma_{41}$ | $b_{411}$ | 0 | 0 | 0 |
| 2 | $\mu_{42}$ | $\sigma_{42}$ | $b_{412}$ | 0 | 0 | 0 |
| 3 | $\mu_{43}$ | $\sigma_{43}$ | $b_{413}$ | 0 | 0 | 0 |

(d) CGN parameter table

Figure 1.1: An example of a) a Bayesian network structure, and three possible types of parameters for one of its variables ($X_4$): b) Discrete domain, assuming that $r_i = i+1$; c) Continuous domain with Gaussian variables; d) Mixed domain with continuous Gaussian variables, assuming that variables $X_1$ and $X_4$ are continuous and $X_2$ is discrete.

conditionally independent of all other variables excluding its MB. If the MB of variable $X_i$ is denoted with $\boldsymbol{Mb}_i$, then this property can be stated as:

$$I(X_i, \boldsymbol{X} \setminus \boldsymbol{Mb}_i \mid \boldsymbol{Mb}_i).$$

### 1.3.1 Bayesian Network Parameterization

The set of local parameters $\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\}$ determine the conditional probability distributions in Equation (1.1). Depending on the type of variables and the underlining assumptions, the conditional probability distributions can be represented with different parameters.

#### Discrete Bayesian Networks

In discrete domains, when a variable $X_i$ has $r_i$ possible values, $\{x_i^1, \ldots, x_i^{r_i}\}$, and its parents $\boldsymbol{Pa}_i$ have $q_i$ possible value-settings, $\{\boldsymbol{pa}_i^1, \ldots, \boldsymbol{pa}_i^{q_i}\}$, then the local parameters of the corresponding node of the BN can be represented with a conditional probability table. Each entry of this table, $\theta_{ijk} = P_{\mathcal{B}}(x_i^k \mid \boldsymbol{pa}_i^j)$, denotes the probability of variable $X_i$ being in its $k^{th}$ value given that its parents are in their $j^{th}$ value-setting. Since all variables are discrete, the number of possible value-settings for the parents can be easily

computed as $q_i = \prod_{X_m \in \boldsymbol{Pa}_i} r_m$. Thus, the local parameters of the BN for the $i^{th}$ variable can be represented by $\boldsymbol{\theta}_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$. Figure 1.1(b) shows an example of a conditional probability table for a discrete variable in a BN.

### Gaussian Bayesian Networks

In continuous domains, it is usually assumed that $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a vector of Gaussian random variables with joint probability distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$, the mean vector, and $\boldsymbol{\Sigma}$, the covariance matrix, are the parameters of the multivariate Gaussian distribution. Although these parameters encode a type of PGM (which will be discussed later in this chapter), BNs have also been used for encoding the joint density function of an $n$-dimensional Gaussian random vector which are then called *Gaussian Bayesian networks* (GBNs) [Geiger and Heckerman, 1994].

The conditional probability distribution represented by the local parameters of each node of a GBN is a univariate Gaussian distribution for the variable corresponding to that node, determined by the values of its parents [Lauritzen, 1992; Geiger and Heckerman, 1994]

$$p_{\mathcal{B}}(x_i \mid \boldsymbol{pa}_i) = \mathcal{N}(\mu_i + \sum_{X_j \in \boldsymbol{Pa}_i} w_{ij}(x_j - \mu_j), \nu_i^2), \tag{1.2}$$

where

1. $\mu_i$ is the mean of variable $X_i$ in vector $\boldsymbol{\mu}$,
2. $\boldsymbol{w}_i$ is a vector of *linear regression coefficients* reflecting the strength of the linear relationship between each parent variable $X_j$ and variable $X_i$, computed as

$$\boldsymbol{w}_i = \boldsymbol{\Sigma}_{<X_i, \boldsymbol{Pa}_i>} \boldsymbol{\Sigma}^{-1}_{<\boldsymbol{Pa}_i, \boldsymbol{Pa}_i>},$$

3. $\nu_i^2$ is the conditional variance of variable $X_i$, computed as

$$\nu_i^2 = \boldsymbol{\Sigma}_{<X_i, X_i>} - \boldsymbol{\Sigma}_{<X_i, \boldsymbol{Pa}_i>} \boldsymbol{\Sigma}^{-1}_{<\boldsymbol{Pa}_i, \boldsymbol{Pa}_i>} \boldsymbol{\Sigma}_{<\boldsymbol{Pa}_i, X_i>}.$$

Here, $\boldsymbol{\Sigma}_{<\boldsymbol{U}, \boldsymbol{V}>}$ denotes a sub-matrix of $\boldsymbol{\Sigma}$ consisting of the rows corresponding to the variables in set $\boldsymbol{U}$ and columns corresponding to the variables in set $\boldsymbol{V}$. Thus, the local parameters of each node in a GBN can be represented with the triplet $\boldsymbol{\theta}_i = (\mu_i, \boldsymbol{w}_i, \nu_i)$. Figure 1.1(c) shows an example of the parameters for a node of a GBN.

### Conditional Gaussian Bayesian Networks

More generally, $\boldsymbol{X} = (X_1, \ldots, X_n)$ can be considered to be an $n$-dimensional mixed random vector containing both discrete and continuous variables. Assume a reordering of the variables that is partitioned into two disjoint sub-vectors, $\boldsymbol{Y} = (Y_1, \ldots, Y_r)$ and $\boldsymbol{Z} = (Z_1, \ldots, Z_{n-r})$, such that

1. $\boldsymbol{X} = (\boldsymbol{Y}, \boldsymbol{Z})$,
2. $\boldsymbol{Y}$ contains only the $r$ discrete variables and $\boldsymbol{Z}$ represents an $(n-r)$-dimensional continuous random vector.

Using this decomposition, $\boldsymbol{X}$ is said to follow a *conditional Gaussian distribution* [Lauritzen and Wermuth, 1989] if the conditional joint probability density function for $\boldsymbol{Z}$ given each value-setting $\boldsymbol{y}$ of variables in $\boldsymbol{Y}$, such that $P(\boldsymbol{y}) > 0$, is an $(n-r)$-dimensional Gaussian distribution

$$p(\boldsymbol{z} \mid \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{y}}). \tag{1.3}$$

Assuming certain conditions and restrictions, a BN can be used to encode the probability distribution of an $n$-dimensional mixed random vector $\boldsymbol{X} = (\boldsymbol{Y}, \boldsymbol{Z})$, with Gaussian continuous variables. Such a BN is called a *conditional Gaussian Bayesian network* (CGN). One of the restrictions in CGNs is that discrete variables should only have discrete parents in the structure $\mathcal{S}$. The local parameters of the nodes of a CGN vary depending on the type of their corresponding variable. For nodes with discrete variables, they simply represent conditional probability mass distributions, similar to those explained for discrete BNs. For nodes corresponding to continuous variables they represent a conditional Gaussian distribution (Equation (1.3)) for each of the possible value-settings of the discrete parent variables. Figure 1.1(d) shows an example of the parameters for a continuous node of a CGN.

### Other Types of Parameterization

Probability functions other than the Gaussian distribution have also been used to encode the parameters of Bayesian networks in continuous and mixed domains. Moral et al. [2001] proposed the use of exponential distributions in a piecewise function called *mixture of truncated exponentials* (MTEs) by partitioning the domain of continuous variables into disjoint hypercubes. MTE densities can be used to approximate other types of distributions and, in some cases (e.g., uniform or categorical), the target distribution can be exactly represented with an MTE density. MTE densities provide a more versatile alternative than discretizing continuous variables (which is another way of dealing with vectors of mixed variables and can be seen as an approximation with a mixture of uniform distributions). With this type of models, there is no restriction on the order of the variables in the network (i.e. discrete variables can have continuous parents).

Another recently proposed parameterization for Bayesian networks is based on polynomial functions [Shenoy and West, 2011]. This piecewise function that is defined by partitioning the domain of continuous variables into disjoint *hyper-rhombuses* [Shenoy, 2011] is called *mixture of polynomials*. Each piece of such a mixture can be a polynomial function of a different degree. Like MTEs, the Bayesian networks defined with the mixture of polynomials densities do not have any restriction on the order between discrete and continuous variables.

Both types of density functions are closed under the basic operations (multiplication and integration) required for probability propagation in Bayesian networks (see below). However, because of their complex definitions requiring the specification of many parameters (e.g. number of pieces, number of terms in each piece, coefficients, powers), they have only been applied to problems with few variables and they are still a topic of active research. Bielza et al. [2011a] compared these parameterization types with each other and with conditional Gaussian distributions in the context of influence diagrams, a specific type of Bayesian networks for decision making applications.

### 1.3.2 Inference in Bayesian Networks

The BN tool is mainly used to reason in domains with intrinsic uncertainty by propagating some given evidence through the model. Generally speaking, the propagation of evidence involves assigning probabilities to the values of a subset $\boldsymbol{X}_E$ of non-instantiated or unobserved variables $\boldsymbol{X}_U$ given the values of some other (observed) variables $\boldsymbol{X}_O = \boldsymbol{X} \setminus \boldsymbol{X}_U$. Basically, this can be done by a process called *marginalization*. In this process the marginal probability of variables in $\boldsymbol{X}_E$ is computed by summing or integrating the joint probability distribution over all possible value-settings for the remaining unobserved variables $\boldsymbol{X}_R = \boldsymbol{X}_U \setminus \boldsymbol{X}_E$. This reasoning mechanism inside the model (i.e. the propagation of evidence through the model) depends on the structure reflecting the conditional independencies between the variables. Cooper [1990] proved that this task is NP-hard in the case of BNs with general multiply connected structures.

The methods proposed for this task can be divided into two main categories: a) exact algorithms [Pearl, 1988; Lauritzen and Spiegelhalter, 1988], and b) approximate algorithms which include deterministic methods [Jensen and Anderson, 1990; van Engelen, 1997; Cano et al., 2003] and methods based on generating samples from the BN [Henrion, 1986; Shachter and Peot, 1989; Casella and George, 1992; Chib and Greenberg, 1995]. For detailed information about these algorithms the reader can refer to [Castillo et al., 1997; Jensen and Nielsen, 2007; Darwiche, 2009]. Here, we describe the probabilistic logic sampling (PLS) method [Henrion, 1986], also known as forward sampling, as an example of the approximate algorithms based on sampling, which is used later in the thesis for generating new samples from a BN.

PLS starts with finding an ancestral or topological ordering of the nodes in BN. In such an ordering, each node appears after its parent nodes according to the BN structure. Next, the conditional probability distributions encoded in the nodes of BN are sampled one-by-one according to their order of appearance in the ancestral ordering. Since the turn for sampling variable $X_i$ is only after its parents in BN structure $\mathcal{S}$ are already sampled, the probability distribution related to this variable can be fully determined, and therefore easily sampled. If the values of some of the variables are given before inference (i.e. $\boldsymbol{X}_O$), these variable are only set to the given values during the sampling process. A set of individuals (or samples) are generated from the BN by repeating the sampling process of the unobserved variables $\boldsymbol{X}_U$. Finally, the probabilities of the values for the variables in $\boldsymbol{X}_E$ are computed from the sampled set (e.g. by computing the frequencies).

Instead of finding the probability of a subset of the variables in the BN, we sometimes need to find a value-setting for these variables that results in the highest probability. The following two inference tasks are directly related to this requirement.

#### Total Abductive Inference

Also known as the *most probable explanation* (MPE) problem [Pearl, 1987], this type of inference finds the most probable value of each unobserved variable of the BN, given the values of the observed variables ($\boldsymbol{X}_O$). More formally, the aim is to obtain the configuration $\boldsymbol{x}_U^*$ for $\boldsymbol{X}_U$ such that

$$\boldsymbol{x}_U^* = \arg \max_{\boldsymbol{x}_U} \rho(\boldsymbol{x}_U \mid \boldsymbol{x}_O). \tag{1.4}$$

Searching for the MPE is just as complex (NP-hard) as probability propagation [Shimony, 1994]. In fact, the MPE can be obtained by using probability propagation algorithms, and replacing the final summation or integration operator in marginalization with a maximization operator [Nilsson, 1998].

#### Partial Abductive Inference

Also known as the *maximum a posteriori* (MAP) problem, this type of inference outputs the most probable configuration for just a subset of the unobserved variables $\boldsymbol{X}_E$ in BN, known as the *explanation set*. Here, the aim is to obtain the configuration $\boldsymbol{x}_E^*$ for $\boldsymbol{X}_E$ such that

$$\boldsymbol{x}_E^* = \arg \max_{\boldsymbol{x}_E} \rho(\boldsymbol{x}_E \mid \boldsymbol{x}_O). \tag{1.5}$$

This problem can be reformulated using an MPE problem, and then marginalizing over all variables in $\boldsymbol{X}_R$. Hence, finding the MAP is more complex than the MPE problem as it can have an intractable complexity (NP-hard) even for cases in which the MPE can be computed in polynomial time (e.g. polytrees) [Park and Darwiche, 2004].

### 1.3.3 Learning Bayesian Networks

The structure and conditional probabilities necessary for characterizing a BN can be provided either externally by experts, which is time consuming and prone to error, or by automatic learning from a database of samples. The task of learning BNs can be divided into two subtasks:

- *structural learning*, i.e., identification of the topology of the BN, and
- *parametric learning*, estimation of the numerical parameters defining the conditional probabilities, for a given network topology.

The different methods proposed for inducing a BN from a dataset of samples are usually classified by modeling type into two approaches [Buntine, 1996; Heckerman, 1998; Neapolitan, 2004; Daly et al., 2011]:

1. methods based on detecting conditional independencies, also known as constraint-based methods, and
2. score+search methods.

#### Constraint-based methods

The input of these algorithms is a set of conditional independence relations between subsets of variables, which they use to build a BN that represents a large percentage (and, whenever possible, all) of these relations [Spirtes et al., 2001]. The PC algorithm [Spirtes and Glymour, 1991] is a well-known example of these methods. Typically, hypothesis tests are used to find conditional independencies from a dataset. Once the structure has been learned, the conditional probability distributions, required to fully specify the BN model are estimated from the dataset. The usual method for estimating the parameters

is maximum likelihood (ML) estimation, although Laplace estimation and other Bayesian estimation approaches based on Dirichlet priors are also common.

Many of the other constraint-based algorithms are inspired by or can be considered as an improved version of the PC algorithm. Cheng et al. [2002] proposed a three-phase dependency analysis algorithm to learn Bayesian networks for encoding monotonic DAG-faithful probability distributions. The algorithm uses a polynomial number of conditional independence tests (a great improvement on PC, which uses an exponential number of tests) that are based on information-theoretic analysis. They also discuss the conditions under which the algorithm is able to obtain correct network structures.

Kalisch and Bühlmann [2007] applied the PC algorithm to estimate the network structure of very high-dimensional problems assuming Gaussian probability distributions for parameters. They adapted their algorithm to estimate the skeleton and equivalence class of the DAG structure (see below). A proof of the algorithm consistency in high-dimensionality was also given. They also proposed an improved version of the algorithm with further robustification [Kalisch and Bühlmann, 2008].

Yehezkel and Lerner [2009] recursively applied a sequence of conditional independence tests, edge direction and structure decomposition to autonomous substructures. In this way their recursive autonomy identification algorithm obtains a hierarchical structure helping it to considerably reduce the number of conditional independence tests. Bühlmann et al. [2010] proposed a simple-PC algorithm based on the concept of partial faithfulness of probability distributions and used it for variable selection in Gaussian linear models.

### Score+search methods

Constraint-based learning is quite an appealing approach as it is close to the semantics of BNs. However, most of the algorithms developed for structure learning fall into the score+search category. As the name implies, these methods have two major components:

1. a scoring metric that measures the quality of a candidate BN with respect to a dataset of samples, and

2. a search procedure to intelligently move through the space of possible BNs, as this space is enormous (see below for further discussion).

**Scoring metrics.** Most of the popular scoring metrics are based on one of the following approaches: i) marginal likelihood, and ii) penalized maximum likelihood. Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of $N$ independent and identically distributed samples, each consisting of a value-setting for the $n$-dimensional random vector $\boldsymbol{X}$, metrics based on *marginal likelihood* maximize the likelihood of BN structure $\mathcal{S}$ with respect to this dataset, $\rho_{\mathcal{S}}(\mathcal{D})$, assuming certain prior distributions for the parameters of BN which allows to compute the likelihood in a closed form [Cooper and Herskovits, 1992; Heckerman et al., 1995].

In discrete domains, a common prior probability assumption for the BN parameters is the Dirichlet distribution, characterized with parameters $\alpha_{ijk}$, which results (assuming a uniform prior distribution for the structures) in a scoring metric usually referred to as

the Bayesian Dirichlet equivalence (BDe) metric [Heckerman et al., 1995]:

$$\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij}+\alpha_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(N_{ijk}+\alpha_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{1.6}$$

where $\Gamma(v)$ is the Gamma function given by $\Gamma(v) = (v-1)!, \forall v \in \mathbb{N}$, and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, where $r_i$ shows the number of different values for the $i^{th}$ variable and $q_i$ represents the number of possible value-settings for its parents. $N_{ij}$ is the number of samples in the dataset that have the $j^{th}$ value-setting for the parents of the $i^{th}$ variable, and likewise $N_{ijk}$ is the number of samples for which the $i^{th}$ variable has its $k^{th}$ value and its parents are in their $j^{th}$ value-setting. In the specific case where all Dirichlet distribution parameters are uniformly set to $\alpha_{ijk} = 1$, the resulting scoring metric is usually called K2 metric, initially proposed for use in the K2 algorithm [Cooper and Herskovits, 1992].

A problem of metrics only considering the likelihood of the BN is that they give better scores to more complex models, thus leading to overfitting to the dataset used for learning BN. The metrics based on *penalized maximum likelihood* try to overcome this shortcoming by adding a penalization term to the likelihood of BN $\mathcal{B}(\mathcal{S}, \Theta)$ with respect to dataset $\mathcal{D}$:

$$\prod_{j=1}^{N}\prod_{i=1}^{n}\rho_{\mathcal{B}}(\boldsymbol{x}_{j<X_i>} \mid \boldsymbol{x}_{j<\boldsymbol{Pa}_i>}) - f(N)dim(\mathcal{B}), \tag{1.7}$$

where $\boldsymbol{x}_{j<U>}$ denotes part of the value-setting $\boldsymbol{x}_j$ corresponding to sub-vector $U \subseteq \boldsymbol{X}$, $dim(\mathcal{B})$ is the dimension of BN (number of individual parameters needed to specify the model), and $f(N)$ is a non-negative penalization function depending on the size of the dataset. Popular scoring metrics like Akaike's information criterion (AIC) [Akaike, 1974] and the Bayesian information criterion (BIC) [Schwarz, 1978] differ in their choice for this penalization function with values $f(N) = 1$ and $f(N) = 0.5 \log N$, respectively. In discrete domains, this metric reduces to

$$\prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\left(\frac{N_{ijk}}{N_{ij}}\right)^{N_{ijk}} - f(N)dim(\mathcal{B}), \tag{1.8}$$

where the dimension of BN can be computed as $dim(\mathcal{B}) = \sum_{i=1}^{n} q_i(r_i - 1)$.

Minimum description length (MDL) score [Rissanen, 1978; Grünwald, 1998] is another type of scoring metric based on information theory and data compression. This score, which is justified by Occam's razor principle favoring less complex models, is closely related to the logarithm of the penalized maximum likelihood score. In simple terms this metric can be described as follows. Suppose that the cost of encoding a dataset $\mathcal{D}$ with a model $\mathcal{B}$ is equal to the cost of describing the model plus the cost of describing the data with this model: $Cost(\mathcal{B}) + Cost(\mathcal{D} \mid \mathcal{B})$. Then the MDL score tries to select the model with the least total cost of description. Usually, the cost is expressed in terms of the number of bits required to represent the description.

A feature of scoring metrics that can greatly help the search algorithm is decomposability. With a decomposable metric, the score of a BN can be computed as the combination of scores obtained for smaller factors (e.g., a single variable). This property will allow the search algorithm to measure the effect of operations involving each factor independently of the effects of other BN factors. The metrics introduced here are all decomposable.

**Search methods.** The number of possible DAG structures for $n$ nodes is given by the following recursive formula [Robinson, 1977]:

$$f(n) = \begin{cases} \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i) & n > 1 \\ 1 & n = 0 \wedge n = 1. \end{cases} \tag{1.9}$$

In fact it has been shown that searching this huge space for the optimal structure (according to a scoring metric) is NP-hard, even with a constraint on the maximum number of parents for each node [Chickering et al., 1994; Chickering, 1996; Chickering et al., 2004]. Therefore, greedy local search techniques [Buntine, 1991], as well as many heuristic search methods such as simulated annealing [Heckerman et al., 1995], Tabu search [Bouckaert, 1995] and EAs (see Section 3.4.1) have been frequently employed for this purpose in the literature.

Here, we explain a basic greedy local structure search method for learning BNs [Buntine, 1991], which is relatively fast and is used in this thesis for inducing a BN from a dataset of samples. The algorithm starts with an initial structure for the network, which can be generated randomly or given based on some prior knowledge of the problem. At each iteration of the algorithm, all possible edge addition, removal and reversal operations are considered, and the one resulting in the best improvement of the scoring metric is selected and applied to the structure of BN. This step is repeated until no more operation can be found to further improve the scoring metric of BN, in which case the algorithm stops.

### 1.3.4 Bayesian Networks in Machine Learning

In machine learning, BNs have been used for both classification and clustering tasks. In these applications the node(s) of BN corresponding to the class variable(s) has a specific interpretation and therefore is treated different than other nodes corresponding to the feature or predictor variables.

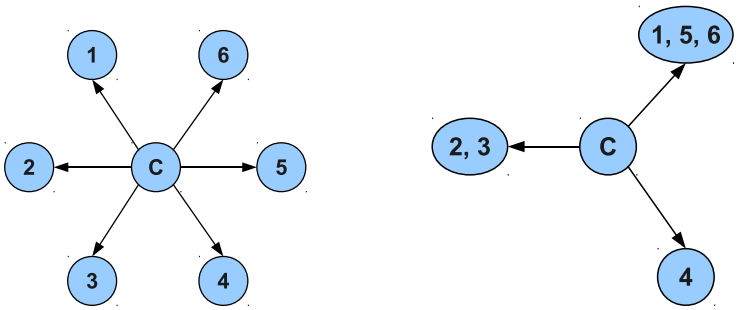
#### Supervised Learning

In recent years, there has been a sizable increase in published research using BNs for supervised classification tasks [Larrañaga et al., 2005]. Bayesian classifiers compute the class-value with the highest posterior probability ($c^*$) for each value-setting of predictor variables $(x_1, ..., x_n)$:

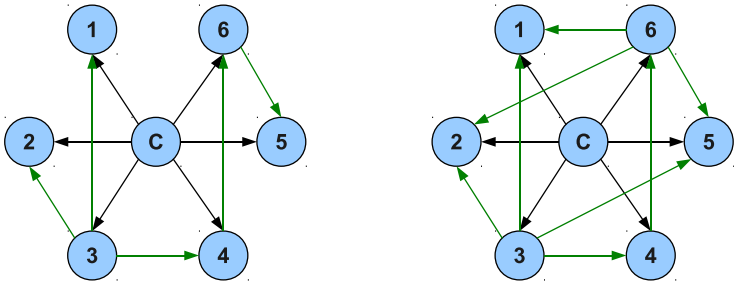
$$\begin{aligned} c^* &= \arg \max_c \; P(C = c \mid X_1 = x_1, \ldots, X_n = x_n) \\ &= \arg \max_c \; \rho(X_1 = x_1, \ldots, X_n = x_n \mid C = c) P(C = c). \end{aligned} \tag{1.10}$$

Different Bayesian classifiers can be obtained depending on the factorization of $\rho(X_1 = x_1, \ldots, X_n = x_n \mid C = c)$. Figure 1.2 shows examples of some Bayesian classifiers. *Naïve Bayes* (NB) [Minsky, 1961] (Figure 1.2a) is the simplest Bayesian classifier. It is built on the assumption that the predictor variables are conditionally independent given the class value

$$\rho(x_1, \ldots, x_n \mid c) = \prod_{i=1}^{n} \rho(x_i \mid c). \tag{1.11}$$




(a) Naïve Bayes (b) Semi naïve Bayes



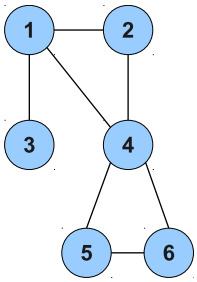

(c) TAN (d) $k$DB, $k = 2$

Figure 1.2: Examples of different types of Bayesian classifier structures.

The *semi-naïve Bayes* (SNB) classifier [Pazzani, 1996] (Figure 1.2b) considers forming new composite variables to avoid the conditional independence assumption of classical NB. These new variables are formed by joining the original predictor variables, and their values are obtained from the Cartesian product of the values of the constituent variables. Pazzani [1996] proposed a greedy wrapper approach for building a SNB classifier, where the irrelevant variables are removed from the model and the correlated variables are joined with the Cartesian product of their values. The *tree augmented naïve Bayes* (TAN) classifier [Friedman et al., 1997] (Figure 1.2c) extends the structure of NB classifier by constructing a tree over predictor variables to account for their relationships. The *k-dependence Bayesian* (kDB) classifier [Sahami, 1996] (Figure 1.2d) also extends NB classifier with a more general structure allowing each variable to have $k$ parents from the predictor variables. Bayesian classifiers can also be defined using the MB of the variables. Specifically, the MB of the class variable specifies the set of predictor variables affecting its posterior probability computation: $P(C \mid X_1, \ldots, X_n) = P(C \mid \boldsymbol{Mb}_C)$.

### Unsupervised Learning

Another major area of machine learning employing BNs is *unsupervised learning* or *clustering*. The clustering of the samples given for an $n$-dimensional random vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ should consider the structural constraint assumptions imposed by the data generation mechanism. In the case of BNs, the constraint states that there should be an edge from the random variable representing the cluster, $C$, to every predictor variable $X_i$. Thus, the factorization of the joint probability distribution for the $(n+1)$-dimensional




(a) Structure

| $X_1$ | $X_2$ | $X_4$ | $\Phi(X_1, X_2, X_4)$ |
|---|---|---|---|
| 1 | 1 | 1 | $\Phi_{1,2,4\text{-}1}$ |
| 1 | 2 | 2 | $\Phi_{1,2,4\text{-}2}$ |
| 1 | 3 | 1 | $\Phi_{1,2,4\text{-}3}$ |
| 1 | 1 | 2 | $\Phi_{1,2,4\text{-}4}$ |
| 1 | 2 | 1 | $\Phi_{1,2,4\text{-}5}$ |
| 1 | 3 | 2 | $\Phi_{1,2,4\text{-}6}$ |
| 2 | 1 | 1 | $\Phi_{1,2,4\text{-}7}$ |
| 2 | 2 | 2 | $\Phi_{1,2,4\text{-}8}$ |
| 2 | 3 | 1 | $\Phi_{1,2,4\text{-}9}$ |
| 2 | 1 | 2 | $\Phi_{1,2,4\text{-}10}$ |
| 2 | 2 | 1 | $\Phi_{1,2,4\text{-}11}$ |
| 2 | 3 | 2 | $\Phi_{1,2,4\text{-}12}$ |

(b) Parameter table

Figure 1.3: An example of a discrete Markov network with a parameter table for the factor $\{X_1, X_2, X_4\}$. It is assumed that $X_1$, $X_2$ and $X_4$ respectively have 2, 3 and 2 possible states.

random vector $(C, \boldsymbol{X})$ is given by

$$\rho_{\mathcal{B}}(c, \boldsymbol{x}) = P_{\mathcal{B}}(c) \prod_{i=1}^{n} \rho_{\mathcal{B}}(x_i \mid c, \boldsymbol{pa}_i). \tag{1.12}$$

Note that this is similar to the factorization considered for BNs in supervised classification. The main difference, however, is that the value of variable $C$ is unknown in clustering problems and has to be estimated using techniques like the *expectation-maximization* (EM) algorithm [Dempster et al., 1977].

## 1.4 Markov Networks

When the interactions between the variables are symmetrical and there is no specific direction for the influence of the variables over each other, an undirected graph is more appropriate for representing the correlations. Markov networks (MNs) are a type of PGM fitted to this need. An MN $\mathcal{M}(\mathcal{S}, \Phi_{\mathcal{C}})$ has two components:

- an undirected graphical structure $\mathcal{S}$, where each variable is depicted by a node and the undirected edges represent homogeneous dependencies between the variables, and

- a set of factors (non-negative functions) $\Phi_{\mathcal{C}}$, each defined over a clique (complete subgraph) of $\mathcal{S}$, that express the affinity of their associated variables and the compatibility of their values.

Figure 1.3 shows an exemplary MN structure and the parameters for one of its factors in discrete domains.

The normalized product of MN factors (according to a specific multiplication rule) define a joint probability distribution over $\boldsymbol{X}$. Let $\mathcal{C} = \{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_\kappa\}$ be a set of cliques of MN structure, such that $\bigcup_{k=1}^{\kappa} \boldsymbol{C}_k = \boldsymbol{X}$. Then, the so-called *Gibbs* distribution $\rho_{\Phi_{\mathcal{C}}}$,

parameterized by the set of factors $\Phi_{\mathcal{C}} = \{\phi_1(\boldsymbol{C}_1), \ldots, \phi_\kappa(\boldsymbol{C}_\kappa)\}$, and factorized over MN structure is given by:

$$\rho_{\Phi_{\mathcal{C}}}(\boldsymbol{x}) = \frac{1}{Z} \prod_{k=1}^{\kappa} \phi_k(\boldsymbol{x}_{\boldsymbol{C}_k}), \tag{1.13}$$

where $Z$ is a normalizing term, called the *partition function*, and is obtained by summing or integrating the unnormalized product of factors over all possible value-settings of the variables. It should be noted that MN parameters (i.e. factors) do not necessarily correspond to marginal or conditional probabilities.

In MNs, two subset of variables $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are *conditionally independent* given a third subset $\boldsymbol{W}$, $I(\boldsymbol{Y}, \boldsymbol{Z} \mid \boldsymbol{W})$, if and only if $\boldsymbol{W}$ *separates* $\boldsymbol{Y}$ and $\boldsymbol{Z}$ ($\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W} \subseteq \boldsymbol{X}$, and each two of these subsets are mutually disjoint). That is, all of the paths between every two nodes respectively in $\boldsymbol{Y}$ and $\boldsymbol{Z}$ passes through at least one node of $\boldsymbol{W}$.

Except the above property, known as the *global Markov property*, there are also two simple types of local conditional independencies encoded in MNs. If $\boldsymbol{Mb}_i$ represents the set of immediate neighbors of variable $X_i$ in MN structure, sometimes referred to as the MB of a variable in MN, then

- *Local Markov property*: Each variable $X_i$ is conditionally independent of all of its non-neighbor variables, given its MB

$$\forall X_i \in \boldsymbol{X} \quad \Longrightarrow \quad I(X_i, \{\boldsymbol{X} \setminus \boldsymbol{Mb}_i\} \mid \boldsymbol{Mb}_i)$$

- *Pairwise Markov property*: Each variable $X_i$ is conditionally independent of its non-neighbor variable $X_j$ given all other variables

$$\forall X_i, X_j \in \boldsymbol{X} : X_j \notin \boldsymbol{Mb}_i \quad \Longrightarrow \quad I(X_i, X_j \mid \boldsymbol{X} \setminus \{X_i, X_j\})$$

The independencies encoded in MN structure and those of factorized Gibbs distribution are related to each other under some specific assumptions. Specifically , the Hammersley-Clifford theorem [Hammersley and Clifford, 1971] states that if all of the conditional independencies encoded in MN structure $\mathcal{S}$ exist in the set of independencies implied by a *positive* Gibbs distribution $\rho_\Phi$ (i.e. a distribution that assigns non-zero probabilities to all possible value-settings of input variables), then $\rho_\Phi$ factorizes over a covering set of cliques, $\mathcal{C}$, of $\mathcal{S}$:

$$\forall \phi_k(\boldsymbol{C}_k) \in \Phi \quad \Longrightarrow \quad \boldsymbol{C}_k \in \mathcal{C}.$$

### 1.4.1 Multivariate Gaussian Distribution

A multivariate Gaussian distribution (MGD) $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ over a vector of $n$ random variables $\boldsymbol{X} = (X_1, \ldots, X_n)$ is defined with two parameters: $\boldsymbol{\mu}$ is the $n$-dimensional vector of mean values for each variable, and $\boldsymbol{\Sigma}$ is the $n \times n$ positive semidefinite (i.e. $\boldsymbol{x}\boldsymbol{\Sigma}\boldsymbol{x}^T \geq 0$, $\forall \boldsymbol{x} \in \{\mathbb{R}^n \setminus \boldsymbol{0}\}$) and symmetric covariance matrix. Here, we consider MGDs with positive definite covariance matrices. Positive definite matrices are guaranteed to be full-ranked and non-singular, and therefore their inverse can be computed.

Geometrically, MGDs specify a set of parallel ellipsoidal contours around the mean vector in $\mathbb{R}^n$. The mean vector determines the bias of each variable's values from origin

and the variances, i.e. entries along the diagonal of the covariance matrix, are responsible for specifying the spread of these values. The covariances (i.e. the off-diagonal entries in the covariance matrix) determine the shape of the ellipsoids.

The typical representation of an MGD, sometimes referred to as the *moment* form, is given by

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^T\right). \tag{1.14}$$

This equation can be transformed to the *information* form representation of MGD [Koller and Friedman, 2009], also known as the canonical or natural form

$$p(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}\boldsymbol{h}\boldsymbol{\Theta}^{-1}\boldsymbol{h}^T\right)}{\sqrt{(2\pi)^n |\boldsymbol{\Theta}^{-1}|}} \exp\left(-\frac{1}{2}\boldsymbol{x}\boldsymbol{\Theta}\boldsymbol{x}^T + \boldsymbol{x}\boldsymbol{h}^T\right) \tag{1.15}$$

where $\boldsymbol{h} = \boldsymbol{\mu}\boldsymbol{\Sigma}^{-1}$ is called the potential vector and $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ is the inverse covariance matrix, also known as the precision, concentration or information matrix.

For a valid MGD represented in the information form, the precision matrix should be positive definite. This matrix, gives the partial covariances between pairs of variables. A zero value in any entry $\theta_{ij}$ of this matrix implies that the corresponding two variables are conditionally independent given all other variables and vice versa:

$$\theta_{ij} = 0 \quad \Longleftrightarrow \quad I(X_i, X_j \mid \boldsymbol{X} \setminus \{X_i, X_j\}).$$

### 1.4.2 Gaussian Markov Random Fields

Pairwise MNs [Hammersley and Clifford, 1971] are a widely used class of MNs, where all of the factors are defined over either single or pairs of variables, i.e. a pairwise MN has two types of factors:

i) Node factors $\phi_i(X_i)$ defined over every single variable $X_i$,

ii) Edge factors $\phi_{ij}(X_i, X_j)$ defined over the ending variables $X_i$ and $X_j$ of every edge.

Pairwise MNs are closely related to MGDs. More specifically, the precision matrix of an MGD defines a set of pairwise Markov properties [Lauritzen, 1996] which can be encoded by a pairwise MN. Therefore the zero pattern of the precision matrix of an MGD directly induces a Gaussian MN, which is known as Gaussian Markov random field (GMRF) [Speed and Kiiveri, 1986; Rue and Held, 2005], a PGM successfully applied for handling uncertainty in many practical domains.

To obtain the structure of this MN, an edge is introduced between every two nodes whose corresponding variables are partially correlated (with a non-zero entry in the precision matrix). The parameters of the MN are node and edge factors obtained, for example, by decomposing the variable exponent of MGD in Equation (1.15) to terms consisting of

single and pairs of variables

$$\exp\Big(-\frac{1}{2}\boldsymbol{X}\boldsymbol{\Theta}\boldsymbol{X}^T+\boldsymbol{X}\boldsymbol{h}^T\Big)=\Big[\exp(-\frac{1}{2}\theta_{11}X_1^2+h_1X_1)\cdots\exp(-\frac{1}{2}\theta_{nn}X_n^2+h_nX_n)\Big]\cdot\Big[\exp(-\theta_{12}X_1X_2)\cdots\exp(-\theta_{1n}X_1X_n)\cdots\exp(-\theta_{n-1,n}X_{n-1}X_n)\Big].$$

Then, the joint Gibbs distribution factorizing over this MN can be obtained by computing the partition function as

$$Z=\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\boldsymbol{X}\boldsymbol{\Theta}\boldsymbol{X}^T+\boldsymbol{X}\boldsymbol{h}^T\right)d\boldsymbol{X}=\frac{\sqrt{(2\pi)^n|\boldsymbol{\Theta}^{-1}|}}{\exp\left(-\frac{1}{2}\boldsymbol{h}\boldsymbol{\Theta}^{-1}\boldsymbol{h}^T\right)}.$$

The possibility of transforming every MGD to a corresponding GMRF allows to work with this type of MNs in an indirect way. On the other hand, converting a pairwise MN with log-quadratic factors to an MGD is not straightforward. In fact, not every pairwise MN with log-quadratic factors can be converted into a valid MGD, as it may not necessarily result in a positive definite precision matrix [Koller and Friedman, 2009].

### 1.4.3 Learning and Sampling Gaussian Markov Networks

Because of the close relation between MGDs and GMRFs, here we consider learning MGDs from data and how to sample them for generating new individuals. The mean vector and covariance matrix of an MGD are computed as the first two moments of the distribution (considering row-wise vectors):

$$\begin{aligned}\boldsymbol{\mu}&=E(\boldsymbol{X})\\ \boldsymbol{\Sigma}&=E((\boldsymbol{X}-\boldsymbol{\mu})^T(\boldsymbol{X}-\boldsymbol{\mu}))=E(\boldsymbol{X}^T\boldsymbol{X})-\boldsymbol{\mu}^T\boldsymbol{\mu}.\end{aligned}$$

The total number of individual parameters that have to be estimated in order to determine an MGD is $(n^2+3n)/2$, i.e. of $O(n^2)$ complexity. Given a dataset $\mathcal{D}=\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$ of $N$ independent and identically distributed samples, the (unbiased) ML estimation of MGD parameters (mean vector and covariance matrix) is given by

$$\boldsymbol{m}=\frac{1}{N}\sum_{j=1}^{N}\boldsymbol{x}_j, \tag{1.16}$$

$$\boldsymbol{S}=\frac{1}{N-1}\sum_{j=1}^{N}(\boldsymbol{x}_j-\boldsymbol{m})^T(\boldsymbol{x}_j-\boldsymbol{m}). \tag{1.17}$$

To generate new individuals from a given MGD $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, first its covariance matrix should be decomposed. Two types of matrix decomposition are often used for this purpose:

- Eigen-decomposition, with $\boldsymbol{\Sigma}=\boldsymbol{U}\boldsymbol{D}^2\boldsymbol{U}^T$, where $\boldsymbol{U}$ is an orthogonal matrix (i.e. $\boldsymbol{U}\boldsymbol{U}^T=\boldsymbol{I}$) consisting of the eigenvectors of $\boldsymbol{\Sigma}$, and $\boldsymbol{D}$ is a diagonal matrix containing the square roots of eigenvalues of $\boldsymbol{\Sigma}$.

- Bartlett or Cholesky decomposition Bilodeau and Brenner [1999], with $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix.

Having a decomposition of covariance matrix, the sampling algorithm generates a vector $\boldsymbol{z}$ of $n$ random values, where each value is sampled independently from the standard univariate Gaussian distribution $\mathcal{N}(0, 1)$. The new individual is then obtained by setting $\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{z}\boldsymbol{D}\boldsymbol{U}^T$ (or $\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{z}\boldsymbol{L}^T$ in case of Cholesky decomposition).

## 1.5 Conclusions

In PGMs, the probability distribution for a vector of random variables is accompanied with a graphical representation. This graphical structure allows a better understanding and interpretation of the relationships between variables. BNs are one of the popular types of PGMs where the graphical structure is a DAG. The reasoning process in BNs usually tries to find the probability of the values for some of the variables given the values of other variables. However, sometimes the goal of inference is to find the value-setting of a subset of variables with the highest probability. Several algorithms have been proposed for learning a BN from a dataset of samples, usually classified into two general strategies: constraint-based learning and score+search. BNs have been used in machine learning tasks, where several types of Bayesian classifiers have been introduced.

MNs are another type of PGMs that encode symmetric interactions between variables, where the direction of the dependency is not important. The parameters of MN define a Gibbs distribution, factorized over the cliques of the MN structure. MN structure can encode several types of conditional independence properties. One of these properties known as the pairwise Markov property is in direct accordance with the correlations between variables encoded in the precision matrix of an MGD. Therefore the parameters of each MGD implicitly encode a pairwise MN.

# Chapter 2

# Evolutionary Optimization with Probabilistic Modeling

## 2.1 Introduction

The difficult and complex problems existing in real-world applications have increased the demand for effective meta-heuristic algorithms that are able to achieve good (and not necessarily optimal) solutions by performing an intelligent search of the space of possible solutions. Evolutionary computation is one of the most successful of these algorithms that has achieved very good results across a wide range of problem domains. Applying their nature-inspired mechanisms, e.g., survival of the fittest and genetic crossover and mutation, on a population of candidate solutions, evolutionary approaches like genetic algorithms [Holland, 1975] have been able to perform an effective and diverse search of the vast solution space of problems.

EDAs [Mühlenbein and Paaß, 1996; Larrañaga and Lozano, 2001; Pelikan, 2005; Lozano et al., 2006] are a relatively recent class of EAs developed by using probabilistic modeling in the framework of EAs. They have proven to be promising optimization algorithms for many difficult problems with high computational complexity. These algorithms explore the search space by building a probabilistic model from a set of selected candidate solutions. This probabilistic model is then used to sample new candidate solutions in the search space. As the result, these algorithms will provide a model expressing the regularities of the problem structure, as well as the final solutions. In this chapter we review the probabilistic models used in EDAs and discuss how they are employed for optimization. This review is published in [Larrañaga et al., 2012].

## 2.2 Evolutionary Algorithms

Over the last few decades several types of EAs, like genetic algorithms (GAs) [Holland, 1975], evolutionary strategy (ES) [Rechenberg, 1973], evolutionary programming (EP) [Fogel, 1966] and genetic programming (GP) [Cramer, 1985; Koza, 1992] have been proposed. They are considered as important meta-heuristic algorithms for solving many real-world problems. Figure 2.1 shows the common framework of a typical EA. Given a fitness function that evaluates the quality of solutions, the algorithm iteratively evolves a

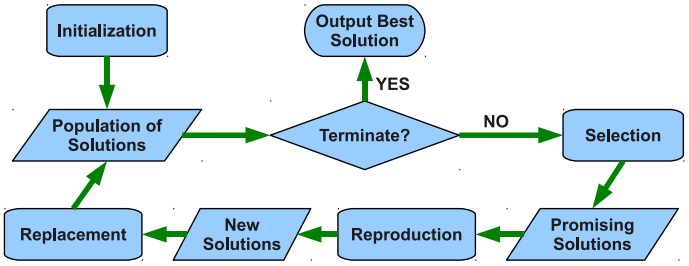



Figure 2.1: Flowchart of a typical evolutionary algorithm.

population of candidate solutions to the problem. Fitter solutions of the population are used to reproduce new offspring solutions (survival of the fittest principle) by applying genetic operators, i.e. crossover and mutation.

### 2.2.1 Genetic Algorithms

GAs are perhaps the most well-known and widely used EAs. Since their introduction [Holland, 1975], they have received an increasing amount of attention and interest, and numerous works have studied their different aspects. A typical GA works by evolving a population of candidate solutions to the problem across a number of generations in order to obtain better solutions. Solutions are usually represented as binary strings, the same as the representation of information in machine language. The algorithm, selects a subset of fitter solutions from the population according to a selection mechanism, e.g. tournament selection, as the parents. These parent solutions reproduce new offspring solutions by applying genetic operators like crossover and mutation. The newly generated solutions then compete for survival with the solutions in the population according to their fitnesses.

The simple and easy to understand mechanism of GAs with their simple solution representation method, has led to their intensive utilization for optimization in a vast variety of domains, from engineering tasks [Gen and Cheng, 2000] to medicine [Alander, 2012]. They have been also extensively used in multi-objective [Deb, 2001], uncertain and dynamic [Goh and Tan, 2009] domains under the general term of evolutionary algorithms. Despite their simple mechanics, several works have also studied the performance of these algorithms from a theoretical point of view [Holland, 1975; Harik et al., 1999a]. For further information on these algorithms see [Goldberg, 1989, 2002].

### 2.2.2 Evolutionary Strategy

ES is one of the early types of EAs designed mainly to deal with continuous domain optimization. A typical ES is specified by two parameters $\lambda$ and $\mu$ which respectively determine the population size and the parents size after applying selection. The main evolutionary operator in this algorithm is mutation, applied together with the selection operator to continue the search. Since the size of the search space in continuous optimization is infinite, ES algorithms usually use smaller population sizes and larger number of generations to evolve the solutions toward the optimum. More details about this type of EAs can be found in [Schwefel, 1995; Beyer, 2001; Beyer and Schwefel, 2002].

### 2.2.3 Genetic Programming

The objective of GP is to evolve functions or computer programs to obtain a desired functionality. The main difference of GP and GA is in the way that the solutions are represented. The usual representation used to encode the solutions in GP is tree structures, where operations are shown as intermediate nodes and operands as terminal nodes of the tree. GP evolves a population of these trees in the general framework of EAs, trying to generate programs that can better achieve the required functionality. In a broader perspective, GP can be used for automatic generation of new content.

To deal with program-tree representation of solutions, the genetic operators used to reproduce new solutions should be adapted accordingly. Crossover usually involves switching the compatible branches of two solutions. In mutation the values of specific tree nodes or a branch of the tree is changed (while respecting the compatibility of the whole solution). Therefore, the solutions in the population can have different sizes. GP and EP are closely related and usually used interchangeably, with the latter putting more emphasis on mutation in the generation of new solutions. The interested reader is referred to the series of books by Koza [Koza, 1992, 1994; Koza et al., 1999, 2003].

### 2.2.4 Complementary Methods

In order to improve the performance of EAs in optimization, several methods have been proposed which modify or add to the general framework of EAs. Here we briefly introduce two of these methods that have been employed to enhance the optimization performance of EAs in some of the applications discussed later in this thesis.

#### Hybridization

Hybridization of an algorithm usually refers to the case where this algorithm is used in conjunction with a different type of method, and thus can cover various types of hybridization between different algorithmic frameworks. The most likely type of hybridization for EAs, is to use a local search method for improving new solution reproduction, which is sometimes referred to as memetic algorithms [Hart et al., 2005]. In these algorithms after generating a new solution using genetic operators, its local neighborhood is searched for fitter solutions using a local search method like hill climbing. This type of hybridization can improve the exploitation ability of EAs in search for optimal solution(s).

#### Cooperative Coevolution

Co-evolutionary algorithms are an extension to the original EAs and are specially designed for optimization problems in complex systems. In coevolution, the fitness of each solution is determined by the way it interacts with other solutions in the population. Basically, two types of coevolution can be considered: competition and cooperation. In competitive coevolution [Stanley and Miikkulainen, 2004], the increase in the fitness of an individual negatively affects the fitness of other solutions. Cooperative coevolution, on the other hand, rewards those solutions that have better collaboration with other solutions [Potter et al., 1995]. In this type of coevolution, usually the problem is decomposed into a number of subproblems and the individuals in the population represent sub-solutions to

these subproblems. Therefore these subsolutions need to cooperate with each other to obtain complete solutions with higher fitness values. The sub-solutions can be either evolved in different populations or in a single population, known as Parisian approach [Ochoa et al., 2008].

## 2.3 Estimation of Distribution Algorithms

A key to the success of EAs is the identification, preservation and effective combination of the fitter partial solutions to the problem during evolution [Harik et al., 1999a]. However, it has been shown that the operators used in traditional EAs fail to properly accomplish this task when certain characteristics are present in the problem. A main reason for this shortcoming is that these algorithms do not properly consider the dependencies and relationships between the variables of the problem, and are not able to thoroughly exploit the information obtained so far, up to the current stage of the search, in order to speed up convergence. There are properties like non-linearity, ill-conditioning and deception in real world problems that without considering them, traditional EAs can have significant challenges for optimization.

Probabilistic modeling offers a systematic way of acquiring this kind of regularities, and therefore can help to achieve a quick, accurate and reliable problem solving [Goldberg, 2002; Pelikan et al., 2002]. EDAs try to overcome the shortcomings of traditional EAs by incorporating probabilistic modeling. For this purpose, instead of genetic operators used in traditional EAs, new candidate solutions to the problem in each iteration are generated using the following two steps:

1. Estimating a probabilistic model based on the statistics collected from the set of candidate solutions, and

2. Sampling the estimated probabilistic model.

In this way the problem regularities encoded in the probabilistic model are used when generating new solutions. Algorithm 2.1 shows the basic steps of an EDA. Worthy of note is that probabilistic models can also be used in EAs for other purposes. For instance, to make decisions on the application of mutation operators, to assess the influence of the different EA parameters on the algorithm behavior, to obtain an estimation of solution qualities without direct fitness evaluation, or even to implement local optimization procedures.

Implicitly, EDAs assume that it is possible to model the promising areas of the search space, and to use this model to guide the search for the optimal solution(s). The probabilistic model learnt in EDAs captures an abstract representation of the features shared by the selected solutions and encodes the different patterns of interactions between subsets of the problem variables. Probabilistic modeling gives EDAs an advantage over other non-model based EAs by allowing them to deal with problems containing important interactions among their variables. This, together with their capacity to solve different types of problems in a robust and scalable manner [Pelikan, 2005; Lozano et al., 2006], has popularized these algorithms, which are sometimes even referred to as competent GAs [Goldberg, 2002; Pelikan et al., 2002] to differentiate them from traditional GA

Estimation of Distribution Algorithm

**Inputs:**

A representation of solutions,

An objective function $f$

$P_0 \leftarrow$ Generate initial population according to the given representation

$F_0 \leftarrow$ Evaluate each individual $\boldsymbol{x}$ of $P_0$ using $f$

$g \leftarrow 1$

**while** termination criteria not met **do**

  $S_g \leftarrow$ Select a subset of $P_{g-1}$ according to $F_{g-1}$ using a selection mechanism

  $\hat{\rho}_g(\boldsymbol{x}) \leftarrow$ Estimate the probability of solutions in $S_g$

  $Q_g \leftarrow$ Sample $\hat{\rho}_g(\boldsymbol{x})$ according to the given representation

  $H_g \leftarrow$ Evaluate $Q_g$ using $f$

  $P_g \leftarrow$ Replace $Q_g$ in $P_{g-1}$ according to $F_{g-1}$ and $H_g$

  $F_g \leftarrow$ Update $F_{g-1}$ according to the solutions in $P_g$

  $g \leftarrow g + 1$

**end while**

**Output:** The best solution in $P_{g-1}$

Algorithm 2.1: The basic steps of an estimation of distribution algorithm

algorithms. The successful application of EDAs to many real-world problems in different domains like: machine learning Inza et al. [2000, 2001a], bioinformatics Armañanzas et al. [2008]; Santana et al. [2010c]; Armañanzas et al. [2011], scheduling Zhang and Li [2011]; Wang and Fang [2012]; Chen and Chen [2013], industrial design and management Sun et al. [2008]; Jiang et al. [2006], protein folding Santana et al. [2007, 2008], software testing Sagarna and Lozano [2006] and composite materials Grosset et al. [2006] have proved their usefulness in practice. There are also many EDA implementations available online, providing a range of possibilities for the application of EDAs to real-world problems [Santana, 2011].

Because of the different nature of both optimization and probabilistic modeling in discrete and continuous domains, EDAs developed for each of these domains also have differences depending on the representation type they use for the problem. Therefore, in the following sections each of these two categories are discussed separately. Here, we do not intend to give an exhaustive list of all proposed EDAs. Rather the aim is to review the different probabilistic models and machine learning methods employed in EDAs. Another common way of categorizing EDAs is by the complexity of the probabilistic models they use. In general, one of the rationales in EDA development has been to find a satisfactory trade-off between the complexity of the probabilistic models they use and how accurately these models represent particular optimization problem characteristics. This is another factor taken into account in reviewing EDAs here. Moreover, for readability we use the algorithm acronyms. Table 2.1 lists the algorithms full names. For recent review papers on EDAs, the interested reader is referred to [Hauschild and Pelikan, 2011; Larrañaga et al., 2012].

### 2.3.1 Discrete EDAs

Early EDAs were developed for discrete and especially binary domains, as it is a common practice in EAs to represent problem solutions with bit strings. *Univariate EDAs*, such as PBIL [Baluja, 1994], cGA [Harik et al., 1999b] and UMDA [Mühlenbein and Paaß, 1996], assume that all variables are independent and thus their joint probability can be factorized as a product of univariate marginal probabilities. The probabilistic model in this case consists of separate nodes containing the probability distribution for each of the problem variables. While some algorithms (e.g., PBIL and UMDA) learn the model from a population of solutions, others (e.g., cGA) update the model using only a few individuals. Consequently, these algorithms are the simplest EDAs and thanks to their simplicity, univariate EDAs are particularly suitable for the theoretical analysis of EDA behavior [González et al., 2002; Zhang, 2004].

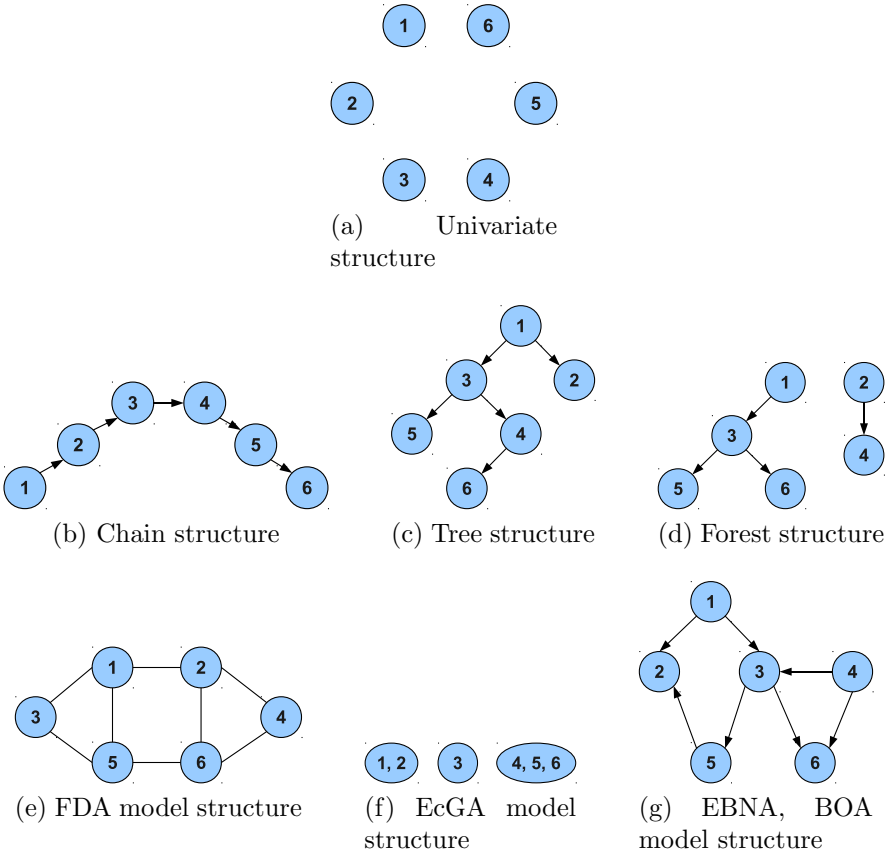
To extend the modeling capability of EDAs, *bivariate models* were used in EDAs. Bivariate models can represent pairwise dependencies between variables using efficient learning methods. MIMIC [De Bonet et al., 1997] uses a chain structured probabilistic model where the probability distribution of all the variables except the head node is conditioned on the value of the variable preceding them in the chain. The structure of the probabilistic model in COMIT [Baluja and Davies, 1997] is a tree, while it is generalized to a forest of trees (dependency graph) in BMDA [Pelikan and Mühlenbein, 1999].

In univariate and bivariate EDAs, the probabilistic model structure is either fixed or is very restricted. Therefore, while they can be efficiently applied to separable problems (without any dependency) or to problems with low degrees of dependency among the variables, they might still rapidly lose their efficiency when applied to more complicated problems, with larger number of variable interactions. A further attempt to improve EDAs is to use models that can capture dependencies between an arbitrary number of variables. Thus the joint probability distribution can be decomposed into factors involving several variables of the problem. Of course, this more flexible modeling by *multivariate EDAs*, capable of learning complex structures, comes at the cost of a greater computational effort. Figure 2.2 shows some examples of possible model structures of different complexity learnt by EDAs.

#### Multivariate EDAs

FDA [Mühlenbein and Mahnig, 1999; Mühlenbein et al., 1999] gives a factorization of the joint probability distribution for a class of problems known as additively decomposable functions. EcGA [Harik et al., 2006] factorizes the joint probability distribution into a number of marginal distributions defined over non-overlapping subsets of variables in a probabilistic model called marginal product model (MPM). An MDL scoring metric is used to search for the proper partitioning of the variables.

EBNA [Etxeberria and Larrañaga, 1999] and BOA [Pelikan et al., 1999] learn a BN from the selected set of solutions in every generation. While both of the algorithms use a greedy local search method to explore the space of possible BN structures, EBNA measures the quality of the structures using the BIC metric and BOA utilizes the BDe metric to score them. BOA is also further extended to hierarchical BOA [Pelikan, 2005] by incorporating diversity-preserving techniques and an improved representation for BN parame-




(a) Univariate structure

(b) Chain structure

(c) Tree structure

(d) Forest structure

(e) FDA model structure

(f) EcGA model structure

(g) EBNA, BOA model structure

Figure 2.2: Examples of different types of model structures used in EDAs.

ters using decision graphs. An improved version of FDA, known as learning FDA, is also proposed that uses BNs to dynamically learn the interdependent variables [Mühlenbein and Mahnig, 1999]. Thanks to the powerful probabilistic model that these algorithms use, they can be applied to solve many difficult problems [Larrañaga et al., 2000b; Pelikan and Hartmann, 2006].

Because of the model learning complexity in general, MN-based EDAs [Santana, 2003; Wang and Wang, 2004; Shakya, 2006; Alden, 2007] are usually applied to applications where the structure of the optimization problem is known and can be easily represented using an undirected graphical model. However, an approximation of the probability distribution, like Kikuchi approximations [Santana, 2005], can also be estimated to obtain the factorization of problem variables. The use of this type of probabilistic models for optimization is still topic of active research and very recently Shakya and Santana [2012] have reviewed the new developments in this type of EDAs.

EDNA [Gámez et al., 2007] uses dependency networks [Heckerman et al., 2001] to model the problem structure. Based on a heuristic approximation, this algorithm uses second-order statistics (similar to bivariate EDAs) for model learning. Dependency networks can represent cyclic dependencies between variables which cannot be encoded in BNs. However, this property prevents the application of sampling techniques used for BNs like PLS method. Therefore, similar to MN-based EDAs, EDNA uses relatively complex Gibbs sampling [Geman and Geman, 1984] procedures to generate new solutions from

the probabilistic model.

### Other Modeling Types

Apart from basic probabilistic models, EDAs have also used other modeling techniques. *Mixture of models* increases the flexibility of joint probability estimation in EDAs, especially for solving multi-modal optimization problems (containing several optima). Pelikan and Goldberg [2000] studied the effect of clustering the set of selected solutions on the performance of UMDA. UEBNA [Peña et al., 2005] represents the mixture with a BN which is learnt by the structural EM algorithm. Santana et al. [2009b] discussed several topics concerning model building in discrete EDAs.

A promising technique is to take into account the fitness information of individuals in the modeling process. EBCOA [Miquélez et al., 2004] adopts this idea, first introduced by Michalski [2000], by dividing the set of selected individuals into different classes according to their fitness values. The probabilistic model of this algorithm can be any of the BN classifiers (Section 1.3.4) with the class node corresponding to the different levels of fitness values (after discretization). Similarly, Shakya and McCall [2007] directly used the fitness values in the estimation of MN parameters in their proposed DEUM framework [Shakya, 2006], assuming that the probability of each solution in the search space can be approximated by its fitness value. Another related idea is to learn EDA probability distributions from both low and high fitness individuals [Hong et al., 2009]. Valdez-Peña et al. [2009] have extended this idea by estimating a distribution for selection operators and used it to identify which solutions would be used for model learning.

Probabilistic models in EDAs can also be used to estimate or predict the fitness values of the new solutions. This approach can be especially useful for problems with a very difficult and time-consuming fitness function. Fitness inheritance modeling has been incorporated into EcGA [Sastry et al., 2004] and BOA [Pelikan and Sastry, 2004] to estimate the fitness value of new individuals and, consequently, reduce the total number of function evaluations consumed by the algorithm to reach the optimum. Brownlee et al. [2008] and Brownlee [2009] used a MN-based fitness model in DEUM to predict the fitness value of the new individuals and also compare their correlations to the true fitness values.

Propagation methods, used for inference in BNs, also have applications in EDAs. Mendiburu et al. [2007] used total abductive inference to find the most probable value-setting of BN variables, as one of the solutions generated from the probabilistic model in EBNA to improve the sampling procedure. Lima et al. [2009] used loopy belief propagation in a local search method in BOA as a way to compute optimal local value-settings of the problem. An analysis of EBNA performance at different stages of evolution is also given by computing the most probable value-setting at each generation [Echegoyen et al., 2009]. AffEDA [Santana et al., 2010b] uses affinity propagation [Frey and Dueck, 2006], another propagation algorithm based on probabilistic modeling, to obtain non-overlapping factorizations of the joint probability distribution.

There have been also attempts to hybridize EDAs with other optimization algorithms, like differential evolution [Sun et al., 2005] and artificial immune systems [de Castro and Zuben, 2009]. Recently, Bengoetxea and Larrañaga [2010] and Ahn et al. [2012] have proposed very similar frameworks for hybridizing EDAs with particle swarm optimization, in continuous and discrete domains respectively. The central idea of these techniques is to

combine the global search ability of EDAs with better local exploitation of other methods. LTGA [Thierens, 2011] takes advantage of structure learning in EDAs and genetic recombination in GAs by building a linkage tree using an agglomerative hierarchical clustering algorithm with mutual information as its distance metric. Using this linkage tree the algorithm determines the crossing point of the parents when applying the crossover operator to the selected solutions for generating new solutions.

### 2.3.2 Continuous EDAs

The usual choice, adopted by most EAs in continuous domain optimization, is to assume a Gaussian distribution for problem variables. Many of the early continuous EDAs as well as their recent improvements are also based on this assumption. $\text{PBIL}_\text{C}$ [Sebag and Ducoulombier, 1998] extends its discrete version to continuous domains by updating a vector of independent Gaussian distributions. $\text{UMDA}_\text{C}$ [Larrañaga et al., 1999, 2000a] uses ML estimation to learn the parameters of the Gaussian distribution for each variable from the population of solutions. $\text{MIMIC}_\text{C}$ [Larrañaga et al., 1999, 2000a] learns the chain structured probabilistic model for continuous variables by adapting the concept of (conditional) entropy for univariate and bivariate Gaussian distributions.

EGNA [Larrañaga et al., 1999, 2000a; Larrañaga and Lozano, 2001] can be considered as the continuous version of EBNA based on a GBN. Two approaches have been proposed for learning the network structure in this algorithm: (i) starting from a complete DAG, likelihood ratio hypothesis tests are used to decide whether the edge between two nodes should be excluded from the network; (ii) performing a greedy local search in the space of possible DAGs using a scoring metric like BGe (continuous version of the BDe metric) or BIC. EBCOA has also been extended to continuous domains [Miquélez et al., 2006] by building BN classifiers that assume Gaussian distributions for the variables given the class variable value. Karshenas et al. [2011b] proposed learning a joint GBN consisting of both variables and objectives in their JGBN-EDA for multi-objective optimization.

IDEA [Bosman and Thierens, 2000a,b] and EMNA [Larrañaga and Lozano, 2001] learn an MGD from the set of selected solutions. While EMNA uses ML estimation, IDEA employs Kullback-Leibler divergence in conjunction with a greedy search algorithm, as well as likelihood ratio statistical hypothesis tests. Further improvements of IDEA have been proposed by scaling the diminishing variances [Grahl et al., 2006] and shifting the distribution mean [Bosman and Grahl, 2008; Bosman et al., 2008].

#### Mixture of Distributions

An extended version of IDEA [Bosman and Thierens, 2001] uses a mixture of normal distributions over clusters of solutions, obtained by applying a clustering algorithm before learning mixture components. rBOA [Ahn et al., 2004] first learns a GBN to obtain a decomposition of the problem variables into smaller subproblems. Then, a separate mixture of GBNs is learnt for each of the subproblems by clustering the solutions in that subproblem. In BGMMEDA [Li et al., 2006], instead of clustering the samples, a boosting technique is applied to estimate a Gaussian mixture model.

MB-GNG [Martí et al., 2011] adopts growing neural gas, a specific single-layer neural network, to determine the location of the components of the mixture of Gaussian distri-

butions. This model learning algorithm is sensitive to, and therefore does not neglect, outliers and is able to automatically adapt its topology while decreasing the accumulated error of the network nodes. The multi-model EDA framework [Weise et al., 2011] extends these mixture methods by applying traditional EA recombination operators to the individual models learnt for each of the clusters in order to improve search space exploration. RM-MEDA [Zhang et al., 2008] learns a piece-wise continuous manifold for multi-objective optimization using the local principle component analysis algorithm. Each model component is a hyper-rectangle with an added Gaussian noise.

### Other Modeling Approaches

Pošík [2008, 2009] proposed the use of *Cauchy* distribution for the purpose of preventing premature convergence. Since the moments of an $n$-dimensional variable with multivariate Cauchy distribution are not defined, the mean vector and covariance matrix of an MGD are computed instead. For sampling new solutions, the scaling factor of the Cauchy distribution is used to obtain isotropically distributed new solutions.

More recently some EDAs have employed *copula* theory to relax the Gaussian assumption for the variables. Copula-based EDAs (CEDAs) [Salinas-Gutiérrez et al., 2009; Wang et al., 2009; Wang and Zeng, 2010; Cuesta-Infante et al., 2010] use the copula function for estimating the joint probability distribution of the variables according to Sklar's theorem. The copula function only uses the marginal univariate probabilities to compute the joint probability distribution. This reduces the computational complexity of model learning. Two-dimensional elliptical copulas as well as Archimedean and empirical copulas and their extensions to higher dimensions are studied in the literature. These copula functions provide the variable interaction structure of the problem when sampling new solutions from the learnt model. In each generation the algorithm selects or constructs a copula function after estimating the univariate marginal distributions and then, generates new samples according to the copula distribution.

CMA-ES [Hansen, 2006] incorporates model estimation into ES. The algorithm learns an MGD as its probabilistic model to generate new solutions. The probabilistic model estimated in each generation is a combination of information collected over several generations, taking into account the path that the optimizer has traversed in the search space (the gradient information). Instead of estimating a new probabilistic model in each generation, the algorithm *adapts* the model during evolution. Thus, the algorithm is able to use smaller population sizes for optimization by spanning model learning over several generations. Because of such an adaptation strategy, some researchers do not completely consider this algorithm as an EDA [Pošík, 2009]. It is worth to note that similar techniques have been proposed for improving the efficiency of other EDAs in optimization [Pelikan et al., 2008; Bosman et al., 2008].

### Non-parametric Probabilistic Models

Other probabilistic models that estimate a non-parametric probability distribution for the variables have also been used in continuous EDAs. IDEA, for example, has employed other models, apart from MGD, like Gaussian kernel distribution (a Gaussian kernel for each sample) or histograms in its framework [Bosman and Thierens, 2000b,a].

Cho and Zhang [2002] proposed a continuous EDA that learns a mixture of factor analyzers using EM algorithm. They also employed a more complicated mixture of variational Bayesian independent component analyzers in a later study [Cho and Zhang, 2004]. MOPEDA [Costa and Minisci, 2003] applies a Parzen estimator that convolves the empirical estimation obtained from a finite data set with a squared integrable kernel function in order to reduce the variance of the probability distribution estimation. Both Gaussian and Cauchy kernels are used alternatively during evolution to utilize their intrinsic complementary characteristics. In KEDA [Luo and Qian, 2009], the width of each kernel is dynamically computed during the optimization.

*Histogram*-based EDAs (HEDAs) discretize each variable's values by dividing their range to a number of bins. Tsutsui et al. [2001] proposed two types of marginal histogram models: (i) a fixed-width histogram (FWH) where the domain of each variable is divided into a fixed number of bins whose height may differ depending on the variable values; (ii) a fixed-height histogram (FHH) where all bins have an equal value generation probability but can have different widths. Consequently, there will be more bins in denser regions and thus modeling will be more accurate.

Ding et al. [2008] proposed two improvements to the above histogram modeling in their HEDA. They introduced a surrounding effect, where the values of each bin can affect the values of its surrounding bins using a special surrounding factor. They also employed a shrinkage strategy where the height of the bin containing the best value of the variable can exceed a predefined threshold. $\text{PBIL}_\text{C}$ is also extended with histograms [Xiao et al., 2009], combining the original updating rule with bin updating, where the bins reaching a predefined height are divided.

Histogram modeling has also been applied to optimization in permutation domains [Tsutsui, 2002; Tsutsui et al., 2006] using two different types of models. The first is an edge histogram matrix where each entry indicates the frequency of two permutation values occurring adjacent to each other in the population. The second is a node histogram matrix that encodes the frequency at which a special value in the permutation occurs at a specific location in the solution. Specific sampling algorithms are developed for these models where a new value is generated according to the value of adjacent permutation locations or the position for which the value is going to be generated.

### 2.3.3 Discrete-Continuous EDAs

MBOA [Očenášek and Schwarz, 2002; Očenášek et al., 2004] adopts binary classification and regression decision trees to solve mixed discrete-continuous optimization problems. The algorithm uses a BDe-like scoring metric to build a decision tree for each variable to encode its related probability distribution. The decision trees allow the algorithm to build individual models (like Gaussian kernels) for specific regions of the search space, stored in different tree leaves.

### 2.3.4 Discussion

Table 2.1 gives a summary of the presented EDAs and their probabilistic models. The algorithms are divided into three different classes according to the complexity of their probabilistic models: univariate, bivariate and multivariate EDAs. Within each class,

Table 2.1: Full name of EDAs and the models they use. Discrete EDAs are shown on a white, continuous on a green and mixed discrete-continuous on a blue background, respectively.

| | Algorithm | Complete Name | Model Used |
|---|---|---|---|
| Univariate | PBIL | Population Based Incremental Learning | - |
| | UMDA | Univariate Marginal Distribution Algorithm | - |
| | $PBIL_C$ | Continuous PBIL | - |
| | cGA | Compact Genetic Algorithm | - |
| | $UMDA_C$ | Continuous UMDA | - |
| | HEDA | Histogram-based EDA | Marginal Histograms |
| Bivariate | MIMIC | Mutual Information Maximizing Input Clustering | Chain |
| | COMIT | Combining Optimizers with Mutual Information Trees | Tree |
| | BMDA | Bivariate Marginal Distribution Algorithm | Forest |
| | $MIMIC_C$ | Continuous MIMIC | Chain |
| | CEDA | Copula-based EDA | Copula Functions |
| Multivariate | FDA | Factorized Distribution Algorithm | Factor Graph |
| | EBNA | Estimation of Bayesian Network Algorithm | Bayesian Network |
| | BOA | Bayesian Optimization Algorithm | Bayesian Network |
| | EGNA | Estimation of Gaussian Network Algorithm | Gaussian Bayesian Network |
| | IDEA | Iterated Density Estimation Evolutionary Algorithm | Gaussian Markov Network |
| | EMNA | Estimation of Multivariate Normal distribution Algorithm | Gaussian Markov Network |
| | MBOA | Mixed BOA | Decision Graphs |
| | MN-FDA | Markov Network based FDA | Markov Network |
| | MOPEDA | Multi-Objective Parzan-based EDA | Mixture of Kernels |
| | rBOA | Real-coded BOA | Gaussian Bayesian Network |
| | EBCOA | Evolutionary Bayesian Classifier based Optimization Algorithm | Bayesian Network Classifiers |
| | MN-EDA | Markov Network EDA | Markov Network |
| | UEBNA | Unsupervised EBNA | Bayesian Network |
| | EcGA | Extended cGA | Marginal Product Model |
| | BGMMEDA | Boosting Gaussian Mixture Model based EDA | Mixture of Gaussian Markov Networks |
| | DEUM | Distribution Estimation Using Markov Random Fields | Markov Network |
| | CMA-ES | Covariance Matrix Adaptation Evolutionary Strategy | Gaussian Markov Network |
| | MARLEDA | Markovian Learning EDA | Markov Network |
| | EDNA | Estimation of Dependency Network Algorithm | Dependency Network |
| | RM-MEDA | Regulatory Model-Based Multi-Objective EDA | Mixture of hyperplanes |
| | KEDA | Kernel density-based EDA | Mixture of Gaussian Kernels |
| | AffEDA | Affinity propagation EDA | Marginal Product Model |
| | MB-GNG | Model Building Growing Neural Gas | Mixture of Gaussian Markov Networks |
| | LTGA | Linkage Tree GA | Hierarchical Dependency Tree |
| | JGBN-EDA | Joint GBN-based EDA | Gaussian Bayesian Network |

the algorithms are ordered chronologically to show how the use of probabilistic models in EDAs has evolved during time.

Initial EDAs mainly considered about the probability distribution of individual variables, in order to perform a more effective search for the solutions of separable problems. In this kind of problems the optimal value of each variable can be obtained regardless of the value of the other variables. In other terms, the way that the value of each variable influences the fitness of the whole solution is not affected by the values of other variables at all.

The limitations of univariate EDAs, brought up the need for a more advanced probabilistic modeling. The main advantage of these new probabilistic models is that they consider a kind of structure for the problem, reflecting an estimation of the interactions between variables. Some algorithms put certain constraints on the structures, e.g. bivariate EDAs can only consider mutual interactions. Some others like FDA consider a fixed structure given beforehand (e.g. for a specific class of problems), and try to find the best parameter estimation that fits this structure. But most of EDAs try to learn the structure dynamically during evolution. A number of algorithms require the variables to be clustered into completely disjoint dependence groups (e.g. EcGA and AffEDA), whereas in others overlapping groups of dependent variables is allowed (e.g. MIMIC and EBNA).

Because of their ability to represent complex patterns of interactions between the problem variables, the use of multivariate probabilistic models has become dominant in EDAs. Usually these algorithms perform a kind of *structure learning* to estimate the probabilistic model, which is very time-consuming in comparison to other parts of the algorithm. Therefore, in practice an upper bound is imposed on the order of interactions that is considered in the structure learning. This restriction can also be imposed implicitly, e.g. using penalized scoring metrics for learning BNs (Section 1.3.3).

The choice of the type of EDA to be used, depends very much on the problem. If the problem at hand is linear, or the variables are not believed to be strongly dependent, then one should use univariate EDAs since they are computationally more efficient. On the contrary, if we are dealing with a problem that has high order of interactions between its variables, then EDAs that use probabilistic models with higher representational capability should be used in order to be able to reach the optimal solution(s) of the problem. The structures estimated by these EDAs can also give a better understanding of unknown problems. Several works have studied the accuracy of these structures and the information we can obtain from them [Lima et al., 2007; Karshenas et al., 2009; Santana et al., 2009a].

In reality, one should compromise between the computational complexity and the optimization capability of these algorithms when applying them to different problems. Based on this observation, there has been many efforts to increase the efficiency of EDAs while keeping their complexity at an acceptable range. Techniques like parallelization and hybridization (discussed in Section 2.2.4) are introduced in the literature which are usually referred to as efficiency enhancement techniques [Pelikan, 2005].

### 2.3.5 Model-Based Genetic Programming

Although probabilistic models were first built into GAs, the idea was soon adopted also in GP. Because of the complex solution representation used in GP, model learning and sam-

pling can be a challenging task. However, several GP algorithms based on probabilistic modeling have been proposed in the literature.

Probabilistic incremental program evolution (PIPE) [Sałustowicz and Schmidhuber, 1997] is a GP algorithm based on univariate factorization of program distribution. A probabilistic prototype tree (PPT) model encodes the probability distribution and is later used to generate new program trees at each generation of the algorithm. Extended compact genetic programming (ECGP) [Sastry and Goldberg, 2003] incorporates the use of marginal product distributions into the context of GP. Also based on a PPT model, ECGP constructs a factorization of the tree program distribution equal to the product of marginal distributions. Each marginal distribution is associated with a subtree of the PPT. The structure of the factorization is learned using a greedy algorithm, similar to EcGA.

The use of BNs for GP was proposed by Yanai and Iba [2003]. This estimation of distribution programming approach is based on the use of the PPT. The conditional probabilities between the nodes of the PPT are computed for the purpose of representing a wider class of probability distributions than PIPE and ECGP. More recently, Hasegawa and Iba [2008] proposed a BN modeling approach for GP that significantly reduces the size of the conditional probability tables. The algorithm also requires fewer samples to construct the BN from the selected solutions.

Another type of model-based GP algorithms involve the use of grammars. These algorithms [Shan et al., 2006; McKay et al., 2010; Bosman and de Jong, 2008] depart from the traditional uses of PGMs since the probability distributions are often associated with the grammar rules and their different contexts of application. For example, Bosman and de Jong [2008] estimate the distribution of programming trees based on the subtrees that actually occur in the data. The representation specifies a set of rules whose expansion leads to trees, and the probability distributions are defined on these rules. For a review of these algorithms, see [Shan et al., 2006] and [McKay et al., 2010].

## 2.4 Conclusions

One of the disciplines that has greatly taken advantage of probabilistic modeling is evolutionary computation, resulting in a new class of algorithms which are called estimation of distribution algorithms. Although this is a relatively new paradigm, numerous studies have investigated their different aspects, and several types of algorithms have been proposed based on this paradigm. These algorithms cover both discrete and continuous domains, and within each domain probabilistic models with different complexities have been used.

EDAs are still a topic of intensive research, and every year many new works related to the theory or application of these algorithms are published. New studies are trying to extend the application of these algorithms to other domains like multi-objective, noisy or dynamic problems. Nevertheless, because of the close relationship that these algorithms have with probabilistic modeling, any new development in the area of probabilistic modeling, specially regarding the efficiency of the learning and sampling methods, can help to achieve competent problem optimization with EDAs.

# Chapter 3

# Evolutionary Algorithms in Bayesian Network Inference and Learning

## 3.1 Introduction

Some of the most relevant inference and learning problems in Bayesian networks are formulated as the optimization of a function. These problems usually have an intractable complexity and therefore are a potential domain for the application of meta-heuristic methods. In the previous chapter we discussed how PGMs and in general probabilistic modeling is used in EA framework to improve the optimization performance in EDAs. However, this collaboration has not been all one-sided, and EAs have also been extensively applied to solve many optimization tasks in PGMs and especially in BNs. In this chapter we review how EAs have been applied for solving some of the combinatorial problems existing in the inference and learning of BNs. This review is published in [Larrañaga et al., 2013].

## 3.2 Triangulation of the Moral Graph

Lauritzen and Spiegelhalter [1988] proposed one of the most popular algorithms for exact inference in BN. The first step of this algorithm is to moralize the BN structure. In this step all variables with a common child are linked together and then all edge directions are removed. The resulting graph is called a moral graph. The second step of the algorithm is the so-called *triangulation* of the moral graph. A graph is triangulated if any cycle of length greater than 3 has a chord. This step is considered as the hardest step (in terms of computational complexity) of this inference algorithm. The resulting structure is then used for evidence propagation and probability computation.

The basic technique for triangulating a moral graph (see also Figure 3.1) is through successive elimination of graph nodes. Before eliminating a node and its incident edges, we check that all of its adjacent nodes are directly connected to each other by adding the required edges to the graph. The nodes are chosen for elimination according to a given order of the variables. The quality of a triangulation is measured by the weight of the

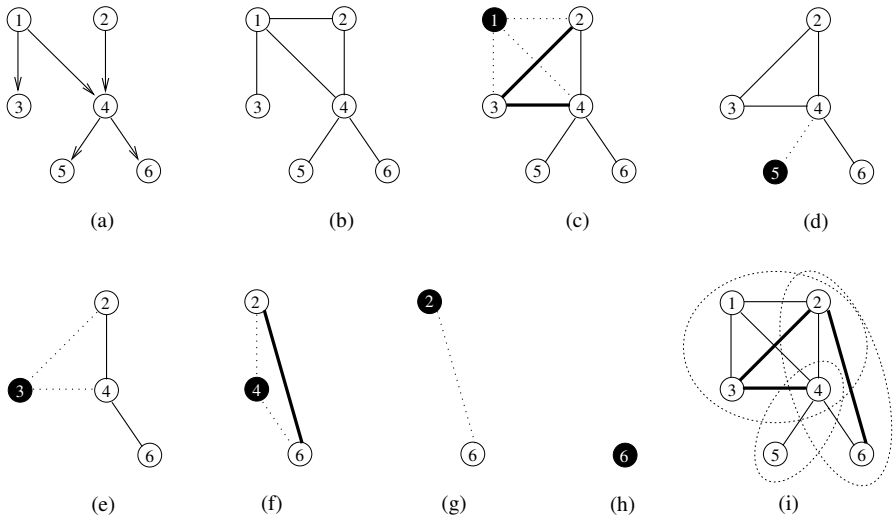



Figure 3.1: An example of the triangulation algorithm. Nodes are eliminated in order: $X_1, X_5, X_3, X_4, X_2, X_6$ and it is assumed that $r_i = i + 1$, $i = 1, \ldots, 6$. a) Initial DAG. b) Related moral graph. c) Eliminate $X_1$: $\boldsymbol{C}_1 = \{X_1, X_2, X_3, X_4\}$, added edges: $< X_2, X_3 >$ and $< X_3, X_4 >$. d) Eliminate $X_5$: $\boldsymbol{C}_2 = \{X_4, X_5\}$. e) Eliminate $X_3$: $\boldsymbol{C}_3 = \emptyset$. f) Eliminate $X_4$: $\boldsymbol{C}_4 = \{X_2, X_4, X_6\}$, added edges: $< X_2, X_6 >$. g) Eliminate $X_2$: $\boldsymbol{C}_5 = \emptyset$. h) Eliminate $X_6$: $\boldsymbol{C}_6 = \emptyset$. i) Total weight of the triangulated graph: $\log_2(2 \cdot 3 \cdot 4 \cdot 5 + 5 \cdot 6 + 3 \cdot 5 \cdot 7) = \log_2 255$.

triangulated graph $\mathcal{S}^t$

$$w(\mathcal{S}^t) = \log_2 \left( \sum_{\boldsymbol{C} \in \mathcal{C}} \prod_{X_i \in \boldsymbol{C}} r_i \right), \tag{3.1}$$

where $\boldsymbol{C}$ denotes a clique of the triangulated graph $\mathcal{S}^t$ composed of vertices $X_i$, each with $r_i$ possible different states. Evidently, the quality of triangulation is fully determined by the order in which the nodes are eliminated. Hence, the search for an optimal triangulation is equivalent to the search for an optimal node elimination sequence, i.e. the search for an optimal permutation of nodes. Several criteria are proposed to search for the optimal node elimination order, from which most of the works try to minimize the weight of the corresponding triangulated graph (Equation (3.1)). It is already demonstrated that the search for an optimal triangulation is NP-hard [Wen, 1991]. Kjærulff [1992] performed an empirical comparison of several triangulation methods, obtaining the best results with the simulated annealing algorithm.

The problem of searching for an optimal node elimination sequence resembles the much researched traveling salesman problem (TSP). The aim of both problems is to find an optimal variable ordering. One important difference, however, is that only the relative order is important in the standard TSP, whereas the absolute order also matters in the node elimination problem. Taking these ideas, Larrañaga et al. [1997] applied a GA with crossover and mutation operators adapted for the TSP path representation. They achieved competitive results compared to simulated annealing, which is the best method to date.

More sophisticated recombination operators are a way to enhance the search for opti-

Table 3.1: Application of evolutionary algorithms to inference in Bayesian networks

| Task | Reference | Representation | Algorithm |
|---|---|---|---|
| Triangul. | Larrañaga et al. [1997] | Permutation of Variables | GA |
| | Wang et al. [2006] | Permutation of Variables | GA |
| | Romero and Larrañaga [2009] | Permutation of Variables | REDA |
| | Dong et al. [2010b] | Permutation of Variables | GA |
| MPE | Gelsema [1995] | Value-setting for Variables | GA |
| | Rojas-Guzmán and Kramer [1996] | Graph | GP |
| | Mengshoel [1999] | Value-setting for Variables | GA |
| | Sriwachirawat and Auwatanamongkol [2006] | Value-setting for Variables | GA |
| MAP | de Campos et al. [1999] | Value-setting for Variables | GA |
| | de Campos et al. [2001] | Value-setting for Variables | UMDA, MIMIC, EBNA |

mal variable ordering. Wang et al. [2006] proposed an adaptive GA able to self-adapt the crossover and mutation operators probabilities, and provided a ranking-based selection operator that adjusts the pressure of selection according to the population evolution. Recently, Dong et al. [2010b] proposed a new GA based on a new rank-preserving crossover operator and a two-fold mutation mechanism that utilizes the minimum fill weight heuristic.

Another alternative to improve search efficiency for this problem has been to use probabilistic modeling. Romero and Larrañaga [2009] proposed an approach based on recursive EDAs (REDAs) for both discrete and continuous representation of the variables. REDAs partition the set of vertices (that are to be ordered) into two subsets. In each REDA call, the vertices in the first subset are fixed, whereas the other subset of variables is evolved with a standard EDA. In the second call, the subsets switch roles.

The research on this problem so far has shown that GAs can obtain results comparable to simulated annealing. A very close behavior is seen when using REDAs, with improved convergence speed. The comparison with other types of optimization algorithms that use other optimization criteria also show that GAs which use the minimizing graph weight as the optimization criterion can find better node elimination orders, provided that proper operators and parameters are used.

## 3.3 Total and Partial Abductive Inference

EAs have also been used to search for the MPE in a BN. Gelsema [1995] used a GA where each individual is a value-setting for the unobserved variables, i.e., a string of integers. Rojas-Guzmán and Kramer [1996] employed a GP where each individual represents the whole BN with all the nodes in the explanation set instantiated to one of their possible states. Mengshoel [1999] used a GA coupled with his proposed probabilistic crowding replacement to perform a more efficient search for the MPE by better preservation of the diversity. Sriwachirawat and Auwatanamongkol [2006] proposed a GA for solving the more complex problem of finding the $k$ MPEs [Nilsson, 1998].

De Campos et al. [1999] proposed a GA for approximate partial abductive inference (MAP) given an evidence set. The individuals in the GA population represent a possible

value-setting only for the variables in the explanation set. The proposed algorithm is also able to find the $k$ MPEs of the explanation set. Discrete EDAs with different degrees of model complexity (UMDA, MIMIC and EBNA) are also used to find the MAP [de Campos et al., 2001].

The common trend in finding the $k$ MPEs is to search in the space of possible value-settings for unobserved variables. The reported results show that if these value-settings are represented on the original BN structure, better results can be obtained in less time using evolutionary search. GAs can reach high probable explanations faster than conventional methods like max-flow propagation [Nilsson, 1998]. Furthermore, the probabilistic modeling of EDAs can speed up the convergence compared to GA, especially when using probabilistic models with high descriptive abilities (e.g. EBNA) [de Campos et al., 2001]. Table 3.1 summarizes the discussed algorithms for BN inference tasks.

## 3.4 Structure Search

Finding the correct BN structure is an important part of the learning process in the search+score strategy which also directly affects BN parameter learning. Heuristic search algorithms and especially EAs can be a promising approach to this problem as the cardinality of the search space is huge (Equation (1.9)). Although most of the proposed algorithms search the *space of DAGs*, there other spaces where the search can be conducted. One of these spaces is the *space of variable orderings* (permutations). Given a total ordering of variables, a simple greedy method, called the K2 algorithm [Cooper and Herskovits, 1992] can be used to obtain a full DAG structure of a BN. With different variable orderings, this algorithm can result in different BN structures. Thus, the space of variable ordering can be searched to obtain the orderings that result in higher scoring BNs.

The K2 algorithm is one of the first algorithms proposed for learning a BN from a dataset. This algorithm uses the K2 metric to score different BN structures (Section 1.3.3). The K2 metric is computed for each variable separately to take advantage of decomposability. Starting from an empty parent set for each node, the variable that results in the highest increase of the node's scoring metric is added to its parent set. This greedy process is repeated until the network score reaches its maximum. Candidate parents are selected according to the variable ordering given to the algorithm. This means that parents can only include variables that have preceded a variable in the given ordering. The algorithm also restricts the maximum number of parents allowed for each node (also given as an input parameter).

Besides the previous two spaces, another possibility is to search the *space of equivalence classes* of BNs [Chickering, 2002], when the scoring metric complies with the equivalence property. Two DAGs are said to be Markov equivalent if they encode the same statistical model, i.e., the same set of conditional independence assertions. This model can be represented by a partial DAG (PDAG), where some of the edges are undirected. A metric that assigns equal scores to Markov equivalent BNs is said to comply with the equivalence property. Using this algebraic relation (which is reflexive, symmetric and transitive), the space of equivalence classes can be searched for the best BN. The BDe metric, for example, is a Markov equivalence-compliant scoring metric.

In this section, we review some of the EA-based methods proposed for BN structure search. Note that we focus our review on the EAs introduced in Chapter 2, which are more relative to the line of research in this thesis, although some other evolutionary approaches like ant colony optimization have also been used for this task [de Campos et al., 2002]. The reviewed methods are divided into three categories depending on the space where they perform the search for finding the best BN structure. The methods are also listed in Table 3.2.

### 3.4.1 DAG Space

Larrañaga et al. [1996c] proposed a GA that encodes the connectivity matrix of the BN structure in its individuals. The algorithm, which uses a marginal likelihood-based metric to score the BN structures, considers two different approaches. In the first approach there is a total ordering assumption between the variables (parents before children), and thus the variation operators (one-point crossover and bit mutation) are closed. This restriction also reduces the cardinality of the search space. In the second approach, there is no such assumption, and the algorithm should deal with a larger space. In this case, a repairing operator is needed to ensure that the variation operators result in a valid BN structure.

To overcome the requirement for a repairing operator, Etxeberria et al. [1997] used the fuse-DAGs algorithm [Matzkevich and Abramson, 1992] to guarantee that the crossover operator satisfies the closure property. Larrañaga et al. [1996b] hybridized two versions of a GA with a local search operator to obtain better structures. Myers et al. [1999] extended the use of GAs for BN learning to domains with missing data, simultaneously evolving the structure of the BN and the missing data in separate populations. At each generation the new solutions generated in both populations are used to compute the BDe score of each network structure.

Cotta and Muruzábal [2002] built phenotypic information into gene-based and allele-based recombination operators in a GA to search for the best structure according to a penalized likelihood-based scoring metric. Using guidelines on how GAs work [Harik et al., 1999a], van Dijk et al. [2003] designed a GA where the recombination operator tries to prevent the disruption of the good BN substructures obtained so far in the population. The algorithm uses an MDL metric as the fitness function for scoring the network structures, and a repairing operator to ensure that structures are acyclic. In the domains with mixed discrete-continuous variables, Mascherini and Stefanini [2005] proposed a mixed GA to search for the optimal CGN, where invalid structures are corrected by deleting inadmissible arcs at random. An extension of the BDe metric is used to measure the fitness of the model for the mixed domain dataset.

Blanco et al. [2003] compared the performance of GAs with two univariate EDAs, namely, UMDA and PBIL, using three different scoring metrics. The reported results, both with and without a total ordering assumption between variables, showed that EDAs are able to obtain better or comparable BN structures. Kim et al. [2005] used fitness sharing in an EA to obtain a diverse population of BN structures. The BNs learnt at the end of evolution are then combined according to Bayes' rule for providing a more robust inference.

Hanzelka [2008] also proposed a hybridization of GA with local search methods performed on single solutions under the term of *Lamarckian evolution*. It uses a Chi-squared

test to determine the edges that should be removed for repairing the structure. After GA terminates, an additional exhaustive search is conducted in the most promising subspace of BN structures found.

Barrière et al. [2009] proposed an EA which uses a cooperative coevolution strategy to evolve a population of conditional independence assertions. The scoring criteria is the Chi-squared test. At the end of evolution, the best conditional independence assertions found (partly stored in an archive) are used to build the structure of the BN.

The EP algorithm proposed by Wong et al. [1999] is based on a set of mutation operators and uses the MDL metric to search for good BN structures. Because of its flexibility in representing the structures without any encoding, no further assumptions on the ordering of the variables are needed in order to apply the mutation operators. The proposed algorithm is later extended by first introducing a merge operator and then hybridizing it using the two-phase constraint-based method [Wong and Leung, 2004]. In the first phase, conditional independencies among the variables are used to reduce the size of the DAG search space. In the second phase, good BN models are searched for using an EA. Replacement of the EA with a cooperative coevolution algorithm is also studied in [Wong et al., 2004].

Most of the works that consider learning BNs by searching in the space of possible DAG structures use a string representation of the connectivity matrix. In this representation the order of variables is important or else a repair operator will be necessary to ensure valid DAG structures after applying genetic operators. Because of this, some methods simultaneously search for variable orderings and topology of BN whereas some others use structure-aware operators to ensure the validity of the resulting DAGs. A similar representation is the list of parents of each variable, leading to solutions of varying sizes. If GP is used for search, DAG structures can be directly evolved and the reported results show better performance of this approach, in terms of the final BN structure score, its closeness to the reference structure and computational time needed for the search [Wong et al., 1999].

Another point is the importance of local search or in general higher exploitation which is shown to find better BN structures. Significant improvement has been reported when the initial search space is reduced by incorporating information about the conditional independencies between variables [Wong and Leung, 2004]. These information is gathered in a pre-evolution phase by performing conditional independence tests, usually with a small order of variables in the condition part to keep the computational complexity of the whole algorithm low. Comparison of different EAs with some standard methods like K2 algorithm or simple deterministic methods like hill climbing show that, especially when the size of the datasets used for learning BNs increases, EAs are able to estimate better structures and usually have a faster convergence.

### 3.4.2 Equivalence Class Space

To eliminate redundancy in the DAG space, van Dijk and Thierens [2004] extended their initial representation to PDAGs to perform the search in the equivalence class space. They also studied the effect of hybridizing the algorithm with local search methods. Jia et al. [2008] proposed an immune GA to search this space, hybridizing principles of artificial immune systems (based on immunology) [de Castro and Timmis, 2002] with GAs. They

employed conditional independence tests for extracting the interactions between variables prior to the evolutionary search of GA and use it (as immune vaccines) in order to reduce the search space.

Muruzábal and Cotta [2004] proposed an EP algorithm to perform the search in the equivalence class space. The algorithm uses several mutation operators to move between Markov equivalent classes [Chickering, 2002] according to BDe metric. Cotta and Muruzábal [2004] compared three versions of EP algorithm that perform the search in the equivalence class space, either directly or with a restriction (inclusion boundary [Castelo and Kočka, 2003]) on the operators.

Two milestone works have paved the way for most other approaches proposed to search the equivalence space: the equivalence class aware operators that allow moving between different classes when applied to any PDAG member of that class; and the inclusion boundary property of the operators that when preserved, can prevent the search from falling into local optima. The greedy search in this space results in faster convergence compared to the search in the DAG space. However, it should be noted that the size of the search space is still exponential in the number of variables. Moreover, many of the EAs that are proposed for performing the search in this space involve operations to convert PDAGs back and forth to/from DAGs which is computationally expensive. Hybridizing EAs with local search has also been reported to improve the results [van Dijk and Thierens, 2004].

### 3.4.3 Ordering Space

Larrañaga et al. [1996a] used the TSP-inspired permutation representation (Section 3.2) to search for the best ordering between the variables using a GA. The K2 algorithm was applied on each ordering to evaluate the quality of different orderings. They compared the performance of different combinations of crossover and mutation operators. Using the same representation and evaluation scheme, Habrant [1999] proposed improved mutation and crossover operators to search for the best BN structure in the real-world problem of time series prediction in finance. Similarly, Hsu et al. [2002] proposed a GA based on the *order crossover* operator to search in the space of permutations. In their method, the fitness of each BN resulting from an ordering after applying the K2 algorithm is measured according to its inference quality (using cross-validation).

The chainGA [Kabli et al., 2007] assumes a chain structure between the variables in the given ordering and evaluates them using the K2 metric in order to bypass the need for the time-consuming K2 algorithm. At the end of evolution however, the K2 algorithm is applied to the best found orderings to obtain a good structure. The algorithm is also applied to the real-world problem of prostate cancer management [Kabli et al., 2008]. Lee et al. [2008] proposed a novel representation of BN structure composed of *dual* chromosomes: a node ordering chromosome and a connectivity matrix chromosome in accordance with its dual (ordering). They applied their proposed GA, with special crossover and mutation operators developed for this representation, to a number of real-world problems which involve learning BNs.

Romero et al. [2004] applied two types of discrete- and continuous-encoded EDAs (UMDA and MIMIC) to obtain the best ordering for the K2 algorithm. For discrete encoding they used a bijective mapping to represent possible orderings of $n$ variables with

Table 3.2: Application of evolutionary algorithms to learning Bayesian networks.

| Space | Reference | Representation | Algorithm |
|---|---|---|---|
| DAGs | Larrañaga et al. [1996c] | Connectivity Matrix | GA |
| | Larrañaga et al. [1996b] | Connectivity Matrix | GA+Local Search |
| | Etxeberria et al. [1997] | Connectivity Matrix | GA |
| | Myers et al. [1999] | Connectivity Matrix | GA |
| | Wong et al. [1999] | Graph | EP |
| | Tucker et al. [2001] | Edge-Time Tuples | EP |
| | Cotta and Muruzábal [2002] | Connectivity Matrix | GA |
| | van Dijk et al. [2003] | Connectivity Matrix | GA |
| | Blanco et al. [2003] | Connectivity Matrix | GA, PBIL, UMDA |
| | Tucker et al. [2003] | Set of Spatial points | GA |
| | Wong and Leung [2004] | Connectivity Matrix | EP |
| | Wong et al. [2004] | Connectivity Matrix | Cooperative Coevolution |
| | Kim et al. [2005] | Connectivity Matrix | GA |
| | Mascherini and Stefanini [2005] | Connectivity Matrix | GA |
| | Jia et al. [2005] | String of Possible Parents | Immune GA |
| | Ross and Zuviria [2007] | String of Possible Parents | Multi-Objective GA |
| | Hanzelka [2008] | Connectivity Matrix | GA+Local Search |
| | Barrière et al. [2009] | Connectivity Matrix | Cooperative Coevolution |
| PDAGs | Muruzábal and Cotta [2004] | Graph | EP |
| | Cotta and Muruzábal [2004] | Graph | EP |
| | van Dijk and Thierens [2004] | Connectivity Matrix | GA+Local Search |
| | Jia et al. [2008] | Connectivity Matrix | Immune GA |
| Orderings | Larrañaga et al. [1996a] | Permutation | GA |
| | Habrant [1999] | Permutation | GA |
| | Hsu et al. [2002] | Permutation | GA |
| | Romero et al. [2004] | Permutation | UMDA, MIMIC |
| | Kabli et al. [2007] | Chain Permutation | GA |
| | Lee et al. [2008] | Permutation+Connectivity Matrix | GA |

$n-1$ random variables. The sampling step of EDAs is accordingly adapted to output a valid permutation of the variables. This adaptation is not necessary for continuous encoding, where each $n$-dimensional real vector can be transformed into a valid permutation of the $n$ variables.

An important decision to make when performing the search in the space of orderings is how to evaluate different candidate orderings. Most of the proposed methods use the K2 algorithm for this purpose which results in a high fitness evaluation cost. Approximating the quality of a solution with a less computationally expensive method, can greatly reduce the overall running time of the algorithms (e.g. as in chainGA). However, this can also cause the quality of the learned BNs to reduce. The reported results on datasets with small number of variables has shown that the evolutionary search with GA obtains results comparable to those of exhaustive search of all possible orderings, while only visiting a small percentage of the whole solution space.

## 3.5 Learning Dynamic Bayesian Networks

BNs have also been used to model time series data. Basically, a natural choice for modeling time series data is to use directed graphical models which can appropriately capture the forward flow of time. If all arcs of the probabilistic model are directed, both

within and between different time slices while the structure is unchanged, the resulting model is called a *dynamic Bayesian network* (DBN) [Murphy, 2002]. Several works have used EAs to learn DBN structures from data. Tucker et al. [2001] use an EP algorithm to seed the population of a GA which then evolves the structures of DBNs. Ross and Zuviria [2007] use a multi-objective GA, where the multi-objective criteria are network likelihood and structural complexity scores. Tucker et al. [2003] propose evolutionary and non-evolutionary operators for learning BN structures from spatial time series data. Jia et al. [2005] apply their immune GA for learning DBNs.

Many of the methods proposed for learning the structure of normal BNs can be adapted to learn DBNs. Simplifying assumptions, e.g. no edges between the nodes in the same time slice, can greatly reduce the computational complexity of learning DBNs. The BNs learned with simple GAs are not so satisfactory, sometimes worse than hill climbing. Because of this, complementary techniques have been employed in the reported works, including generation of non-random initial population with GP, incorporation of additional operators in the recombination process (e.g. the add vaccine of immune GA), and performing a multi-objective search instead of single objective search to take into account several criteria when learning DBNs. All of these complementary techniques have been reported to yield better DBNs in terms of the scoring metric or structural accuracy compared with simple GA.

## 3.6 Learning Bayesian Network Classifiers

Finding an appropriate subset of predictor variables by removing redundant and irrelevant variables can be remarkably helpful for classification using BNs. EAs are one of the search techniques extensively used for this task, which is usually called *feature subset selection*. Liu et al. [2001] used GAs to search for an optimal subset of predictor variables for their improved NB classifier, whereas Inza et al. [2000] applied EBNA for feature subset selection in a number of different datasets. They also compared the proposed method with two GA-based and two greedy search algorithms [Inza et al., 2001a]. PBIL and COMIT are used for feature selection in the problem of predicting the survival of cirrhotic patients, and the results are compared with two versions of GA [Inza et al., 2001b]. Blanco et al. [2004] used EDAs for gene selection in cancer classification problem using an NB classifier.

The reported results show that both GA and EDA versions perform better than simple deterministic hill climbing algorithms like forward selection and backward elimination. A comparison between EDAs also show that using more powerful probabilistic models allow selecting better feature subsets, which for many of the tested data sets are also better than the feature subsets found by GA.

Robles et al. [2003] used EDAs in their interval estimation NB classifier to search for proper class and conditional probabilities within their respective confidence intervals. An EDA with a continuous representation is used to search for the best combination of probability values in these intervals. They also used EDAs to improve the search for new hybrid variables in the SNB classifier [Robles et al., 2004]. A comparison with standard techniques like "forward selection and joining of variables", or "backward elimination and joining of variables" [Pazzani, 1996] show that EDA-based search and joining of variables

Table 3.3: Learning Bayesian classifiers with evolutionary algorithms.

| | Reference | Classifier | Algorithm |
|---|---|---|---|
| FSS | Inza et al. [2000] | Naïve Bayes | EBNA |
| | Liu et al. [2001] | Naïve Bayes | GA |
| | [Inza et al., 2001a] | Naïve Bayes | EBNA, GA, Greedy Search |
| | Inza et al. [2001b] | Naïve Bayes | PBIL, COMIT, GA |
| | Blanco et al. [2004] | Naïve Bayes | EDA |
| Classification | Sierra and Larrañaga [1998] | Markov Blanket | GA |
| | Sierra et al. [2001] | Markov Blanket | GA |
| | Robles et al. [2004] | Naïve Bayes, Semi-Naïve Bayes | EDA |
| | Kline et al. [2005] | General Bayesian Network | GA |
| | Flores et al. [2007] | Naïve Bayes | $\text{UMDA}_\text{C}$ |
| | Zhu et al. [2007] | Markov Blanket | GA |
| | Reiz et al. [2008] | TAN | GA |
| | Dong et al. [2010a] | TAN | GA |

can find significantly better classifiers, though at a higher computational complexity.

Flores et al. [2007] proposed the use of $\text{UMDA}_\text{C}$ to search for the optimal discretization of all predictor variables simultaneously for the NB classifier. Reiz et al. [2008] employed Prüfer numbers to encode TAN classifiers and search for the optimal structure using GAs. AIC, BIC and Hannan-Quinn information criteria were employed as fitness measures. Dong et al. [2010a] designed genetic operators to evolve TAN classifier structures with the objective of maximizing the likelihood function.

Sierra and Larrañaga [1998] used GAs to search for the optimal MB of the class variable in a real-world classification problem. They compared the resulting MB-based classifiers with NB classifier and a Bayesian classifier learned by likelihood maximization and show that the MB-based classifiers have higher classification accuracy. This method was also employed in a multi-classifier schema to classify intensive care unit patient data [Sierra et al., 2001]. Zhu et al. [2007] proposed an MB-embedded GA for gene selection in microarray datasets and showed that using GA to search for the MB of the class variable results in higher classification accuracy. Kline et al. [2005] applied GAs to search for the most accurate BN structure to predict venous thromboembolism according to gathered data. Also in the field of BN-based clustering, Peña et al. [2004] applied UMDA to search for the optimal dependency structure between predictor variables in unsupervised learning using a specific representation of BNs.

Table 3.3 shows these methods along with the classifiers and EAs they used.

## 3.7 Conclusions

BNs are an important class of PGMs, which have proven to be very useful and effective for reasoning in uncertain domains. They have been successfully used in machine learning tasks like classification and clustering. They have been studied at length over the last three decades and many methods have been proposed to automate their learning and inference. Nevertheless, many of these methods involve difficult combinatorial search

problems that directly affect their widespread use, especially for large problem instances, and thus they require advanced search techniques like meta-heuristics.

EAs are general-purpose stochastic search methods inspired from natural evolution and have been frequently applied to solve many complex real-world problems. Because of their advantages, different types of EAs have been used in BN learning and inference tasks. A wide range of tasks like triangulation of the moral graph in BN inference, abductive inference, BN structure learning in difference search spaces, BN classifier learning and learning dynamic BNs from data streams have employed evolutionary algorithms, which has led to significant improvements in the computational time and performance of these tasks.

This topic is still an active field of research and with the intrinsic complexity of BN tasks, EAs are always a potential competitor. Especially, EDAs with their ability to account for the interactions between variables seem to be a promising approach for further study. So far, several works have empirically compared the conventional approaches to BN tasks (e.g. see [Tsamardinos et al., 2006] for a comparison between several BN structure learning methods). An interesting future work that can complement this review is to perform similar empirical comparisons of the evolutionary approaches presented here, on standard datasets.

# Part II

# Regularization-Based Continuous Optimization

# Chapter 4

# Introduction to Regularization

## 4.1 Introduction

Estimating the parameters of mathematical models or solutions to systems of equations is a common task in statistics and machine learning problems. Methods like least squares error or ML estimation are commonly used to obtain unbiased solutions to these problems, and there are theoretical studies justifying these methods. However, in certain situations, e.g. when the problem at hand is ill-posed [Tikhonov and Arsenin, 1977; Neumaier, 1998], the application of these methods is not possible or can result in unstable solutions which are highly sensitive to the changes in the training dataset. This directly affects the generalization of these solutions and the wide-spread use of the machine learning techniques based on them.

Regularization is an approach to improve the estimation of solutions (e.g. model parameters) by reformulating the model estimation problem. Regularization-based model estimation attempts to decrease the generalization error of the solutions (i.e. estimated models) by reducing their high variance at the cost of introducing a little bias into the model, and therefore obtaining a better bias-variance trade-off in model estimation [Stein, 1956; Efron, 1982]. The approach is applicable considering both the expected or the worst-case generalization error [Sugiyama et al., 2004]. In addition to obtaining numerically stable solutions, i.e. robust estimation, this approach allows solving problems which would be otherwise unsolvable. An example is small-sample model estimation where the number of parameters to be estimated is more than the number of observations in the training dataset, also known as "large $n$, small $N$" or "$n \gg N$" problems. Estimating covariance matrices from high-dimensional datasets is one of these problems where employing the commonly-used ML estimator can result in ill-conditioned estimations.

There are many works, especially in the statistics community, studying the application of regularization techniques for model estimation. These techniques have been applied to the estimation of different types of models like kernels [Sugiyama et al., 2004; Guo et al., 2008], probability distributions [Dudík et al., 2007; Fraley and Raftery, 2007], linear regression [Tibshirani, 1996; Schmidt, 2005; Hesterberg et al., 2008; Beck and Eldar, 2008], logistic regression [Wainwright et al., 2006; Koh et al., 2007; Ravikumar et al., 2010], log-linear models [Andrew and Gao, 2007], generalized linear models [Friedman et al., 2010a] and graphical models, especially under Gaussian distribution assumption [Schäfer and Strimmer, 2005a; Li and Gui, 2006; Kramer et al., 2009]. One of the application

domains frequently used to evaluate Gaussian graphical model estimations is discovering the associations between genes in genetics, a problem characterized by a large number of genes (i.e. variables) and a small number of observations. In the following sections, we take a closer look at the use of regularization in the estimation of regression and graphical models, and introduce some of the methods proposed in the literature. It has been shown that under a Gaussian distribution assumption, these two types of regularized models are closely related [Witten and Tibshirani, 2009].

## 4.2 Regularized Regression Models

Consider a simple linear regression model for estimating the values of a continuous response (output) variable $Y$, given a set of $n$ predictor (input) variables $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon,$$

where $\epsilon$ is a homoscedastic zero-mean Gaussian noise: $\mathcal{N}(0, \sigma^2)$. Given a set of $N$ observations of the form $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ is a value-setting for the variable vector $\boldsymbol{X}$ and $y_i$ is the corresponding response value, ordinary least squares (OLS) estimation of the response variable $(\hat{Y})$ minimizes the sum of squared errors between the predicted value and the actual value:

$$\arg\min_{(\beta_0, \boldsymbol{\beta})} \left( \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2 \right) = \arg\min_{(\beta_0, \boldsymbol{\beta})} \left( \sum_{i=1}^{N} \left(y_i - \left(\beta_0 + \boldsymbol{\beta}\boldsymbol{x}_i^{\mathrm{T}}\right)\right)^2 \right), \tag{4.1}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$. This kind of estimation has often shown bad prediction accuracy and interpretability [Tibshirani, 1996]. The decomposition of the expected prediction error of the estimation shows that despite its low bias, this type of model fitting often has a large variance of prediction accuracy. Moreover, the interpretability of the models is especially important when dealing with a large number of variables where only a strong subset of dependencies are required to appear in the model.

Regularization techniques try to improve model estimations like that in Equation (4.1) by introducing a penalization term, imposed on the values of model parameters, denoted by $J(\boldsymbol{\beta})$. For example, the regularized OLS estimation is given by

$$\arg\min_{(\beta_0, \boldsymbol{\beta})} \left( \sum_{i=1}^{N} \left(y_i - \left(\beta_0 + \boldsymbol{\beta}\boldsymbol{x}_i^{\mathrm{T}}\right)\right)^2 + \lambda J(\boldsymbol{\beta}) \right), \tag{4.2}$$

where $\lambda > 0$ controls the amount of penalization. From a prediction accuracy point of view, the use of penalization term in the least squared error, will reduce the estimation variance at the cost of introducing some bias. Some of the most popular regularization techniques are as follows.

- Ridge regression [Hoerl and Kennard, 1970] with $J(\boldsymbol{\beta}) = \sum_{j=1}^{n} \beta_j^2$. This penalization term (also called an $\ell_2$-regularization term) causes the regression parameters to shrink towards zero, although they do not become exactly zero.

- LASSO (Least Absolute Shrinkage and Selection Operator) [Tibshirani, 1996] or $\ell_1$-regularization with $J(\boldsymbol{\beta}) = \sum_{j=1}^{n} |\beta_j|$. This type of penalization term has the appealing property of setting some of the regression parameters exactly equal to zero, which will result in a behavior similar to that of variable selection and hence the name [Hastie et al., 2009].

- Elastic net [Zou and Hastie, 2005], which is a combination of the previous two terms, i.e. $J(\boldsymbol{\beta}) = \sum_{j=1}^{n} \left(\alpha\beta_j^2 + (1-\alpha)|\beta_j|\right)$, where $\alpha \in (0,1)$ controls the combination of the two previous regularization terms. This type of regularization has a grouping effect where correlated variables will be in or out of the model together. A difference between elastic net and LASSO penalties is that with elastic net penalty the maximum number of variables which can be selected in the estimated model is not limited to at most the number of observations ($N$). Therefore, this kind of regularization is especially useful for problems with a large number of variables and a small number of observations for model estimation.

In all of the above penalization terms, since the intercept parameter ($\beta_0$) can be estimated by the average absolute value of the responses ($y_i$), it is usually left out, and without loss of generality we can assume it is equal to zero.

An important parameter that affects the outcome of regularized model estimation is the value of the regularization parameter $\lambda$. Larger values of this parameter impose a higher penalization on the model parameters and therefore more parameters will be shrunk toward zero whereas smaller values (those close to zero) allow larger values for the model parameters. Although in some cases there are theoretical proposals [Ledoit and Wolf, 2004] for selecting the value of this parameter or the bounds of an effective value, one should usually use a trial-and-error strategy to select the best value. A more general approach is to obtain the solutions for a range of possible $\lambda$ values. The solutions thus obtained, by varying $\lambda$ values, form the *regularization path* or the profile of the regularization technique. Having the entire path of solutions to model estimation, methods such as (k-fold) cross-validation, mean squared error, Mallows' $C_p$ statistic [Mallows, 1973], AIC [Akaike, 1974] or BIC [Schwarz, 1978] can be used to select one of the models.

Adding the regularization term changes how the model parameters (solutions of the model estimation) are computed. In the case of ridge regression, optimal values can be computed using a closed-form formula. However, the LASSO penalization will render the equation non-differentiable and thus there are no closed-form formulas for calculating the optimal solutions. Nevertheless, there are numerical optimization algorithms that can very efficiently compute the solutions along the whole regularization path and with the same computational cost as that of ridge regression [Hastie et al., 2009].

A well-known technique for regularized model estimation is the least angle regression (LARS) [Efron et al., 2004]. This method can be considered as an improvement of the forward stage-wise model selection [Hastie et al., 2009] which adds the variables one by one to the model. LARS is an iterative method in which the variables are added to the model but never removed from it. In each step, the (absolute) correlation between the current residual (error in estimation) and the predictor variables are used to choose which variables should next be added to the model. From a geometrical point of view, this corresponds to the angle between predictor variables and the response variable.

The LARS algorithm can be used, with a simple modification, to efficiently fit models

for LASSO [Efron et al., 2004] and elastic net [Zou and Hastie, 2005] estimations. The entire sequence of LARS steps with $n < N$ variables requires $O(n^3 + Nn^2)$ computations, which is the cost of an OLS model fitting with $n$ variables. If $n \gg N$, the computational cost is of order $O(N^3)$, since the algorithm will terminate at the saturated OLS fit after $N$ variables are added to the model.

The promising variable selection property of LASSO or $\ell_1$-regularization has encouraged its wide-spread use and several variants of this type of regularization have been proposed. Fused LASSO [Tibshirani et al., 2005] puts an additional penalty on the difference between parameter values so that the parameters of adjacent variables would have similar values. In group LASSO [Yuan and Lin, 2006] an $\ell_1$-regularization term is applied on predefined groups of parameters to reduce the number of variable groups included in the model. Smoothed LASSO [Meier and Bühlmann, 2007] is proposed for model estimation from time series data, where adjacent data points are believed to be more relevant than distant ones. To decrease the amount shrinkage in the parameters values towards zero, in adaptive LASSO [Zou et al., 2006] a weighted penalization is applied on the model parameters depending on the importance of the variables which is adapted according to the training dataset. A similar approach is the relaxed LASSO technique [Meinshausen, 2007] which first selects the subset of important variables using a standard LASSO estimation, and then another model is estimated only considering the selected variables and this time with a zero or very small value of the regularization parameter ($\lambda$). Hesterberg et al. [2008] reviewed many of these methods and discussed their properties.

## 4.3 Regularized Graphical Models

Many methods are proposed in the literature which use regularization techniques to obtain a better estimation of models with a graphical structure. Specifically, when using LASSO-based methods, one of the main motivations is the *sparsity* of the estimated graphical structures. MNs are one of the models for which several works have studied the use of regularization in their estimation. Considering the correspondence between the zero-pattern of precision matrix and the structure of GMRF, different algorithms are proposed to obtain a regularized estimation of MGD, usually called regularized Gaussian graphical model (GGM) estimation.

Some of these GGM learning methods directly regularize the estimation of the covariance matrix [Bien and Tibshirani, 2011] to obtain a sparser covariance graph model [Chaudhuri et al., 2007]. Others consider regularized estimation of the precision matrix [Schäfer and Strimmer, 2005a; Li and Gui, 2006; Yuan and Lin, 2007; Levina et al., 2008; Banerjee et al., 2008; Ravikumar et al., 2009], and study the properties of this type of GGM estimation, like its consistency. There are also works which have studied regularization in the estimation of both covariance matrix and its inverse [Bickel and Levina, 2008]. The efficiency of these methods is of special importance in order to allow their application to high-dimensional data [Yuan, 2008]. Friedman et al. [2010b] performed a comparison of several GGM estimation methods based on $\ell_1$-regularization. In a different approach, Chen et al. [2010] proposed a regularized version of the Tyler's method [Tyler, 1987] for covariance matrix estimation in elliptical distributions for application to

high-dimensional data.

Regularized estimates of GGMs are also obtained in a Bayesian framework, where using a special prior probability distribution on the model parameters, a MAP estimation is performed [Li and Yang, 2005; Marlin and Murphy, 2009]. In addition, regularization has been used to obtain a sparse estimation of covariance matrices from non-Gaussian data [Ravikumar et al., 2011], and also for the estimation of discrete MNs [Lee et al., 2007; Banerjee et al., 2008]. The use of regularization techniques for graphical model estimation is still a topic of active research and there are still several aspects of this approach like its efficiency or accuracy when applied to different types of high-dimensional problems which need a more thorough investigation. In the rest of this section we will introduce in more detail some of the methods for regularized GGM estimation which are used later in the next chapters of the thesis.

### Regularized Neighborhood Selection

Meinshausen and Bühlmann [2006] proposed a method to discover the conditional independence relationships of the precision matrix from a set of observations. The method computes the set of potential neighbors for each variable using regularized regression on other variables. They define the neighborhood set of a variable $X_j$ as the smallest subset of variables so that, when given, $X_j$ is conditionally independent of all remaining variables. To compute the complete dependency structure between all variables, $n$ distinct $\ell_1$-regularized regression models are estimated:

$$\arg \min_{(\beta_0,\boldsymbol{\beta})} \left( \sum_{i=1}^{N} \left( \boldsymbol{x}_{i<j>} - (\beta_0 + \sum_{\forall k \in J, k \neq j} \beta_k \boldsymbol{x}_{i<k>}) \right)^2 + \lambda \sum_{\forall k \in J, k \neq j} |\beta_k| \right), \tag{4.3}$$

where $J$ denotes the set of indices of the variables in $\boldsymbol{X}$.

Since the dependency between two variables is computed in two different regression models, they used two different strategies to decide about the existence of such dependency in the final structure. An AND strategy requires both variables to be present in the neighborhood set of each other, whereas an OR strategy will insert the dependency as soon as one of the variables appears in the other's neighborhood set. Based on a number of assumptions, they showed that this method can asymptotically obtain consistent sparse models for high-dimensional data.

A similar technique has also been applied in the estimation of a directed graphical structure for GBNs. In this approach, first an estimation of each variable's MB is obtained with $\ell_1$-regularized regression models. Then this information is incorporated in the search for BN structure within different search spaces [Schmidt et al., 2007; Vidaurre et al., 2010].

### Covariance Shrinkage

Schäfer and Strimmer [2005b] proposed a method for shrinkage estimation of covariance matrix. In shrinkage estimation an unrestricted high-dimensional model $\boldsymbol{S}$ (e.g. the ML estimation of the covariance matrix) is shrunk towards a restricted lower-dimensional target model $\boldsymbol{T}$ with fewer parameters (e.g. a diagonal covariance matrix where all off-diagonal elements are set to zero):

$$\boldsymbol{W} = \lambda \boldsymbol{T} + (1 - \lambda)\boldsymbol{S}, \tag{4.4}$$

where $\lambda \in [0, 1]$ denotes the shrinkage intensity and its optimal value can be analytically computed using a closed formula, assuming certain consistency conditions [Ledoit and Wolf, 2004]. When working with finite populations where these conditions cannot be assured, the shrinkage intensity can be estimated from the data by minimizing a mean squared error loss function.

Since there are many parameters in the high-dimensional model ($\boldsymbol{S}$) that need to be fitted, the estimation variance will be high. On the other hand, the variance of the estimation of fewer parameters in the low-dimensional model ($\boldsymbol{T}$) is lower but it is considerably biased with respect to the true model. It has been shown that the combination in Equation (4.4) is a systematic way to obtain a regularized estimate of the model that outperforms each of the individual estimators in terms of both accuracy and statistical efficiency.

This method has been successfully used for detecting gene associations networks from high-dimensional microarray data [Kramer et al., 2009; Yates and Reimers, 2009]. Schäfer and Strimmer [2005b] compared the structure recoverability of this method and that of the previous technique [Meinshausen and Bühlmann, 2006], when using $\ell_1$-regularization, on a number of synthesized covariance matrices. The reported results suggested that the true positive rate of the models built with the covariance shrinkage approach is considerably higher than those built with the neighborhood selection method, which tends to insert a lot of spurious dependencies to the structure.

### Graphical LASSO

Based on the previous works for regularized GGM estimation, Friedman et al. [2008] proposed a method to maximize the regularized log-likelihood estimation of an MGD:

$$\max_{\boldsymbol{\Theta}} \Big\{ \log\det(\boldsymbol{\Theta}) - \operatorname{trace}(\boldsymbol{S}\boldsymbol{\Theta}) - \lambda\|\boldsymbol{\Theta}\|_1 \Big\}, \tag{4.5}$$

where $\boldsymbol{\Theta}$ denotes the estimation of precision matrix and $\boldsymbol{S}$ is the empirical covariance matrix computed from the dataset. $\det(\cdot)$ is the determinant operator, $\operatorname{trace}(\cdot)$ gives the sum of the main diagonal elements of the input matrix, and $\|\cdot\|_1$ computes the sum of absolute values of the matrix entries.

To solve the optimization problem in Equation (4.5), its derivative is computed using element-wise sub-gradients (to overcome the non-differentiability of the absolute value function), and then set to zero. Thus, the problem is decomposed into a number of block-wise optimizations over rows (or columns) of $\boldsymbol{W} = \boldsymbol{\Theta}^{-1}$ and $\boldsymbol{S}$, partitioned as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_{11} & \boldsymbol{w}_{12}^{\mathrm{T}} \\ \boldsymbol{w}_{12} & w_{22} \end{bmatrix}.$$

$\boldsymbol{S}$ is partitioned similarly. It can be shown that each of these block-wise optimization problems is equivalent to an $\ell_1$-regularized OLS of the form

$$\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \left\| \boldsymbol{\beta}\boldsymbol{W}_{11}^{\frac{1}{2}} - \boldsymbol{s}_{12}\boldsymbol{W}_{11}^{-\frac{1}{2}} \right\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \right\}. \tag{4.6}$$

After estimating the parameters $\boldsymbol{\beta}$ of optimization problems for each row (column), the covariance matrix estimation is updated by setting $\boldsymbol{w}_{12} = \boldsymbol{\beta}\boldsymbol{W}_{11}$. The process is repeated until the estimated values for the entries of covariance matrix converge.

Instead of solving $n$ separate regularized regression problems, the graphical LASSO algorithm couples and solves these problems together in the same matrix $\boldsymbol{W}$. In fact, as the same matrix is used the information can be shared across the problems. The regularized neighborhood selection proposed in [Meinshausen and Bühlmann, 2006] can be seen as a special case of graphical LASSO approximation related to the case where $\boldsymbol{W}_{11} = \boldsymbol{S}_{11}$.

## 4.4 Conclusions

In order to decrease the generalization error of the estimated models, regularization techniques reformulate the model estimation problem by penalizing the values of the model parameters. This approach improves the stability of model estimations in the sense that they have a lower variance, though at the cost of a little bias. Regularization techniques are especially used for model estimation in high-dimensional domains, where the number of variables is larger than the number of observations in the data.

Regularization has been incorporated in the estimation of different types of models, and several regularization techniques have been proposed in the literature. One of these methods is regularization with an $\ell_1$ penalty, which causes some of the model parameters to become exactly zero, an appealing property which can be used for variable selection, and in general parsimonious model estimation. Several variants of this type of regularization have been proposed and applied for estimating different types of models, including GGMs, which will be used in the next chapters for probabilistic modeling in EDAs.

# Chapter 5

# Regularized Model Learning in EDAs

## 5.1 Introduction

The ultimate goal of the model learning step in EDAs is to estimate probabilistic models that assign higher probabilities to a neighborhood close to the optimal problem solutions (specified by the corresponding fitness function). This is based on the assumption that the regions of the search space with a higher probability are more likely to be sampled when generating a set of new solutions in EDAs. However, it is almost impossible to estimate such a probabilistic model directly at one go, especially for high-dimensional problems with a complex structure.

Iterative model learning and factorized estimation of the probability distribution are two main techniques employed to facilitate model learning in EDAs. If the model is estimated across several generations, the algorithm can visit more regions of the search space and gradually improve its estimation as, due to the limitation of computational resources, algorithms have to work with a finite population of solutions. Moreover, techniques like univariate or bivariate factorization, or more generally, multivariate BN learning, which imposes a factorization over problem variables, are able to estimate the joint probability distribution as the product of simpler factors.

Despite promising performance for solving many real-world problems, there are still shortcomings in the behavior of EDAs that have made them the topic of active research. Several studies have tried to analyze the behavior of EDAs [Mühlenbein et al., 1999; Pelikan et al., 2000; González et al., 2002; Zhang and Mühlenbein, 2004; Grahl et al., 2005; Yuan and Gallagher, 2009]. However, their results are mainly based on impractical assumptions or are limited to only specific problems and/or models. For example, convergence analysis of FDA is based on infinite population size assumption [Zhang and Mühlenbein, 2004], or only the univariate modeling in $\text{UMDA}_\text{C}$ is considered for the analysis of the optimization behavior on different fitness functions [Grahl et al., 2005; Yuan and Gallagher, 2009]. Especially in continuous domains, which is the scope of this chapter, there are many difficulties with model estimation that prevent EDAs from exhibiting the expected behavior.

The ability of the chosen probabilistic model to fit the solutions of a given problem, which is referred to as model capacity [Bosman and Grahl, 2008], can greatly affect

model estimation. Thanks to their good analytical properties, Gaussian distributions have been the probabilistic model of choice in most of continuous EDAs [Sebag and Ducoulombier, 1998; Bosman and Thierens, 2000a; Larrañaga et al., 2000a; Larrañaga and Lozano, 2001; Ahn et al., 2004]. However, a robust estimation of the Gaussian distribution relies on acquiring adequate statistics that are often not available from the population of continuous EDAs. This will usually cause EDAs to fall into premature convergence (or rather stalemate). To overcome this shortcoming, techniques like variance scaling [Yuan and Gallagher, 2005; Bosman and Thierens, 2007; Pošík, 2008] or eigenvalue resetting [Wagner et al., 2004; Dong and Yao, 2008] have been proposed in the literature.

Regularization techniques are widely used in statistics and machine learning to obtain a more robust estimation of probabilistic models with lower prediction error. Model estimation in EDAs has some characteristics that motivate the use of regularization techniques. Lack of adequate statistics can cause the estimated model to become highly biased to specific regions of the search space. This reduces the generalization ability of the estimated model which is an important factor affecting the generation of new solutions. The use of regularization can reduce the generalization error of the estimated model in EDAs. Another important issue is the EDA scalability with regard to problem size. Estimating the probability distribution model of huge search spaces requires large population sizes. Since the model estimation and sampling parts of EDAs are very time-consuming, algorithm efficiency will decline steeply for increasing population sizes, not to mention the memory constraints regarding large datasets. Being able to estimate a model of comparable quality using much smaller populations is a major requirement in these algorithms.

Very recently, regularization has been used in EDAs for discrete optimization. Yang et al. [2010] used regularized regression in the context of a BOA to obtain a reduced set of candidate parents for each variable before searching for the correct BN structure. Malagó et al. [2011] proposed the use of regularized logistic regression to learn the structure of the MN in the DEUM framework. In a different context, Karshenas et al. [2011a] studied some of the methods for integrating regularization techniques into the model estimation of continuous EDAs. In this chapter we analyze some of the methods to regularized model learning in EDAs for continuous optimization in high-dimensional settings. This study is published in [Karshenas et al., 2013b].

## 5.2 Approaches to Regularized Model Learning

We focus on continuous domain optimization, modeled with Gaussian distributions. For this reason, two approaches for employing regularization techniques in model learning are considered:

- The first approach is based on obtaining a regularized estimation of the dependency structure between variables, and uses this structure to estimate the covariance matrix of an MGD.

- The second approach applies techniques that directly obtain a regularized estimation of MGD.

The first approach, has to explicitly obtain a structure of variable dependencies. The method we consider here for structure estimation consists of three steps. In the first step, regularized regression models (Equation (4.2)) are used to determine the dependencies between each variable and the other variables. These models estimate the value of a variable given the value of other variables. To favor sparser structures, which can be of considerable help for model estimation in high-dimensional problems and thus allow for better EDA scalability, only those regularization techniques that result in explicit variable selection (i.e. LARS, LASSO and elastic net) are considered in this step.

Since the whole regularization path is computed for the regression model of each variable, in the second step, the regression solution (vector of regression coefficients) resulting in the lowest generalization error of the estimated model is selected. The variable dependency information obtained from $n$ independent regularized regression models is then combined in the third step to construct a single structure. Apart from the AND and OR strategies proposed by Meinshausen and Bühlmann [2006], a third strategy, denoted as "DAGS", is considered by searching the reduced search space constrained with variable dependencies [Schmidt et al., 2007].

Finally, the variable dependency structure learnt in this approach is used to obtain an estimation of the MGD. If the resulting structure is a DAG, the the final model will be a GBN. The local greedy search algorithm explained in Section 1.3.3 with BIC metric is used to learn the GBN within a search space constrained by the variable dependency structure. If an undirected graph is learnt as the structure, it can be transformed to a DAG [Lauritzen, 1996], be used as a dependency network [Heckerman et al., 2001], or be used as the zero pattern in the estimation of covariance matrix or its inverse [Chaudhuri et al., 2007; Bien and Tibshirani, 2011]. Here we adopt a simple algorithm proposed by Hastie et al. [2009, Section 17.3.1] to estimate the covariance matrix of MGD, whereas its mean vector is obtained using ML estimation. This algorithm obtains a constrained ML estimation of the MGD by adding Lagrange constants for all independencies imposed by the given structure.

For the second approach, the shrinkage estimation and graphical LASSO methods are considered for estimating an MGD[1]. The estimated probabilistic models in either of the approaches (with directed or undirected structures) will then be used in the sampling step of EDA to generate new solutions. In the case of models with directed structure, the PLS algorithm, and for models with undirected structure, the sampling algorithm explained in Section 1.4.3 is used to generate new solutions.

## 5.3 Analyzing Regularized Model Learning Methods

To better examine the methods introduced in the two approaches for regularized model estimation, they are evaluated from different perspectives in this section, comparing their merits and disadvantages. For this purpose, a number of reference Gaussian models with a predefined variable dependence structure are synthesized. Table 5.1 shows the details of these reference models. In these models, which all have 100 variables (except the tri-variate model which has 99 variables), the variables are organized in different blocks.

[1] In fact, these methods obtain an estimation of the covariance matrix, and the mean vector of MGD is obtained with ML estimation.

Table 5.1: Synthesized Gaussian distributions used as reference models.

| Name | Block Size | Num. Blocks | Num. Dependencies | Num. Independencies |
|---|---|---|---|---|
| Uni | 1 | 100 | 0 | 10000 |
| Bi | 2 | 50 | 100 | 9900 |
| Tri | 3 | 33 | 198 | 9603 |
| Quad | 4 | 25 | 300 | 9700 |
| Quint | 5 | 20 | 400 | 9600 |
| Vigint | 20 | 5 | 1900 | 8100 |

All variables within a block are correlated, whereas there is no inter-block dependency at all. The reference models differ as to the size of their blocks of dependent variables. They range from a block size of one (no dependency) in the univariate model to 20 in the vigint-variate model.

These reference Gaussian models are used to generate a population of solutions that will be used for model estimation with each of the regularization methods. In this study, a combination of $3 \times 3$ methods from the first approach and two methods from the second approach, namely the shrinkage estimation and graphical LASSO, are considered. Different combinations in the first approach result from considering each of the LARS, LASSO and elastic net regularization techniques with each of the AND, OR and DAGS merging strategies. In all of these combinations, the Mallows' Cp statistic is used for selecting the best regression solution in the regularization path of each variable's model. The $n/N$ ratio, where $N$ is the population size, is fixed to 10 for all populations generated from the reference models to resemble model estimation in a high-dimensional problem.

### 5.3.1 True Structure Recovery

Recovering the true structure of the probabilistic model is one of the requirements for a good model estimation algorithm. The structural accuracy of the estimated models can reveal the learning algorithm capability to capture the interdependencies between problem variables. One of the ways to measure the accuracy of structures learnt in the regularized model estimation methods is to compute the confusion matrix entries, i.e. the number of *true positive* (TP), *false positive* (FP), *false negative* (FN) and *true negative* (TN) links of the model structure. Here, we consider two well-known indicators computed using these measures:

- Sensitivity (TP/(TP+FN)): what percentage of the actual dependencies are correctly learnt.
- Specificity (TN/(TN+FP)): what percentage of the actual independencies are correctly learnt.

These indicators can give a clearer insight into the effects of different regularized estimation methods on the inclusion of spurious and excess links, or missing real dependencies. Figures 5.1 and 5.2 show, respectively, the sensitivity and specificity of the model structures estimated with the methods in the first approach. Since the univariate model actually does not have any dependencies, it is not included in the analysis of structure sensitivity. All the results are averaged over 30 independent runs. When the resulting

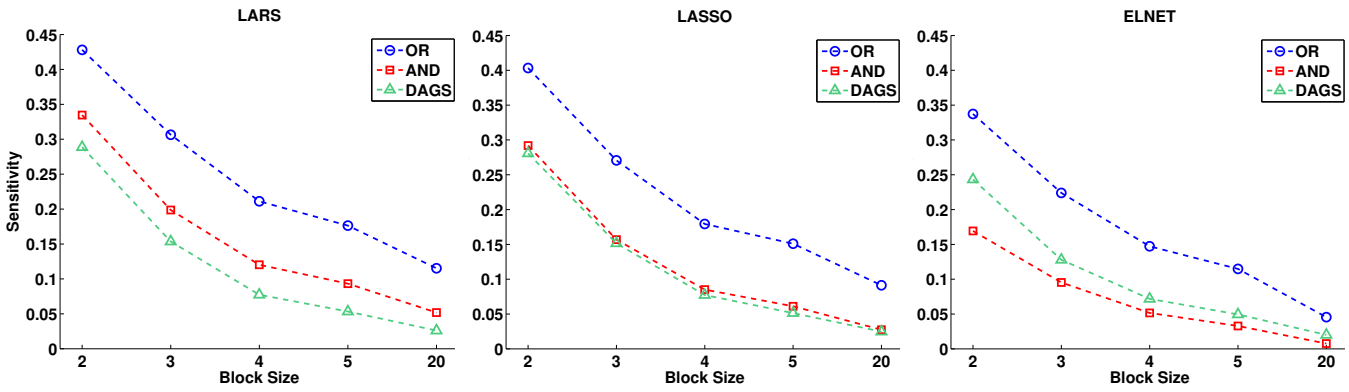



Figure 5.1: Average sensitivity of the model structures learnt with the methods in the first approach.

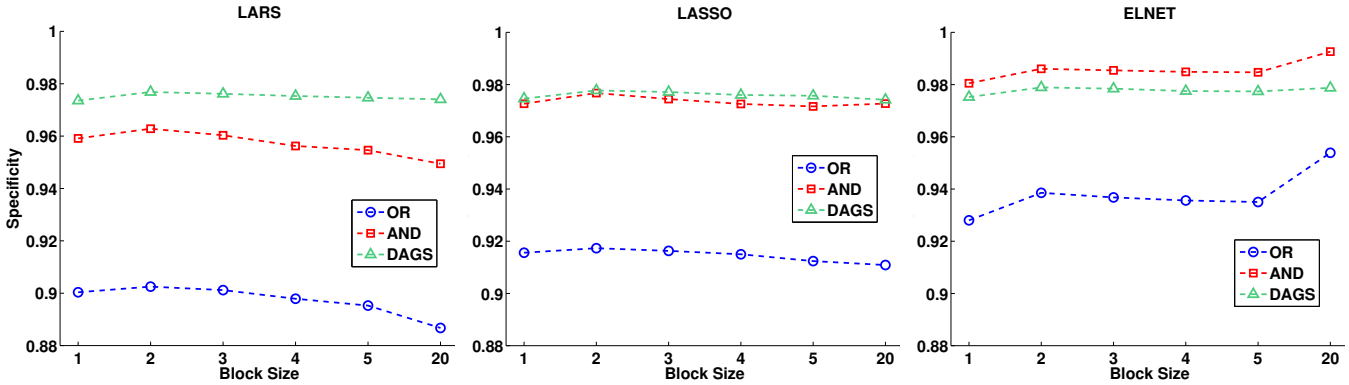
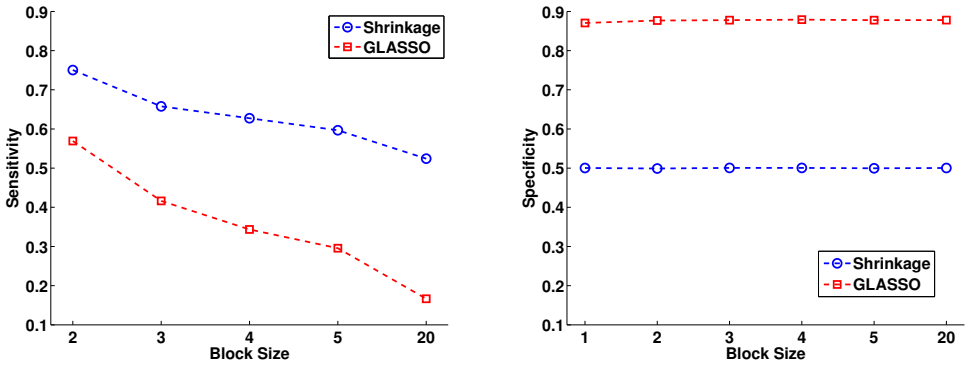



Figure 5.2: Average specificity of the model structures learnt with the methods in the first approach.

structure is a DAG, its undirected counterpart, obtained by removing the direction of the links, is considered for computing the indicators. This is because from a variable interdependency point of view, the important thing is the existence of a dependency not its direction.

The sensitivity results show that the proportion of true links captured by the methods in the first approach using different combinations of the regularization techniques and merging strategies drops as the size of the blocks of dependent variables increases. The OR strategy combined with all three regularization techniques has resulted in better sensitivity, as expected because it greedily adds links to the structure. On the other hand, the performance of the AND strategy changes with different regularization techniques from the sensitivity point of view. Whereas the LARS-AND combination is capturing more TP links than the LARS-DAGS combination, their sensitivity behavior comes very close to the LASSO technique, and the sensitivity of the AND strategy falls below the DAGS strategy when using the elastic net technique. The sensitivity results of all merging strategies decrease for elastic net, but this regularization technique appears to affect the AND strategy more than other strategies. One possible explanation for this behavior is the grouping effect of the elastic net technique, which causes several variables to be added to or removed from the regression model together. Therefore, when the AND strategy is




(a) Sensitivity (b) Specificity

Figure 5.3: Average accuracy of the model structures learnt in the second approach.

used to couple all regression models, a group of variables is less likely to have a mutual relationship (as required by the AND strategy). Thus the resulting structure becomes sparser and may miss many TP links.

From the specificity point of view, the results show that the tendency of regularization techniques to obtain sparser models causes the structures learnt by the methods in the first approach to include very few spurious links. Increased block sizes do not seem to considerably affect these methods. Note at this point that, as shown in Table 5.1, the number of model independencies is larger with smaller block sizes. Therefore, normally one expects to see smaller specificity values for smaller block sizes. As block size increases, regularization techniques like LARS and LASSO tend to add more dependencies between the variables. However, as we saw with the sensitivity results, many of these links are incorrect and, thus, the number of FP links increases. Another point to be considered here is the total number of dependencies in a model compared with the sample size used for learning. This can affect the sparsity assumption based on which the consistency of some of the regularized model estimation methods in the first approach is shown [Meinshausen and Bühlmann, 2006].

When comparing the specificity and sensitivity results of different combinations of regularization techniques and merging strategies, they seem to be complementary. For example, the DAGS strategy results in poorer sensitivity results compared with the other two merging strategies combined with the LARS technique, but specificity is better. Whereas the sensitivity behavior of the AND strategy combined with the elastic net technique is worse than others, the same combination obtains better specificity results compared with all other combinations for all block sizes. Therefore, if a specific method tends to add more links to the structure (like the LARS-OR combination), the probability of recovering TP links will clearly increase, but so will the possibility of adding FP links. On the other hand, conservative methods like the ELNET-AND combination will hit fewer TP links but also add fewer spurious links to the structure.

Figure 5.3 shows the sensitivity and specificity of the structures learnt by the methods in the second approach. Since these methods do not explicitly obtain a structure, the structure encoded in the covariance matrix of the estimated MGD is used to evaluate their ability to recover the true structure. This structure is obtained from the zero pattern in the inverse covariance matrix by introducing a link between every two variables whose

respective entry in the inverse covariance matrix is not zero. The results show that the two methods in the second approach also follow a similar trend to the methods in the first approach: a decrease in the sensitivity and an almost uniform specificity as the size of the dependent variables block increases. These two methods discover more links than the methods in the first approach. Shrinkage estimation, especially, tends to add many links to the structure, and the increase in the number of dependent variables has less effect on this method. This results in a sensitivity of more than 50% for all block sizes. However, the specificity results show that many of the links that this method adds to the model are spurious and this method generally tends to obtain denser structures than other methods in either of the approaches. The graphical LASSO method behaves more like the methods in the first approach, and especially the LARS-OR and LASSO-OR combinations, as this method also uses a similar regularization mechanism by adding a LASSO penalization term to the ML estimation of the MGD.

### 5.3.2 Time Complexity

The computational time needed by an algorithm is an important feature that can affect its range of application and how it is going to be applied. In the case of EDAs, it is particularly critical since they make intensive use of learning methods. In this section, we examine the time complexity of the methods in each of the two approaches. The methods in the first approach have include three-step structure learning plus a parameter estimation of the target MGD. As already mentioned, the computational complexity of computing the whole regularization path of a regularized regression model for a variable with the LARS technique (which is also used as the base algorithm for the other two regularization techniques considered here) is $O(n^3 + Nn^2)$. Note that in high-dimensional problems, which is the case in our study, it is far less than this. The cost of selecting the best regression solution from the regularization path is $O(lN)$, where $l$ is the number of different solutions found for the regression model, and it is of order $O(n)$. These two steps are repeated for each of the $n$ variables.

The AND and OR strategies used in the third step to merge the $n$ models learnt for the variables are simple and only require $O(n^2)$ computations. However, the DAGS strategy is usually very costly and has a computational complexity of $O(kNn^2 + kn^4)$, where $k$ is the number of iterations that it takes the local greedy algorithm to search the constrained space. Finally, the algorithm used to estimate the covariance matrix according to the structure obtained by AND and OR strategies has a cost of $O(Nn^2)$.

The shrinkage estimation and graphical LASSO methods both have a total computational complexity of $O(Nn^2)$. The covariance matrix computation dominates the time requirement of computing the mean vector of the MGD, which thus does not influence the total cost of model estimation. The sampling algorithm used to generate $M$ new solutions from the estimated MGD requires $O(n^3 + Mn^2)$ computations. This is also the case for the PLS algorithm when sampling a Bayesian network.

Figures 5.4 and 5.5 show the total time that it takes all the methods in the first approach to learn a probabilistic model and then sample this model to generate a population of 1000 solutions, on a machine with 2.66 GHz Intel Core-i5 processor and 6 GB of memory. All the results are averaged over 30 independent runs. As expected, the DAGS strategy takes longer than the other merging strategies, though the choice of the

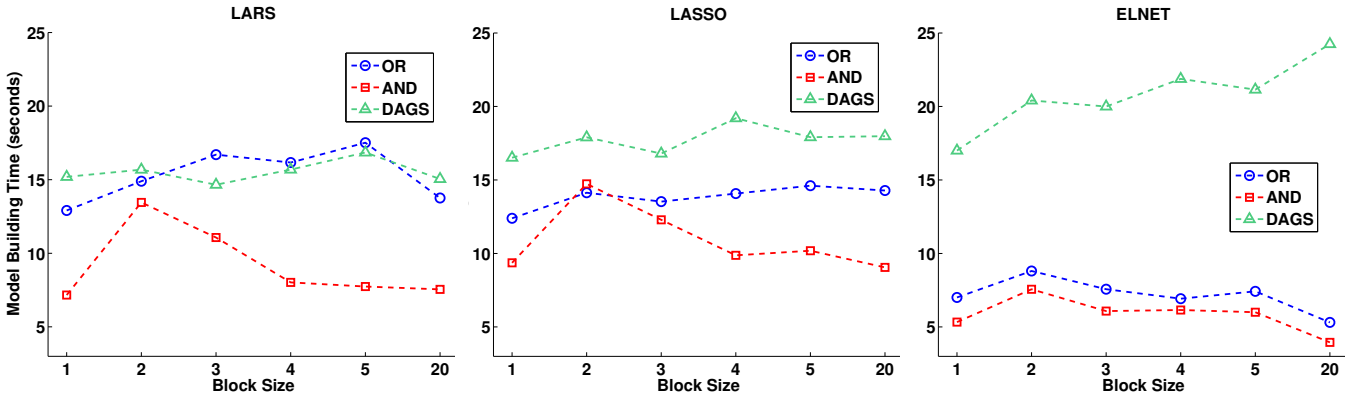



Figure 5.4: Average model building time for the methods in the first approach.

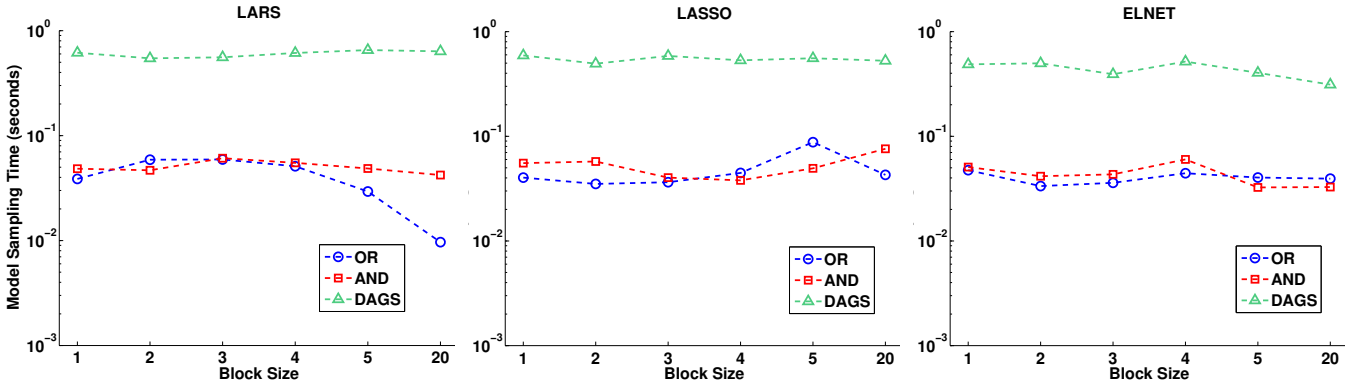



Figure 5.5: Average model sampling time for the methods in the first approach.

regularization technique has a big influence on this strategy, with the LARS technique requiring less and elastic net technique more time. The regularization technique also has an impact on the AND and OR strategies combined with the elastic net technique, which takes less time than the other two regularization techniques. This can be explained by the sparser structures obtained by this regularization technique which in turn would lead to the estimation of fewer parameters. However, sparser structures have the opposite effect on the time requirement of the DAGS strategy. This can be traced back to the fact that the greedy local search algorithm is forced to reject many of the possible moves when searching a constrained space and it therefore takes longer to find a valid move.

The question is then whether searching in a constrained space can cause the learning time of a Bayesian network to decrease at all. Figure 5.6 compares the model learning time required when searching an unconstrained space and a space constrained using the LARS technique. Both methods use the same greedy local search algorithm with a BIC scoring metric. Here, reference models with the increasing number of variables and equal size of dependent variables blocks, set to 5 (quint variate model), were sampled to generate populations of solutions, with a fixed $n/N$ ratio of 10. These populations were then used for learning Bayesian networks. The results are averaged over 30 independent runs. The LARS-AND method is also included in the results for better comparison. We found that while, the model learning time of the constrained DAGS is larger, for smaller number of variables the computational time required by the unconstrained DAGS grows faster as

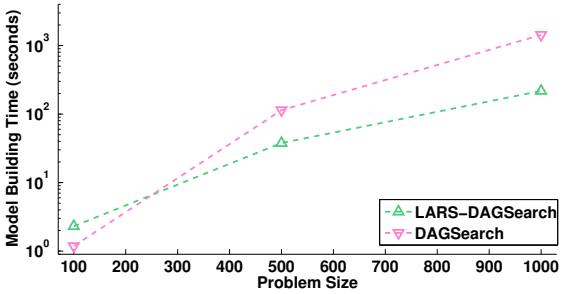
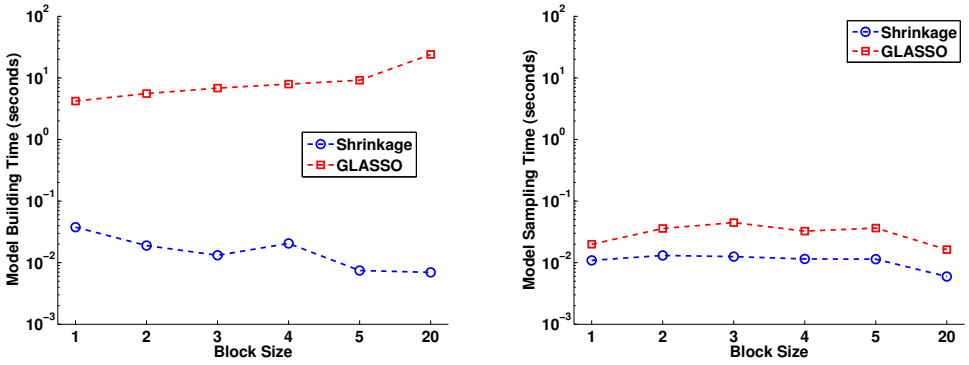



Figure 5.6: Average model-building time for constrained and unconstrained DAG search algorithms on different problem sizes.




(a) Model Building Time (b) Model Sampling Time

Figure 5.7: Average model-building (left) and sampling (right) time for the methods in the second approach.

the number of variables increase and becomes considerably larger than its constrained counterpart. This is because for larger number of variables, the search space becomes so huge that any kind of space-constraining information can serve as a heuristic to improve the search. For the difference in the estimation accuracy of these two methods, see [Schmidt et al., 2007; Vidaurre et al., 2010].

As the model sampling times show (Figure 5.5), it takes longer to generate new solutions from a Bayesian network using the PLS algorithm than the algorithm used to sample an MGD. Although the same algorithm is used to sample the models learnt by the AND and OR strategies, the results show that the estimated models can affect the sampling time depending on the regularization technique and the block size. Generally, however both the model learning and sampling times do not appear to change considerably by increasing the size of dependent variables block, since all these models have the same dimension. Figure 5.6 shows, on the other hand, that the increase in the number of variables raises the algorithm time requirement, as expected from the computational complexity analysis.

Figure 5.7 shows the average model-learning and sampling times for the two methods in the second approach, on the same machine. Sampling times are computed for generating a population of 1000 solutions from the learnt model. The graphical LASSO method takes significantly longer to estimate the model than shrinkage estimation, and

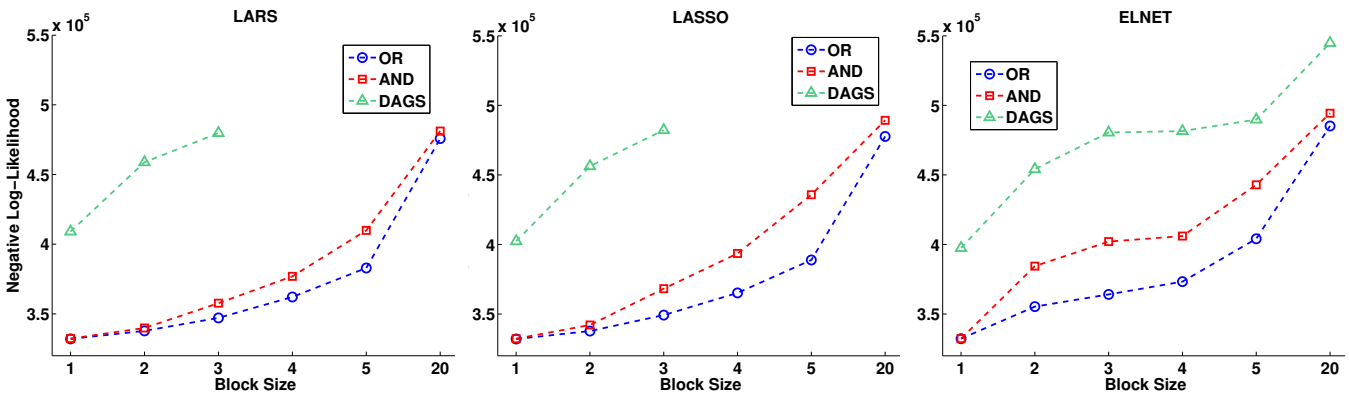



Figure 5.8: Average negative log-likelihood of the reference Gaussian distributions obtained for model estimation methods in the first approach.

its time requirements closely follow those of the methods in the first approach, especially the LARS-OR and LASSO-OR combinations (whose structural accuracy is similar, too). Again, the size of blocks of dependent variables does not have a substantial effect on the computational time of these two methods, although further experiments have shown that graphical LASSO is highly sensitive to the violation of the conditions in the sparsity assumption. Section 5.3.4 discusses this issue in more detail. The time requirements of the shrinkage estimation method make it a perfect candidate for use within more complex algorithms like EDAs that require very fast-performing components.

### 5.3.3 Likelihood Function

Estimating an accurate structure is an important feature of a probabilistic model estimation method. For the purpose of model estimation and sampling in EDAs (Algorithm 2.1), however, it is more important to be able to generate solutions that are very close to the real problem solution, which is assumed to be representable with a probability distribution. One way to investigate this closeness is to compare the model estimated in EDA with the actual probabilistic model underlying the problem. Measures like Kullback-Leibler divergence [Kullback and Leibler, 1951] or Hellinger distance [Le Cam and Lo Yang, 2000] can be used for this purpose. Another possibility is to evaluate the overall model estimation and sampling of EDAs in order to also take into account the inevitable model sampling error that is present in practice. This is also closer to the actual procedure enacted in each generation of an EDA. The evaluation measure computed in this study is the negative log-likelihood (NLL) of the reference Gaussian models (see Table 5.1) given the population of solutions generated from the estimated models.

Figure 5.8 shows the NLL values of the reference Gaussian models with different block sizes, computed from the populations generated using the models learnt with the methods in the first approach. Each of the generated populations had a size of 1000. For the first two regularization techniques, the NLL values obtained with the DAGS strategy on larger block sizes tend to infinity and are therefore not included in the figures. Merging strategies are ordered similarly for all regularization techniques, with the NLL of OR strategy turned out to be better and DAGS worse. This is consistent with the structural accuracy results and reveals the mixed effect of sensitivity and specificity analysis for the

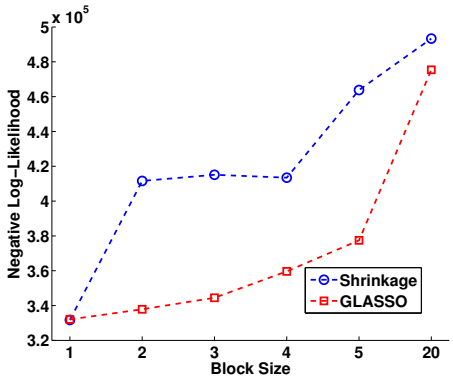



Figure 5.9: Average negative log-likelihood of the reference Gaussian distributions obtained for the estimation methods in the second approach.

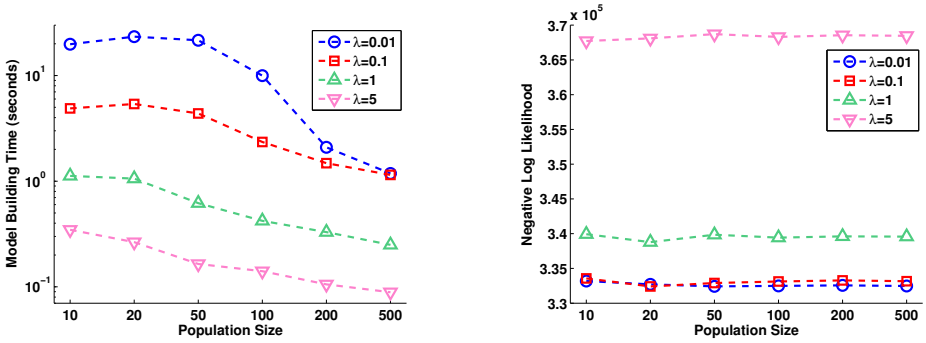
methods in the first approach. The NLLs computed for these methods grow visibly as block size increases, suggesting that as the number of dependent variables in the problem increases the estimated models and populations generated from these models move farther from the reference models, from a log-likelihood point of view.

Figure 5.9 shows the NLL results for the two methods in the second approach, where graphical LASSO results in better NLL values than the shrinkage method. Shrinkage estimation is able to capture more dependencies, but, at the same time, it adds many spurious links to the structure. As a result, the estimated probability distribution generates solutions that are less likely producible by the reference models. From the NLL values for graphical LASSO, and for LARS-OR and LASSO-OR used in the first approach, we also find that these methods behave similarly. Also, the NLL values computed for graphical LASSO and for the ML estimation of the MGD are almost equal, showing that, from a log-likelihood point of view, the addition of a LASSO penalization term does not have much impact on the probability estimation of the graphical LASSO method.

### 5.3.4 Regularization Parameter in Graphical LASSO Method

The regularization techniques employed in the first approach output the whole regularization path for varying values of the regularization parameter ($\lambda$) and then select the parameter resulting in the best regression solution according to a model selection metric. Also, the value of the shrinkage intensity in the shrinkage estimation method is analytically computed according to the learning data set. However, the regularization parameter of the graphical LASSO method is left open, as it is not practicable to compute the whole regularization path for this method, especially for high-dimensional problems.

To examine the influence of the regularization parameter, we apply the graphical LASSO method with different values of this parameter for model estimation. The model learning times and the corresponding NLL values computed for this method are shown in Figure 5.10. Models are estimated from populations with increasing sizes, generated from the bivariate reference model by decreasing the $n/N$ ratio from 10 to 0.2 (gradually getting away from high-dimensionality) in order to also investigate the effect of population size on this method. NLL values are computed using populations of size 1000 generated from the estimated models. All of the results are averaged over 30 independent runs.




(a) Model Building Time (b) Negative Log-Likelihood

Figure 5.10: Average model-building time (left) and negative log-likelihood of the bivariate Gaussian distribution (right) obtained for the graphical LASSO method with different regularization parameter ($\lambda$) values.

Clearly, whereas model estimation is faster with larger values of the regularization parameter, the solutions generated from these models are less likely from the reference model point of view. On the other hand, small values of this parameter (close to zero) will result in better NLL values, though at a higher computational cost. This observation suggests that, as reported in a previous study [Karshenas et al., 2011a], the choice of a value for the regularization parameter of this method call for a trade-off between how good an estimation is and the time required to estimate the model.

The model learning times also show that, as population size grows the time required by the graphical LASSO method with a specific value of the regularization parameter gradually decreases. This is why it is expected to be more costly to estimate a model, with a specific dimension and block size, from populations with larger sizes. One possible explanation for this behavior is that with larger populations that better fulfill the sparsity assumption [Meinshausen and Bühlmann, 2006], more harmonious statistics can be collected from the population, allowing the method to converge faster. However, as the computed NLL values show, the population size increase does not affect the probability of the generated solutions from the reference model point of view. Based on these results, a 0.1 value is used for the regularization parameter of the graphical LASSO method throughout the thesis.

## 5.4 Conclusions

The use of regularized model estimation in EDAs for continuous optimization was proposed and studied in this chapter. It was argued that the use of regularization techniques can lead to a more robust model estimation in EDAs. This improves performance in high-dimensional settings, while keeping the population size and therefore the number of function evaluations relatively low.

Considering a high-dimensional setting, different methods for regularized estimation of Gaussian distributions in two general approaches were analyzed from several points of view: true structure recovery, time complexity, and likelihood. The analysis results showed the properties of these methods, like their tendency in including dependencies

between variables and the learning/sampling times as the number of variables and their interactions increase. It was observed that an important factor affecting regularized model estimation is the level of variable dependencies in the problem.

# Chapter 6

# Regularization-Based EDA in Continuous Optimization

## 6.1 Introduction

Given a fitness or objective function $f : \mathbb{D} \mapsto \mathbb{R}$, the goal of (single-objective) optimization is to find solution(s) $\boldsymbol{x}^*$ such that (assuming smaller values of $f$ are preferred)

$$\boldsymbol{x}^* = \arg \min_{\boldsymbol{x} \in \mathbb{D}} f(\boldsymbol{x}) \tag{6.1}$$

If the optimization algorithm does not use any problem-specific information for finding the optimal solution(s), it is called a black-box or general-purpose optimization algorithm. In continuous domains, we have $\mathbb{D} \subseteq \mathbb{R}^n$, where $n$ is the dimension of the input domain. Since this set is uncountable, in general it is not possible to find an upper bound on the number of steps required to reach the optimal solution(s) to a continuous optimization function. Therefore, in this case the algorithms that can get closer to the optimal solutions faster (e.g. in smaller number of steps) are favorable.

In the last chapter, we saw how regularization techniques can be used for model estimation in EDAs. The analysis especially considered high-dimensional settings to study the performance of regularized model learning from small populations with respect to several criteria. In this chapter we analyze this type of model learning from the continuous function optimization point of view and compare its performance with several other well-known continuous EDAs. This work is published in [Karshenas et al., 2013b].

## 6.2 RegEDA: Regularization-based EDA

The results of the regularized model estimation analysis can be used as a guideline for employing these methods in EDAs. The regularization-based EDA (RegEDA) proposed in this chapter utilizes the regularized model estimation methods in the course of optimization, trying to obtain a better estimation of the distribution of the problem solutions, in order to improve performance.

The constraints on time and computational resources restrict the number of different regularized estimation methods that can be tested and compared. Therefore, for the experiments in the rest of the chapter, some of the discussed methods were selected

for use in RegEDA. We selected the LARS-OR and LARS-AND methods from the first approach. They appear to offer a better compromise between the computational time requirements and the quality of the estimated models, from both the structural accuracy and NLL points of view. The resulting algorithms are called "RegEDA-LARS-OR" and "RegEDA-LARS-AND", respectively. From the second approach, both the shrinkage estimation and graphical LASSO methods were selected and respectively used in "RegEDA-Shr" and "RegEDA-GL" algorithms. The properties of both methods motivate further investigation regarding optimization. In the rest of this chapter, we study the performance of these four versions of RegEDA in function optimization.

## 6.3 Experiments

The proposed RegEDAs are applied for continuous function optimization in order to investigate how regularized model estimation affects the optimization behavior and performance of EDAs when applied in a high-dimensional setting. The optimization results of the four versions of RegEDA are compared against another four Gaussian distribution-based EDAs, which are: i) Continuous UMDA [Larrañaga et al., 2000a], ii) EGNA [Larrañaga et al., 2000a], iii) EMNA [Larrañaga and Lozano, 2001], and iv) CMA-ES [Hansen, 2006]. All of the algorithms are implemented in Matlab®. Specifically, we have used Mateda-2.0 package[1] [Santana et al., 2010a], which provides the implementation of UMDA and EMNA, as the basis of our implementation framework. For the implementation of LARS technique in RegEDA-LARS-OR and RegEDA-LARS-AND algorithms, we have used the code provided by K. Sjöstrand[2], and the covariance shrinkage method is implemented with the code provided by K. Murphy[3]. The implementation of the graphical LASSO is adapted for Matlab[4] from the original code provided for the algorithm [Friedman et al., 2008].

For the implementation of EGNA and especially its GBN learning, we have used the code provided by M. Schmidt and K. Murphy[5] [Schmidt et al., 2007]. It slightly differs from the method described in Section 1.3.3 by allowing the greedy method to restart the structure search from a new random structure after reaching a local optimum of the score function, up to a maximum number of node score evaluations. At the end, the highest scoring BN in all these sub-searches is returned. Since the PLS algorithm is used to sample the estimated GBN, the algorithm is referred to as "EGNA-PLS" in the presented results. Finally, we have used the Matlab implementation of CMA-ES provided by N. Hansen[6].

### 6.3.1 Optimization Functions

The continuous optimization functions used for the experiments are listed in Table 6.1. These optimization functions, defined on $n$ input variables, have different properties that

[1] http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/MATEDA.html
[2] http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3897/zip/imm3897.zip
[3] http://www.uni-leipzig.de/~strimmer/lab/software/m-files/covshrink-kpm.zip
[4] http://cig.fi.upm.es/components/com_phocadownload/container/GraphicalLasso.zip
[5] http://www.cs.ubc.ca/~murphyk/Software/DAGlearn/DAGLearn.zip
[6] http://www.lri.fr/~hansen/cmaes_inmatlab.html

allow to examine the performance of the tested optimization algorithms, in the presence of different problem features. The 2D fitness landscape of some of these functions, is shown in Figure 6.1. The definitions of the functions are as follows.

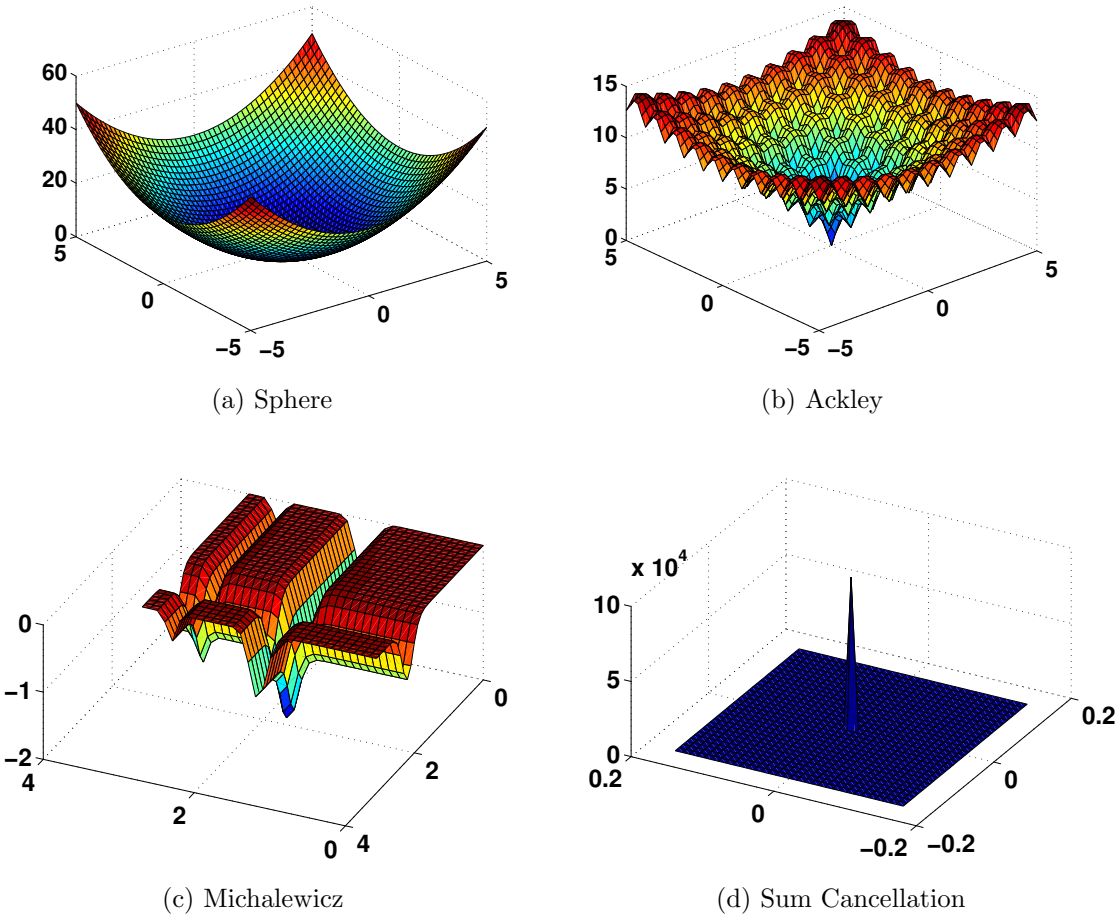
- Sphere
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2.$$
- Ackley
$$f(\boldsymbol{x}) = -a \exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + e,$$
where $a$, $b$ and $c$ are the parameters of the function and are set to 20, 0.2 and $2\pi$, respectively. $e$ is Euler's number.
- Tablet
$$f(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2.$$
- Cigar-Tablet
$$f(\boldsymbol{x}) = x_1^2 + 10^4 \sum_{i=2}^{n-1} x_i^2 + 10^8 x_n^2.$$
- Michalewicz
$$f(\boldsymbol{x}) = -\sum_{i=1}^{n} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right),$$
where $m$ is the parameter of the function and is set to 10. The optimum value of the function is different for different numbers of variables.
- Sum Cancellation
$$f(\boldsymbol{x}) = \left(10^{-5} + \sum_{i=1}^{n}\left|\sum_{j=1}^{i} x_j\right|\right)^{-1}.$$

The Sphere function is a simple optimization problem without any interdependency between variables. Following the smooth downhill path will lead an optimization algorithm to the optimal solution of the function. The Ackley function has a rugged landscape, although some local regions of the search space can provide information about the global structure of the problem. The first variable of the Tablet function is scaled causing the optimization algorithms to be more sensitive to changes of this variable, and therefore the promising values of other variables may not be properly encoded in EDA model estimation. The Cigar-Tablet function extends the Tablet function by introducing three different levels of scalings for the variables. The Michalewicz function does not have a proper global structure and requires more exploration for detecting promising basins of attraction. The Sum Cancellation function is very similar to a problem called needle in the haystack, especially for larger dimensions, and the function is not separable. Therefore, a very small search space is considered for this function.

Table 6.1: The optimization functions used in the experiments, their optimum solution ($\boldsymbol{x}^*$) and optimum function value ($f^*$). The number of variables is denoted with $n$.

| | Name | Type | Domain | $\boldsymbol{x}^*$ | $f^*$ |
|---|---|---|---|---|---|
| 1 | Sphere | min | $[-5, 5]^n$ | $\mathbf{0}$ | 0 |
| 2 | Ackley | min | $[-32, 32]^n$ | $\mathbf{0}$ | 0 |
| 3 | Tablet | min | $[-7.5, 7.5]^n$ | $\mathbf{0}$ | 0 |
| 4 | Cigar-Tablet | min | $[-7.5, 7.5]^n$ | $\mathbf{0}$ | 0 |
| 5 | Michalewicz | min | $[0, \pi]^n$ | –[a] | –[a] |
| 6 | Sum Cancellation | max | $[-0.16, 0.16]^n$ | $\mathbf{0}$ | $10^5$ |

[a] Depends on $n$.




(a) Sphere

(b) Ackley

(c) Michalewicz

(d) Sum Cancellation

Figure 6.1: Fitness landscape of some of the optimization functions in two dimensions.

### 6.3.2 Experimental Design

Five different dimensions are tested for each of the functions: 10, 20, 50, 100 and 200 variables. Population size is set to $N = 10\ln(n)$ for all of the algorithms in an attempt to emulate high-dimensional settings as the number of variables increase, starting with a certain number of solutions. For each algorithm-function-dimension combination, 20 independent runs are performed. The initial population is randomly generated us-

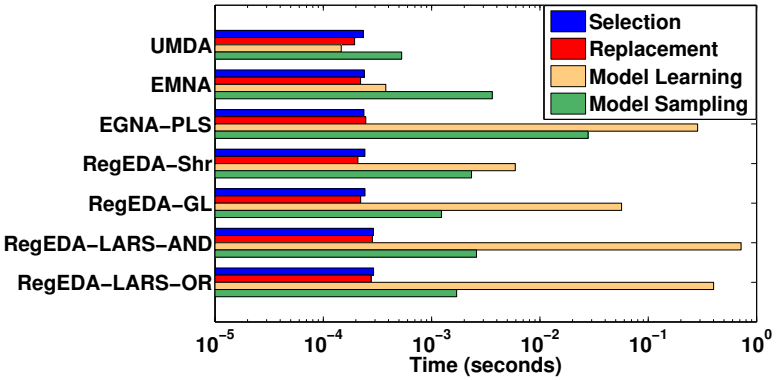



Figure 6.2: Average time requirements for the main steps of RegEDAs and their comparison with other EDAs. The number of variables is 50.

ing a uniform distribution over the domain of variables. All algorithms terminate when the maximum number of generations, set to 500, is reached. Apart from this stopping criterion, when an algorithm gets stuck in a stalemate situation for 100 consecutive generations, it stops so as not to waste computational time. A stalemate situation is verified if the improvement in the fitness function is less than $10^{-8}$. Figure 6.2 gives an insight into the computational time requirements of the main steps of RegEDAs, which are compared with three other Gaussian-based EDAs used in the experiments of this section. The presented results are obtained on a machine with 2.66 GHz Intel Core-i5 processor and 6 GB of memory, and are averaged over 500 generations of a run.

For all EDAs, solutions are selected using truncation selection with a $\tau = 0.5$ threshold. Except for CMA-ES, which completely replaces the population in each generation with new solutions, all other algorithms generate $N/2$ new offspring solutions in the sampling step. The newly generated solutions are repaired using a simple repairment strategy, where unacceptable values are replaced by a new value randomly chosen from the domain of the corresponding variable. An elitist replacement strategy is used to incorporate the offspring solutions into the population, where the best $N$ solutions from the combination of offspring solutions and solutions in the population are selected to form the next-generation population. The initial standard deviation of CMA-ES (one of algorithm's input parameters) is set to one third of each variable's domain, as it is suggested in [Hansen, 2006]. This algorithm is considered to be in a stalemate situation if the improvement in the fitness function is less than $10^{-12}$ (which is less strict than other algorithms).

### 6.3.3 Results

Figures 6.3–6.8 show the average best achieved values (BAVs) along the evolution path of RegEDAs and the other four EDAs, applied to the optimization functions. The presented results are averaged over the 20 runs performed. For runs that an algorithm terminates before reaching the maximum number of generations, the rest of evolution path is padded with the BAV of the last executed generation. For some of the functions, the BAVs are depicted on a logarithmic scale so as to better discriminate algorithms' performance.

The results for all functions show that, when considering the distance between BAVs

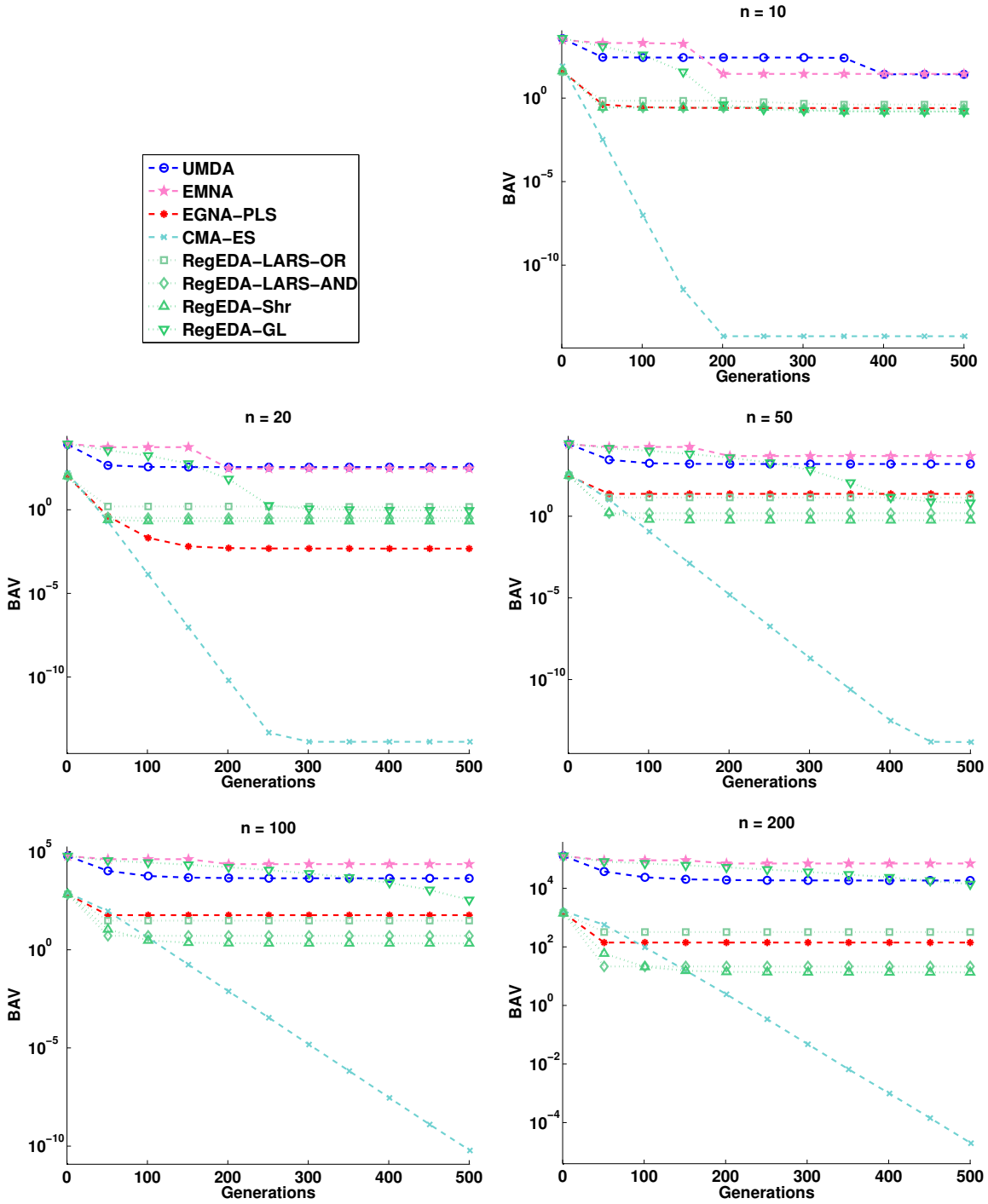



Figure 6.3: Average BAVs for Sphere function

and the optimal function value, the performance of all algorithms drops as the number of variables increases. However, this curse of dimensionality affects some algorithms (like CMA-ES and EGNA-PLS) a lot more than others. The optimization behavior of RegEDAs for most of the tested functions, suggests that these algorithms are less affected by this phenomenon.

The comparison of ML estimation in EMNA and UMDA with the regularized model

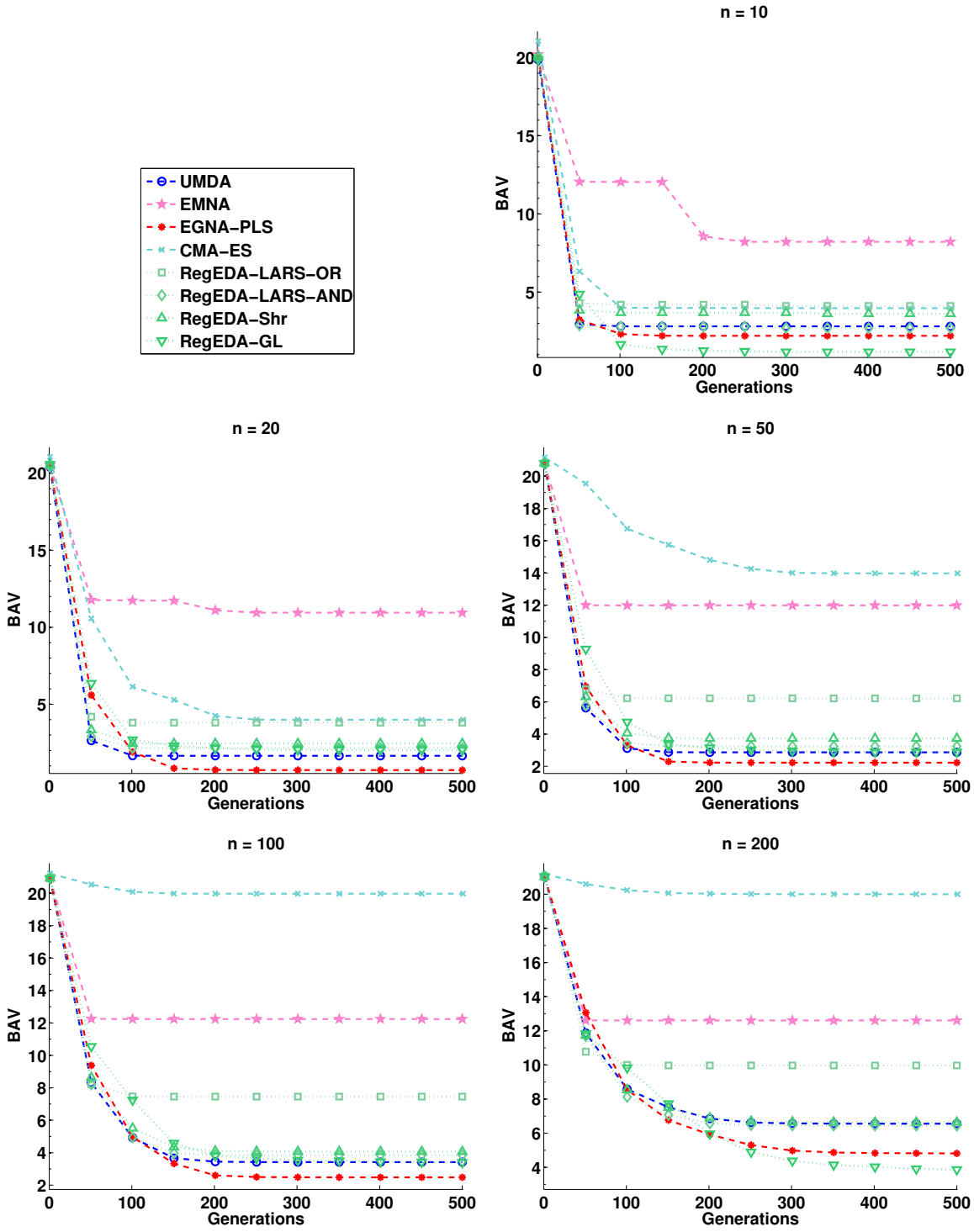



Figure 6.4: Average BAVs for Ackley function

estimation in RegEDAs better illustrates the difference in the performance of RegEDAs and how they are affected by the increases in the problem size. Since all other parts of the tested algorithms are the same (except for CMA-ES), the similarities and differences in the optimization performance of these algorithms can be attributed to the model estimation methods that they employ. For example, the performance of RegEDA-Shr and UMDA is

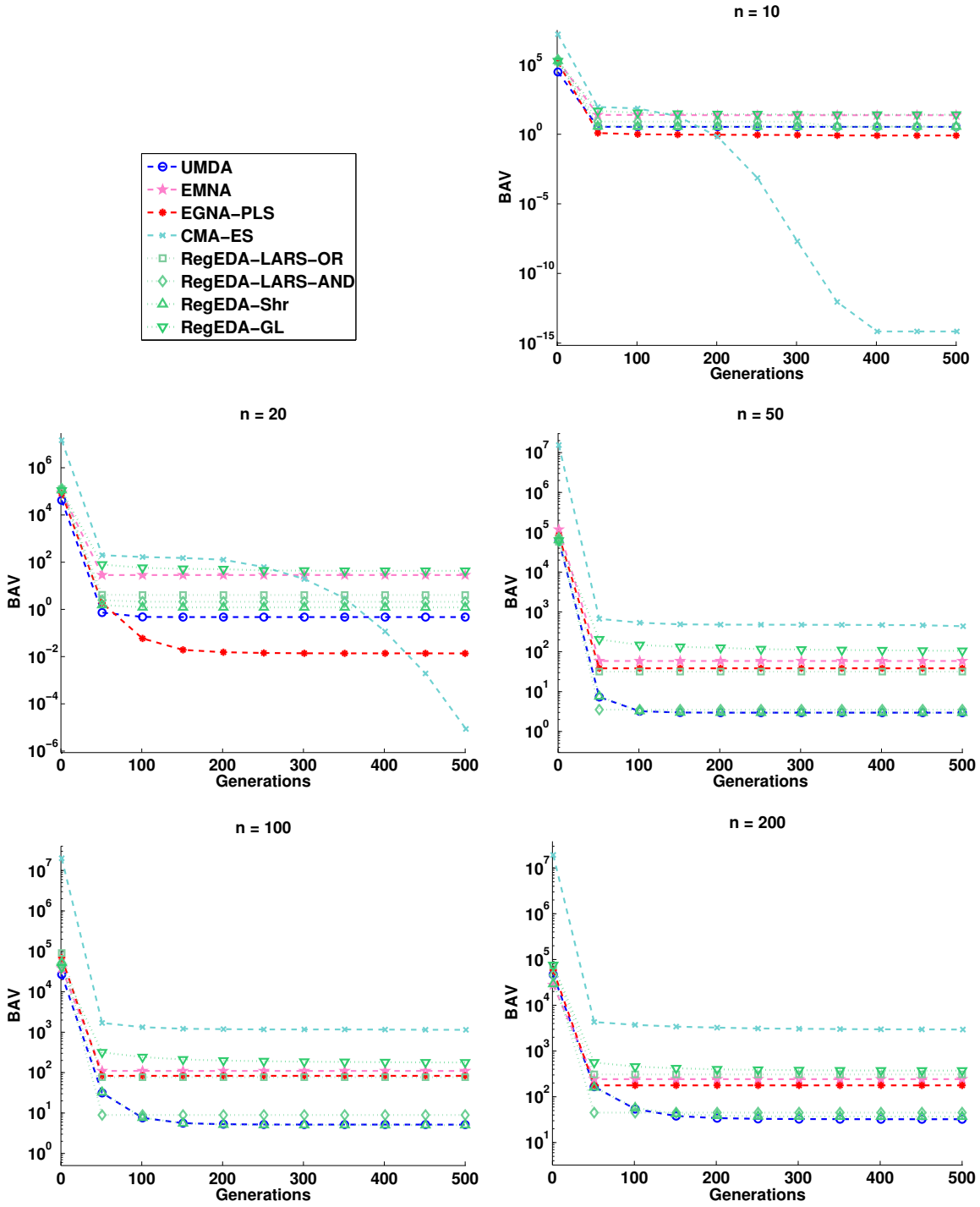



Figure 6.5: Average BAVs for Tablet function

very close for most of the functions, suggesting that the shrinkage estimation method is shrinking most of the off-diagonal entries in the covariance matrix to close-to-zero values. This property is especially useful when dealing with separable optimization problems. A comparison of the performances of RegEDA-Shr, and UMDA and EMNA for Sphere functions (Figure 6.3), clearly shows that the model estimation employed in RegEDA-Shr is more efficient, since it leads to a regularized combination of the models used in EMNA

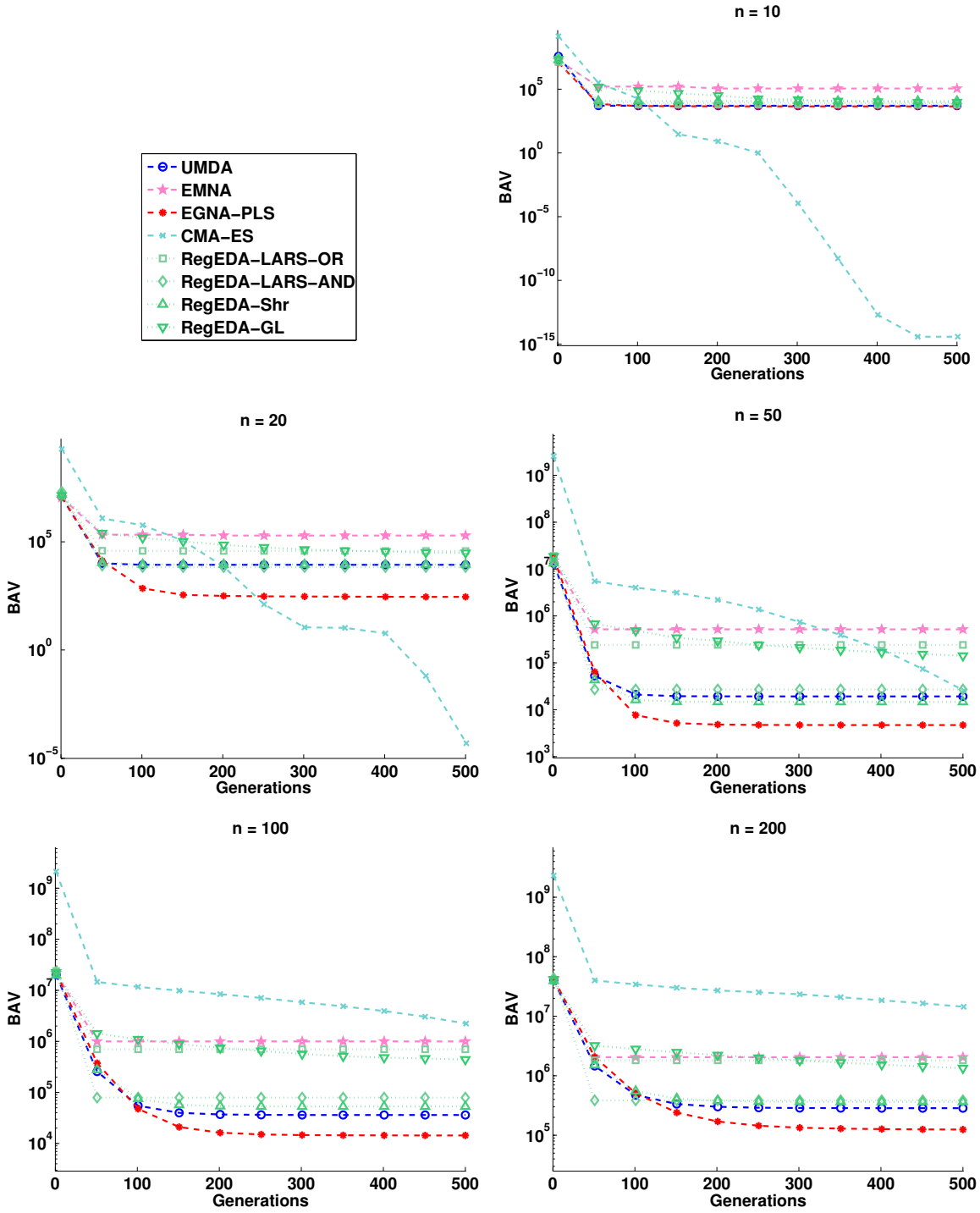



Figure 6.6: Average BAVs for Cigar-Tablet function

and UMDA.

The conservative merging strategy used in RegEDA-LARS-AND model estimation causes fewer dependencies to be added to the model, leading to sparser structures. The fact that, for many of the functions, this algorithm behaves similarly to UMDA, and therefore RegEDA-Shr, shows that the sparsity pattern of these structures is very similar to diagonal matrices in the presence of problem separability. When the problem is not

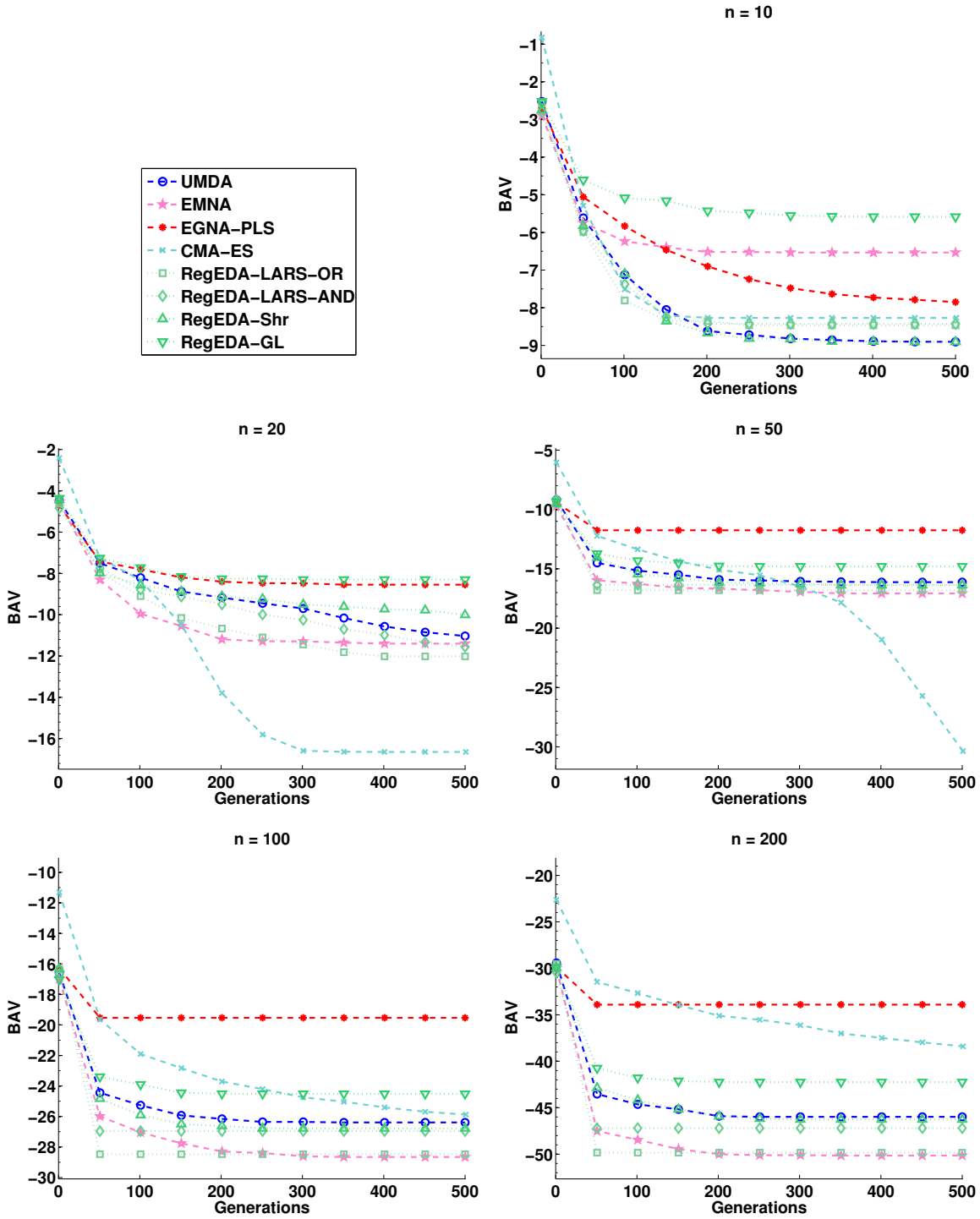



Figure 6.7: Average BAVs for Michalewicz function

separable (like the Sum Cancellation function – Figure 6.8), RegEDA-LARS-AND can, thanks to this regularized model estimation method, perform a more concentrated search by including only what are, according to the regularization technique, the more important links. Therefore this algorithm is able to outperform other algorithms with the increase in problem dimensionality.

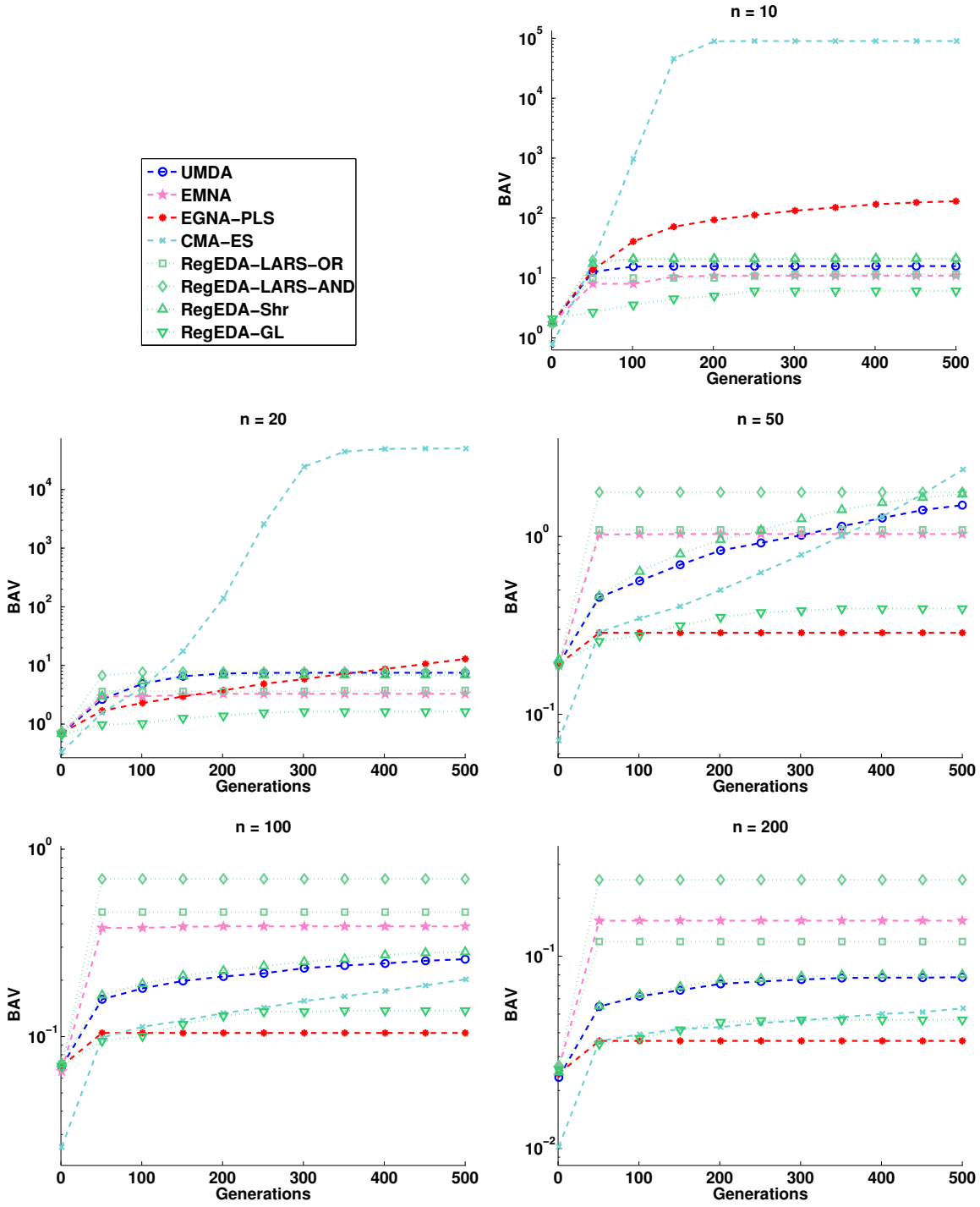



Figure 6.8: Average BAVs for Sum Cancellation function

The optimization performance of RegEDA-GL compared with EMNA is especially interesting in these experiments. The model estimation method in the two algorithms differs only as to the regularization term added to ML estimation of MGD in the graphical LASSO method (see Section 4.3). The results show that, for some of the functions (like Sphere and Ackley – Figures 6.3 and 6.4), regularization improves the average performance of RegEDA-GL while, for some others (Michalewicz and Sum Cancellation –

Figures 6.7 and 6.8), this algorithm is not able to obtain good BAVs as EMNA. Considering the properties of these functions, it appears that the use of regularization will result in better model estimation if the problem has a global structure, whereas lack of such information has a negative effect on the regularized model estimation used in RegEDA-GL.

The similar optimization behavior of RegEDA-LARS-OR and EMNA for many of the functions suggests that the MGDs estimated with the greedy merging strategy employed in RegEDA-LARS-OR are very similar to those obtained by ML estimation. For example, a good option, in the absence of useful properties in some of the functions (like Michalewicz – Figure 6.7), for use by the optimization algorithm, could be a coarse estimation of the search space. Model estimation in RegEDA-LARS-OR simulates this coarse estimation by adding many links to the structure considering all possible variable dependencies. However, the use of regularization to detect the dependencies, in the presence of specific problem properties (like separability) causes the algorithm to perform a finer search.

### 6.3.4 Discussion

There are some common points concerning the performance of the algorithms on the tested functions. First, most of the algorithms show a different optimization behavior from one function to another depending on the different features of these functions. This suggests that although no explicit rotation and translation is applied, the set of selected functions form a good benchmark for the experiments. On the other hand, the Gaussian distribution-based model estimation used in the algorithms is invariant to these transformations (except UMDA). Second, the performance of all the algorithms is affected by the increase in the number of variables, from the viewpoint of the distance between BAVs and optimal function value. Some of the algorithms, like CMA-ES, appear to be able to improve their BAVs if given more time, but it is evident from the results that their performance is influenced considerably more than the RegEDAs proposed here, when the number of variables increase.

Thirdly, the population size of all algorithms is equally set to be of a logarithmic order of the problem size, resembling a high-dimensional setting. Thus, for most of the functions and especially larger dimensions, almost all the algorithms get stuck in a stalemate situation in the early generations of the search. For the purpose of studying the effect of regularized model estimation, none of the algorithms (except CMA-ES, which uses specific variance scaling techniques) use any kind of explicit diversity preservation techniques. The fact that all other parts of the algorithms are the same is helpful for gaining a better understanding of how different model estimation methods affect the optimization behavior.

Table 6.2 shows the statistical analysis results for the BAVs obtained by the algorithms on each dimension of each function. The non-parametric Friedman rank test [Derrac et al., 2011] is used to check for the statistical difference in algorithm's performance. The null hypothesis that all the algorithms have an equal average rank is rejected with a p-value less than $10^{-8}$ for all functions and all dimensions. The values shown in each entry of Table 6.2 are the final average BAVs obtained by the algorithm in the column, applied on the function-dimension combination in the row. The numbers in parentheses show

Table 6.2: The results of statistical tests on BAVs of the algorithms (refer to text for more explanation).

| Function | Dim. | UMDA | EMNA | EGNA-PLS | CMA-ES | RegEDA-LARS-OR | RegEDA-LARS-AND | RegEDA-Shr | RegEDA-GL |
|---|---|---|---|---|---|---|---|---|---|
| Sphere | 10 | 2.67e+01 (0, 6) | 2.81e+01 (0, 6) | 2.50e-01 (2, 1) | 5.53e-15 (7, 0) | 4.10e-01 (2, 1) | 2.31e-01 (2, 1) | 1.61e-01 (2, 1) | 1.55e-01 (2, 1) |
| | 20 | 3.58e+02 (0, 5) | 2.90e+02 (0, 5) | 4.76e-03 (4, 0) | 1.33e-14 (6, 0) | 1.50e+00 (0, 4) | 3.22e-01 (3, 1) | 2.10e-01 (3, 1) | 9.15e-01 (2, 2) |
| | 50 | 1.58e+03 (0, 4) | 4.88e+03 (0, 5) | 2.35e+01 (0, 3) | 1.50e-14 (5, 0) | 1.40e+01 (1, 2) | 1.53e+00 (3, 0) | 5.66e-01 (4, 0) | 6.51e+00 (2, 1) |
| | 100 | 4.35e+03 (0, 4) | 2.30e+04 (0, 5) | 5.78e+01 (1, 3) | 6.02e-11 (5, 0) | 3.05e+01 (2, 1) | 5.13e+00 (4, 0) | 2.12e+00 (4, 0) | 3.44e+02 (0, 3) |
| | 200 | 1.84e+04 (0, 5) | 6.99e+04 (0, 5) | 1.41e+02 (2, 2) | 1.98e-05 (5, 0) | 3.21e+02 (2, 2) | 2.15e+01 (3, 0) | 1.35e+01 (5, 0) | 1.38e+04 (0, 3) |
| Ackley | 10 | 2.81e+00 (1, 0) | 8.22e+00 (0, 6) | 2.20e+00 (2, 0) | 3.97e+00 (3, 0) | 4.12e+00 (0, 3) | 2.70e+00 (1, 0) | 3.62e+00 (1, 2) | 1.17e+00 (3, 0) |
| | 20 | 1.65e+00 (2, 0) | 1.09e+01 (0, 6) | 7.27e-01 (2, 0) | 3.99e+00 (2, 0) | 3.81e+00 (0, 3) | 2.15e+00 (1, 0) | 2.45e+00 (1, 0) | 2.01e+00 (1, 0) |
| | 50 | 2.87e+00 (3, 0) | 1.20e+01 (0, 5) | 2.22e+00 (4, 0) | 1.40e+01 (0, 4) | 6.22e+00 (0, 4) | 3.24e+00 (3, 0) | 3.73e+00 (1, 1) | 2.89e+00 (3, 0) |
| | 100 | 3.42e+00 (3, 0) | 1.22e+01 (0, 5) | 2.48e+00 (5, 0) | 2.00e+01 (0, 5) | 7.46e+00 (0, 4) | 3.84e+00 (3, 1) | 4.07e+00 (2, 1) | 3.39e+00 (3, 0) |
| | 200 | 6.56e+00 (2, 2) | 1.26e+01 (0, 5) | 4.82e+00 (5, 0) | 2.00e+01 (0, 5) | 9.97e+00 (0, 3) | 6.45e+00 (3, 1) | 6.64e+00 (2, 2) | 3.87e+00 (6, 0) |
| Tablet | 10 | 3.40e+00 (2, 1) | 2.32e+01 (0, 6) | 8.07e-01 (2, 0) | 6.84e-15 (6, 0) | 3.55e+00 (2, 1) | 3.40e+00 (2, 1) | 3.62e+00 (2, 1) | 2.55e+01 (0, 6) |
| | 20 | 4.73e-01 (2, 1) | 2.84e+01 (0, 5) | 1.36e-02 (5, 0) | 8.63e-06 (6, 0) | 4.07e+00 (1, 2) | 2.10e+00 (2, 2) | 1.21e+00 (2, 2) | 4.30e+01 (0, 6) |
| | 50 | 2.96e+00 (5, 0) | 5.88e+01 (0, 3) | 3.86e+01 (2, 3) | 4.39e+02 (0, 5) | 3.18e+01 (2, 3) | 3.53e+00 (5, 0) | 2.97e+00 (5, 0) | 1.05e+02 (0, 5) |
| | 100 | 5.17e+00 (5, 0) | 1.10e+02 (1, 3) | 8.32e+01 (2, 3) | 1.15e+03 (0, 6) | 7.69e+01 (2, 2) | 8.97e+00 (4, 0) | 5.04e+00 (5, 0 | 1.78e+02 (0, 5) |
| | 200 | 3.23e+01 (5, 0) | 2.43e+02 (1, 3) | 1.78e+02 (2, 2) | 2.95e+03 (0, 6) | 3.08e+02 (1, 3) | 4.51e+01 (4, 0) | 3.69e+01 (5, 0) | 3.74e+02 (0, 4) |
| CigarTablet | 10 | 5.01e+03 (1, 1) | 1.13e+05 (0, 6) | 4.53e+03 (1, 1) | 3.77e-15 (7, 0) | 6.04e+03 (1, 1) | 7.80e+03 (1, 1) | 1.20e+04 (1, 1) | 9.46e+03 (0, 1) |
| | 20 | 8.61e+03 (3, 1) | 1.95e+05 (0, 5) | 2.85e+02 (4, 0) | 4.91e-05 (6, 0) | 3.72e+04 (0, 4) | 6.75e+03 (3, 1) | 8.98e+03 (1, 2) | 3.05e+04 (0, 4) |
| | 50 | 1.93e+04 (3, 0) | 5.17e+05 (0, 5) | 4.71e+03 (5, 0) | 2.57e+04 (3, 1) | 2.41e+05 (0, 5) | 2.71e+04 (3, 1) | 1.47e+04 (3, 0) | 1.42e+05 (0, 5) |
| | 100 | 3.61e+04 (4, 0) | 1.00e+06 (0, 4) | 1.44e+04 (5, 0) | 2.24e+06 (0, 5) | 7.00e+05 (0, 4) | 7.93e+04 (3, 1) | 5.30e+04 (3, 0) | 4.42e+05 (1, 2) |
| | 200 | 2.85e+05 (4, 0) | 2.05e+06 (0, 4) | 1.25e+05 (6, 0) | 1.44e+07 (0, 5) | 1.82e+06 (0, 4) | 3.86e+05 (3, 1) | 3.64e+05 (3, 1) | 1.34e+06 (1, 2) |
| Michalewicz | 10 | -8.90e+00 (2, 0) | -6.53e+00 (0, 5) | -7.85e+00 (1, 0) | -8.27e+00 (2, 0) | -8.47e+00 (2, 0) | -8.43e+00 (2, 0) | -8.91e+00 (2, 0) | -5.58e+00 (0, 6) |
| | 20 | -1.10e+01 (2, 1) | -1.14e+01 (2, 1) | -8.54e+00 (0, 5) | -1.66e+01 (7, 0) | -1.20e+01 (2, 1) | -1.16e+01 (2, 1) | -1.00e+01 (0, 1) | -8.30e+00 (0, 5) |
| | 50 | -1.61e+01 (2, 1) | -1.71e+01 (2, 1) | -1.18e+01 (0, 6) | -3.04e+01 (6, 0) | -1.68e+01 (2, 0) | -1.63e+01 (2, 1) | -1.64e+01 (2, 1) | -1.48e+01 (0, 6) |
| | 100 | -2.64e+01 (2, 2) | -2.87e+01 (5, 0) | -1.95e+01 (0, 6) | -2.59e+01 (1, 2) | -2.85e+01 (5, 0) | -2.70e+01 (2, 0) | -2.68e+01 (2, 2) | -2.45e+01 (0, 5) |
| | 200 | -4.60e+01 (2, 2) | -5.01e+01 (5, 0) | -3.39e+01 (0, 5) | -3.84e+01 (0, 5) | -4.98e+01 (5, 0) | -4.72e+01 (3, 0) | -4.63e+01 (2, 2) | -4.22e+01 (0, 3) |
| SumCan | 10 | 1.58e+01 (1, 2) | 1.09e+01 (0, 2) | 1.91e+02 (4, 0) | 9.00e+04 (6, 0) | 1.16e+01 (1, 2) | 2.05e+01 (1, 1) | 2.11e+01 (1, 1) | 6.07e+00 (0, 6) |
| | 20 | 7.51e+00 (3, 0) | 3.28e+00 (0, 5) | 1.29e+01 (3, 0) | 5.00e+04 (3, 0) | 3.76e+00 (0, 5) | 7.74e+00 (3, 0) | 6.83e+00 (3, 0) | 1.63e+00 (0, 5) |
| | 50 | 1.50e+00 (2, 0) | 1.03e+00 (1, 3) | 2.87e+01 (0, 6) | 2.38e+00 (4, 0) | 1.09e+00 (1, 3) | 1.77e+00 (4, 0) | 1.73e+00 (4, 0) | 3.93e-01 (0, 4) |
| | 100 | 2.59e-01 (2, 2) | 3.88e-01 (3, 0) | 1.05e-01 (0, 6) | 2.02e-01 (1, 3) | 4.62e-01 (5, 0) | 6.94e-01 (5, 0) | 2.83e-01 (2, 2) | 1.38e-01 (0, 5) |
| | 200 | 7.77e-02 (2, 1) | 1.53e-01 (3, 0) | 3.63e-02 (0, 6) | 5.36e-02 (1, 2) | 1.19e-01 (2, 1) | 2.50e-01 (6, 0) | 8.01e-02 (2, 1) | 4.66e-02 (0, 5) |

the results of pairwise comparisons using Bergmann-Hommel's procedure as the post-hoc test. The significance level for this test is set to 0.05. The first number shows how many algorithms are significantly worse than the algorithm in each column, and the second number shows how many algorithms are significantly better.

The results of statistical tests are evidently consistent with the average algorithms performance. For example, RegEDA-Shr and UMDA do not have a statistically different performance for most of the functions. Also whereas, CMA-ES is able to obtain significantly better BAVs for smaller dimensions of the functions, it rapidly becomes less proficient as the number of variables increases and ends up being significantly worse than many other algorithms. For larger dimensions, one of the proposed RegEDAs are always ranked as first or second (without any statistical difference from the first ranked algorithm) for all functions, except Sphere. The pairwise statistical comparisons also show that these algorithms are able to obtain statistically better BAVs than most other algorithms. Looking at the performance over all functions, RegEDA-LARS-AND appears to have a better overall performance than other RegEDAs for larger problem sizes.

## 6.4 Conclusions

The use of regularized model estimation in EDAs for continuous optimization was studied. Based on the results of the analysis, some of the regularized model estimation methods were selected for use, resulting in four different versions of RegEDA. These different versions were applied to a set of continuous optimization functions, featuring different properties, and the results were compared with those of other Gaussian-based EDAs. The results showed that the increase in problem dimensionality, with a population size which is logarithmic in the number of variables, affects the performance of the proposed RegEDAs less than other Gaussian-based EDAs. Specific problem properties can play a vital role in the algorithm performance. The statistical analysis results showed that RegEDAs are able to obtain BAVs that are significantly better than those of the other algorithms for larger dimensions of most tested functions.

Of all the versions of RegEDA, RegEDA-LARS-AND and RegEDA-Shr have proved to have a better average optimization behavior, with RegEDA-LARS-AND having statistically better overall performance for larger dimensions. The comparison of the behavior of RegEDA-LARS-OR and RegEDA-GL with EDAs using ML estimation (like EMNA) helped to clarify how the use of regularization can affect the optimization performance of EDAs. RegEDA-GL is usually able to maintain a relatively diverse population along the whole evolution path because of its regularized model estimation. We found that this property can help the algorithm to obtain better optimization results in the presence of specific function characteristics.

# Chapter 7

# Regularization in Factorized Gaussian Markov Random Field-Based EDA

## 7.1 Introduction

Approaches to continuous optimization with EDAs [Kern et al., 2004], can be divided into two general categories: (i) Discretization of problem domain and then application of discrete EDAs [Tsutsui et al., 2001]; (ii) Direct application of EDAs based on continuous probabilistic models [Gallagher et al., 1999; Bengoetxea et al., 2002; Bosman and Thierens, 2006]. In the latter approach, Gaussian distributions have been commonly used as probabilistic models in this area [Bosman and Thierens, 2000a; Larrañaga and Lozano, 2001; Ahn et al., 2004], considering either non-overlapping factorized distributions (e.g. continuous UMDA) or the distributions defined by dependencies encoded in PGMs (e.g. GBNs).

In Chapter 2, we saw that most of the research on the use of MNs in EDAs has been mainly focused on discrete domain optimization, like Santana [2003]; Shakya and McCall [2007]; Shakya et al. [2012]. In this chapter a continuous EDA based on the Gaussian distribution is proposed which is analyzed from a GMRF perspective. It is shown that the analysis of undirected models, as it is done in discrete MN-based EDAs, can be also extended to continuous domains. We especially consider the role of the precision matrix and use marginal product factorizations as a particular case of undirected PGMs. The previously studied regularization methods are employed in this type of modeling accompanied by a clustering technique, namely the affinity propagation algorithm, to find the factorized components of the model. The work in this chapter is published in [Karshenas et al., 2013b].

## 7.2 GMRF-Based EDA with Marginal Factorization

Factorization of the joint probability distribution can represent both marginal and conditional independence relationships between the variables. If the factorization is only based on the marginal independence relationships between the sets of variables then it is

called a marginal product factorization or an MPM. MPMs can be represented with both directed and undirected PGMs. In particular, when a GMRF represents an MPM, its set of cliques can be partitioned into non-overlapping groups and therefore it is possible to independently estimate the probability distribution for the variables in each group.

EDAs using MPM representation of the probability distribution are appropriate when variables in the problem can be divided into a number of disjoint groups. Many of the real-world problems actually consist of several smaller components that are either independent or weakly connected. In the discrete case, two well known examples of EDAs that use MPMs are UMDA [Mühlenbein and Paaß, 1996] and EcGA [Harik et al., 2006]. A review of some of the continuous EDAs based on marginal product factorization is given later in Section 7.3.

In this section, we describe a new approach for learning a subclass of GMRFs that represent MPMs. There are several alternatives for learning undirected continuous models within EDAs, from which some are:

i) Estimation of MGD's covariance or precision matrix.

ii) Learning the structure of GMRF and its factors.

iii) Hybrid approaches.

The first approach includes ML estimation as well as other covariance matrix selection techniques discussed in the literature [Yuan and Lin, 2007; Ravikumar et al., 2009; Witten and Tibshirani, 2009]. The methods for inducing an undirected (in)dependence structure between variables are the typical choice in the second approach. Usually these methods use techniques like statistical hypothesis tests or mutual information (entropy) between variables to decide about their (in)dependence. Based on these (in)dependencies, the local neighborhood of each variable is obtained which can then be combined to obtain a full structure and compute the factors for the related variables. The third class of methods combines the computation of the covariance or precision matrix with the identification of the neighborhood structure of the variables. The approach introduced here belongs to the third class of methods.

### 7.2.1 A Hybrid Approach to Learn GMRF

The main idea of our algorithm is to identify the putative neighborhood of each variable by learning a regularized regression model, similar to the first approach in Section 5.2. Knowing that in this model, the dependence of $X_i$ on each of the variables in $\boldsymbol{X} \setminus X_i$ is represented with a weight, in the second step, variables are clustered into disjoint factors according to the strength of their dependency weights. Finally, an MGD is estimated for each factor separately. The main steps of the proposed method are presented in Algorithm 7.1.

#### Detecting Variables Dependencies using Regularization

We have seen that some of the regularization techniques have the promising property of setting some of the model parameters exactly to zero. In the case of regularized regression models, these parameters are the weights of the dependencies between variables.

```
1 for each variable do
2     Compute its linear dependence weights on all other variables
3 end for
4 Cluster the variables into disjoint cliques using the weight matrix
5 for each clique do
6     Estimate an MGD for the variables in the clique
7 end for
```

Algorithm 7.1: Hybrid GMRF-EDA learning algorithm

Therefore, the use of regularization leads to sparser dependency structures which can be partitioned into separate groups of variables. The three techniques considered in the first approach to regularized model learning in Section 5.2 are used here to obtain the weight of dependencies between variables: LARS, LASSO and ELNET.

### Clustering by Affinity Propagation

Clustering methods are used to group objects into different sets or clusters, in such a way that each cluster comprises similar objects. Clusters can then be associated to labels that are used to describe the data and identify their general characteristics. Among the best known clustering algorithms are k-means [Hartigan and Wong, 1979] and k-center clustering [Agarwal and Procopiuc, 2002] methods.

Affinity propagation (AP) [Frey and Dueck, 2007] is a recently introduced clustering method which takes as input a set of measures of similarity between pairs of data points and outputs a set of clusters of those points. For each cluster, a typical or representative member, which is called exemplar, is identified. We use AP as an efficient way to find the MN neighborhood of the variables from the linear weights output by regularized regression methods.

AP takes as input a matrix of similarity measures between each pair of points and a set of preferences which are a measure of how likely each point is to be chosen as exemplar. The algorithm works by exchanging messages between the points until a stop condition, which reflects an agreement between all the points with respect to the current assignment of the exemplars, is satisfied. These messages can be seen as the way the points share local information in the gradual determination of the exemplars. For more details on AP, see [Frey and Dueck, 2007].

In the context of our GMRF-EDA, each variable will correspond to a point and as the similarity between two points $X_i$ and $X_j$, the absolute value of weight $w_{ij}$ obtained from the regularized regression model of variable $X_i$ is used. Since in general $w_{ij} \neq w_{ji}$, the similarity matrix is not symmetric. AP also takes advantage of the sparsity of the similarity matrix obtained from regularized estimation, when such distribution of similarity values is available. The message-passing procedure may be terminated after a fixed number of iterations, when changes in the messages fall below a threshold, or after the local decisions stay constant for some number of iterations.

#### Estimating MGDs for the Factors

In the final step of the proposed hybrid model learning algorithm, an MGD of the variables in each factor is estimated. We expect that the factorized distribution obtained by this model estimation will give a more accurate estimation of the target MGD (of all variables) in comparison to learning a single multivariate distribution for all of the problem variables.

### 7.2.2 Sampling Undirected Graphical Models for Continuous Problems

Sampling is one of the most problematic steps in EDAs based on undirected PGMs. The problem is mainly related to the loops existing in the model structure that does not allow a straightforward application of simple sampling techniques like PLS method [Henrion, 1986] for BNs. The application of Markov Chain Monte Carlo (MCMC) methods like Gibbs sampling [Geman and Geman, 1984] also has a high computational complexity.

Obtaining decomposable approximations of MNs to allow the application of PLS algorithm [Santana, 2003], merging cliques of an MN to capture as many dependencies as possible before applying exact or approximate sampling algorithms [Höns, 2005], and computation of MPE based on belief propagation [Mendiburu et al., 2007] are among other options for sampling undirected graphical models that have been used in discrete EDAs and could be applied to continuous problems. The method we adopt here is to independently sample the MGDs for each of the factors.

## 7.3 Related Work

A number of works have proposed the use of MPMs for continuous problems. In [Fossati et al., 2007] and [Li et al., 2008] two different algorithms are proposed that learn variants of EcGA for real-valued problems. Both algorithms are based on discretizing the continuous variables previous to the construction of MPM. Lanzi et al. [2008] propose a version of EcGA that instead of mapping real values into discrete symbols, models each cluster of variables using a multivariate probability distribution and guides the partitioning process using an MDL metric for continuous distributions. To learn the clusters of variables, an adaptive clustering algorithm, namely the BEND random leader algorithm, is used. Recently, Dong et al. [2011] have proposed to compute the correlation matrix as an initial step to do a coarse learning such as identifying weakly dependent variables. These variables are independently modeled using a univariate Gaussian distribution while the other variables are randomly projected into subspaces (clusters) that are modeled using an MGD.

Regularization techniques have been very recently used for learning MNs in EDAs. They have been used for optimization based on undirected models both in discrete [Santana et al., 2011] and continuous domains [Karshenas et al., 2011a]. In [Malagó et al., 2011], the task of selecting the proper structure of the Markov network is addressed by using $\ell_1$-regularized logistic regression technique. In [Ochoa, 2010], the class of shrinkage EDAs has been introduced. The results presented there show that shrinkage regularization can dramatically reduce the critical population size needed by EMNA in the optimization of continuous functions.

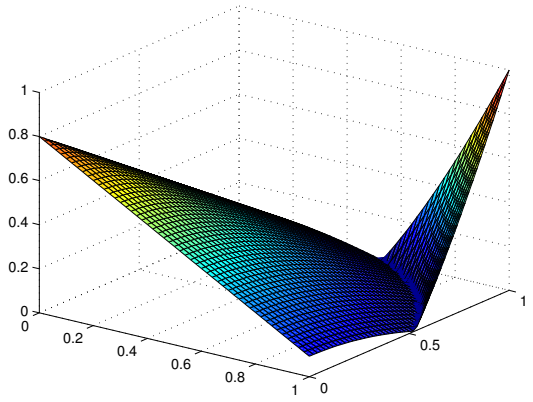



Figure 7.1: 2-dimensional continuous trap function

Finally, AP has also been applied in EDAs. AffEDA [Santana et al., 2010b], uses this method to learn MPMs for discrete optimization with EDAs. In a different context, AP has been applied as a niching procedure for EDAs based on Markov chain models [Chen and Hu, 2010].

## 7.4 Experiments

The main objective of our experiments is to study the proposed GMRF-EDA in optimization when learning a factorized distribution model, and compare its behavior with that of other EDAs. For this purpose, RegEDA-Shr and RegEDA-GL from the previous section and continuous UMDA [Larrañaga et al., 2000a], the EDA that assumes a total independence between the variables are selected. We also analyze the accuracy of the estimated methods in recovering an accurate structure of the problem.

### 7.4.1 Benchmark Functions

Two classes of functions that represent completely different domains of difficulty in terms of optimization are used to evaluate the performance of the algorithms: an additive deceptive function and a simplified protein folding model. The 2D-deceptive function [Pelikan et al., 2003], which is a maximization problem, is composed of the aggregation of 2-dimensional continuous trap functions which have a local optimum with a large basin of attraction and an isolated global optimum (Figure 7.1)

$$f_{2D-deceptive}(\mathbf{x}) = \sum_{i=1}^{n/2} f_{2D-trap}(x_{2i-1}, x_{2i})$$

where

$$f_{2D-trap}(x,y) = \begin{cases} 0.8 - \sqrt{\frac{x^2+y^2}{2}} & \text{if } \sqrt{\frac{x^2+y^2}{2}} \leq 0.8 \\ -4 + 5\sqrt{\frac{x^2+y^2}{2}} & \text{otherwise} \end{cases}.$$

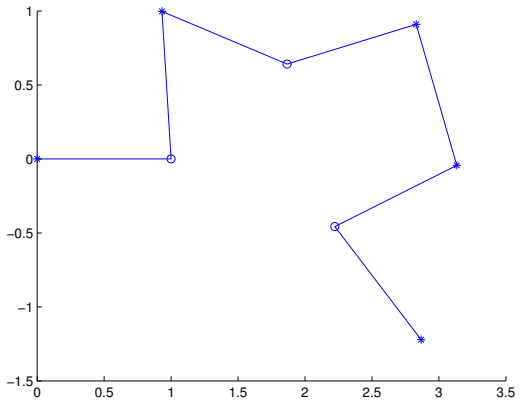



Figure 7.2: One possible configuration of the Fibonacci sequence $S_5 = BABABBAB$

Off-lattice models are simplified protein models that, in opposition to the HP simplified model [Dill, 1985], do not follow a given lattice topology. Instead, the 2D or 3D coordinate in the real axis define the positions of the protein residues. Among the off-lattice models with known lowest energy states is the $AB$ model [Stillinger et al., 1993], where $A$ stands for hydrophobic and $B$ for polar residues. The distances between consecutive residues along the chain are held fixed to $b = 1$, while non-consecutive residues interact through a modified Lennard-Jones potential. There is an additional energy contribution from each angle $\theta_i$ between successive bonds. The energy function for a chain of $n$ residues, which should be minimized, is shown in Equation (7.1).

$$E = \sum_{i=2}^{n-1} E_1(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} E_2(r_{ij}, \zeta_i, \zeta_j), \tag{7.1}$$

where

$$E_1(\theta_i) = \frac{1}{4}(1 - cos\theta_i)$$

and

$$E_2(r_{ij}, \zeta_i, \zeta_j) = 4(r_{ij}^{-12} - C(\zeta_i, \zeta_j) r_{ij}^{-6}).$$

Here, $r_{ij}$ is the distance between residues $i$ and $j$ (with $i < j$). Each $\zeta_i$ is either A or B, and $C(\zeta_i, \zeta_j)$ is $+1$, $+\frac{1}{2}$, and $-\frac{1}{2}$ respectively for $AA$, $BB$, and $AB$ pairs, giving strong attraction between $AA$ pairs, weak attraction between $BB$ pairs, and weak repulsion between $A$ and $B$ pairs [Hsu et al., 2003]. We only consider Fibonacci sequences defined recursively by

$$S_0 = A, \quad S_1 = B, \quad S_{i+1} = S_{i-1} * S_i \tag{7.2}$$

where $*$ is the concatenation operator. Figure 7.2 shows a possible configuration for sequence $S_5 = BABABBAB$.

A 2D off-lattice solution of the $AB$ model can be represented as a set of $n-2$ angles. Angles are represented as continuous values in $[0, 2\pi]$. The first two residues can be fixed at positions $(0, 0)$ and $(1, 0)$. The other $n-2$ residues are located from the angles with respect to the previous bond. We look for the set of angles that defines the optimal

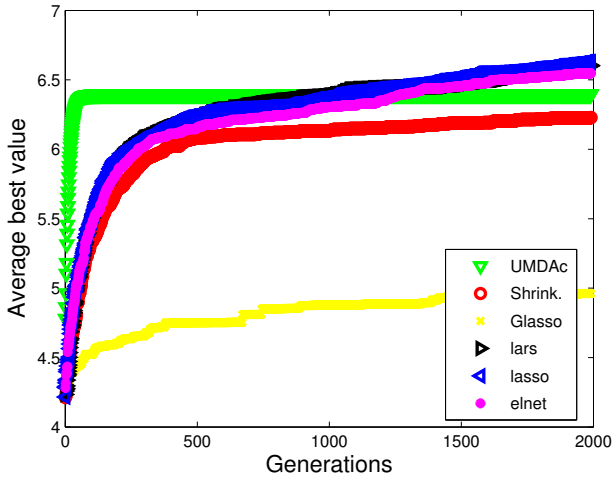



Figure 7.3: Results of different EDAs for the $f_{2D-deceptive}$ function.

off-lattice configuration. As instances, we consider Fibonacci sequences for numbers 6 and 7. The respective sizes of these sequences, in the same order, are $n = 21$ and $n = 34$.

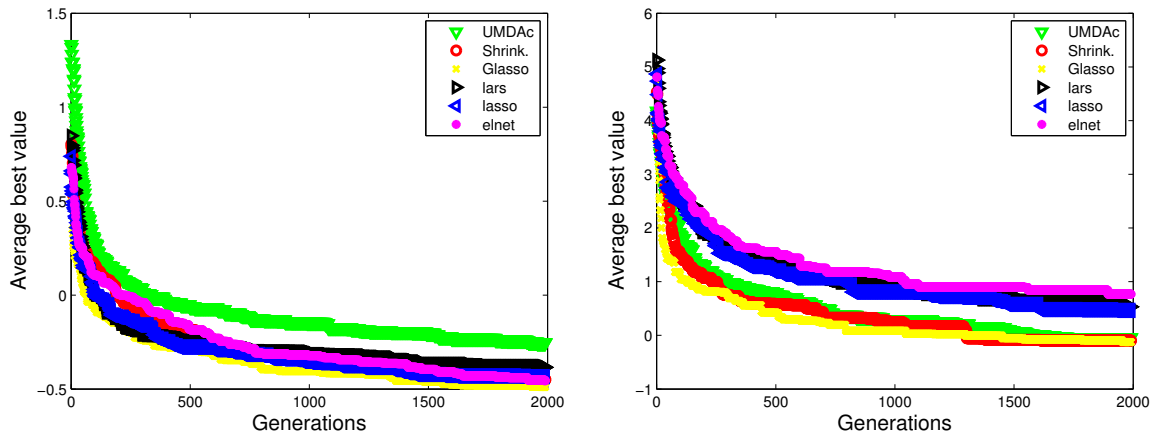
### 7.4.2 Results

Figure 7.3 shows the average BAVs in each generation of different algorithms for the 2D-deceptive function with 30 variables. All of the EDA variants in this experiment use truncation selection with $\tau = 0.5$. The population size is set to $N = 5n$ and the maximum number of generations is 2000. The results are averaged over 30 independent runs.

It can be appreciated that UMDA, which considers a fixed total factorization of the distribution, starts outperforming other algorithms in the earlier generations but as the evolution continues, the initial lead of this algorithm is lost. The EDAs based on regularized model learning with graphical LASSO and shrinkage estimation methods, which consider no explicit factorization of the distribution, have a poor behavior and are not able to reach the best solutions achieved by UMDA. On the other hand, the hybrid GMRF-EDA, with different regularization techniques (LARS, LASSO and ELNET), is able to constantly improve the average BAV and obtain better results at the end of optimization.

Next, we investigate the behavior of the algorithms on the off-lattice protein folding model. It is worth to mention that this is a very hard problem where the structural interactions between the variables are not clearly defined. There seems to be dependencies between adjacent variables in the AB sequence. However, the way in which other interactions between the AB residues are translated to dependencies in the model is not clear.

Figure 7.4 shows the average BAVs obtained during evolution for the off-lattice protein folding models corresponding to Fibonacci sequences 6 and 7. For the smaller instance of this problem (with sequence number 6) the results are averaged over 30 independent runs and for the larger instance over 15 runs. It can be seen that UMDA is outperformed by all other EDAs from the initial stages of the search for the first folding problem (Figure 7.4(a)). In this instance, the best results are achieved when estimating a single MGD with graphical LASSO, although the MPMs learnt with ELNET regularization




(a) Sequence $s_6$ (b) Sequence $s_7$

Figure 7.4: Results of different EDAs for two different instances of the off-lattice AB models.

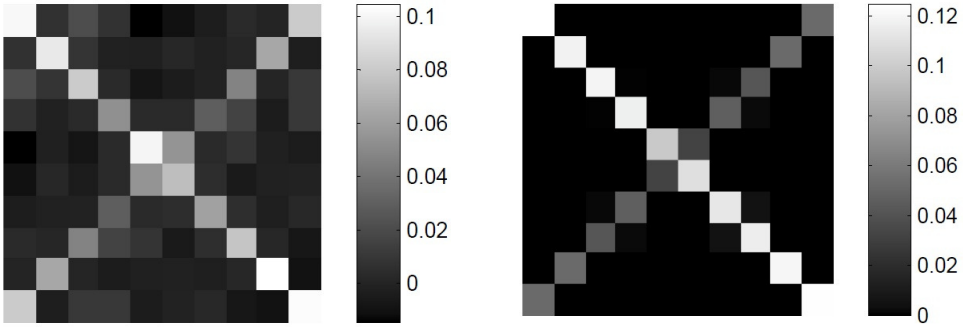
technique in the hybrid GMRF-EDA closely follows.

On the second instance of the off-lattice protein problem (Figure 7.4(b)), the performance of UMDA gets very close to RegEDA-GL and RegEDA-Shr, and outperforms GMRF-EDA which tries to decompose the problem by learning dependencies between variables. There seems to be an important variability between the characteristics of the different off-lattice protein instances. In some situations such as the Fibonacci sequence number 6, capturing dependencies between the variables of the problem contributes to improvement of the quality of the obtained solutions. However, there are cases where learning the dependencies explicitly does not improve the results of the simpler univariate models. Similar optimization behavior has also been reported for some of the discrete additively decomposable problems like 2D Ising spin glasses [Hauschild et al., 2007] where the necessity of discovering the dependencies between variables is not clear.

An interesting issue is to observe the disparate behavior exhibited by RegEDA-GL. It behaves very different in comparison to all other algorithms for 2D-deceptive function, achieving the worst results. Nevertheless, it reaches the best results for the off-lattice protein models. More research on this type of regularized model learning is needed to discern which mechanisms explain the difference between the performance of the algorithm for these two classes of functions.

### 7.4.3 Influence of the Regularization Parameter

The properties of regularized model learning were studied from different points of view in Chapter 5. There, we saw how the value of the regularization parameter can affect the models estimated with graphical LASSO algorithm (Section 5.3.4). To extend that study, in this section we also examine how the change in the values of this parameter can influence the models estimated during evolution in RegEDA-GL. For this purpose, the models estimated for optimizing the 2D-deceptive function with cross-related variables




(a) $\lambda = 0.001$ (b) Dynamic $\lambda$

Figure 7.5: Precision matrices learned by two different learning methods

(i.e. first with last, second with penultimate, etc.) are analyzed.

Since the variable dependency structure of this problem is known, we can study the accuracy of the estimated model structures when using different configurations for setting the regularization parameter ($\lambda$) during optimization. Figure 7.5 represents the precision matrices obtained at the end of an RegEDA-GL run on 2D-deceptive function with 10 variables with two different configurations.

The precision matrix depicted in Figure 7.5(a) is obtained when using a constant value for the regularization parameter throughout the whole EDA run. On the contrary, the precision matrix shown in Figure 7.5(b) is obtained when EDA starts with a small value for the regularization parameter in the early generations and dynamically increases it along the evolution. As it can be seen, while using a constant value for the regularization parameter a relatively good structures can be estimated, but with a dynamic parameter setting, this regularization method can obtain a better estimation of the MGD's precision matrix and its corresponding GMRF structure.

## 7.5 Conclusions

Based on the correspondence between MGDs and GMRFs, we discussed some topics in learning and sampling these probabilistic graphical models, when employed in EDAs. Specifically, some of the methods that can be used to approach the learning and sampling of GMRFs were presented. GMRF-EDA was introduced as an algorithm that combines regularized regression with affinity propagation to learn regularized MPMs. This is a different approach to learn more accurate MPMs for continuous problems with separable components. Preliminary results on the continuous 2D-deceptive function showed promising optimization performance of this regularized model learning approach.

The proposed method was also tested on the real-world problem of predicting the secondary structure of proteins using the off-lattice protein folding model. The results of the experiments on two instances of this problem showed that the proposed hybrid GMRF-EDA has a comparable performance on one of the instances to that of RegEDAs which learn a single MGD without factorization, but is outperformed on the other. We also studied the influence of regularization parameter setting during optimization to de-

termine how the characteristics of the regularization method influence the outcome of the EDAs.

# Part III

# Multi-objective Optimization

# Chapter 8

# Evolutionary Multi-Objective Optimization with Probabilistic Modeling

## 8.1 Introduction

Many of the real world optimization problems are often best characterized by more than one objective. When trying to solve these problems, several, possibly conflicting, criteria should be fulfilled simultaneously. Usually none of the objectives can be preferred over the others by the decision maker (DM) in order to apply single-function optimization methods for solving these problems. Moreover, the existence of conflicting criteria means that trying to improve one of the objectives will result in worse values for some other. Therefore, it is more reasonable to solve these problems by considering all the objectives together.

Let $\mathcal{F} = \{f_1, \ldots, f_m\}$ be the set of objective functions for a problem, where each objective function is defined as $f_j : \mathbb{D} \mapsto \mathbb{R}$, and assume that all of them should be minimized. Then an unconstrained continuous multi-objective optimization problem (MOP) is defined as

$$\min_{\boldsymbol{x} \in \mathbb{D} \subseteq \mathbb{R}^n} \; \boldsymbol{f}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\big). \tag{8.1}$$

$\mathbb{D}$ determines the set of feasible value-settings for the vector of input variables $\boldsymbol{X} = (X_1, \ldots, X_n)$ and sometimes it is referred to as the decision space. The objective functions of an MOP map the solutions in $\mathbb{D}$ to another space which is called the objective space of the problem.

In general, with more than one objective function, the optimum solution to the MOP in Equation (8.1) is not unique anymore, even if the constituting objective functions are not multi-modal. Thus, the goal of a multi-objective optimization algorithm is to search for solutions which result in an optimal trade-off between different objectives of the problem. One of the most frequently used approaches is to employ the notion of Pareto optimality [Pareto, 1897] and the corresponding Pareto dominance relation [Goldberg, 1989].

**Definition 8.1. (Pareto Dominance)** *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two solutions of the MOP defined in Equation* (8.1) *in the decision space $\mathbb{D}$. Then $\boldsymbol{x}$ Pareto dominates $\boldsymbol{y}$, denoted as $\boldsymbol{x} \prec \boldsymbol{y}$,*

*if and only if:*

1. $\forall f_j \in \mathcal{F} \quad f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{y})$, *and*
2. $\exists f_k \in \mathcal{F} \quad f_k(\boldsymbol{x}) < f_k(\boldsymbol{y})$.

Pareto dominance defines a *partial* order between the solutions, i.e. an irreflexive, antisymmetric and transitive relation. Using this relation, the solutions to an MOP can be ranked into a number of disjoint Pareto non-dominated sets (or Pareto sets for short) where every two solutions in the same set have exactly equal objective values or are *incomparable* in terms of Pareto dominance. The corresponding projections of these Pareto sets onto the objective space are called Pareto non-dominated fronts (similarly Pareto fronts). The best Pareto set containing optimal trade-off solutions (no other solution in $\mathbb{D}$ dominates them) is called the Pareto optimal set, denoted as $\mathbb{D}^*$, and its projection onto the objective space is called the Pareto optimal front. No further improvement in one of the objectives of the solutions in the Pareto optimal set can be obtained without aggravating another objective. In general, the number of solutions in the Pareto optimal set can be exponentially large or even infinite in the case of continuous domains and thus multi-objective optimization algorithms try to obtain a good approximation of this set.

Because of the complexity of solving MOPs, a promising approach which is gaining an increasing interest is to use EAs, giving rise to evolutionary multi-objective optimization (EMO) algorithms [Zitzler et al., 2000; Deb, 2001; Abraham et al., 2005; Coello Coello et al., 2007], otherwise known as multi-objective EAs. Although these algorithms do not ensure the optimality of the solutions compared with some of the conventional mathematical optimization techniques, they have been successfully applied to many complex real-world MOPs. Besides their simple mechanism which motivates their wide-spread use in different applications, another advantage of these algorithms encouraging their use for multi-objective optimization is their population-based search allowing them to simultaneously explore several regions of the search space, and find a set of trade-off solutions in contrast to a single solution, in an individual run.

A successful strategy adopted by many EMO algorithms is to use the same solution reproduction method (see Figure 2.1) as the one used in single-objective EAs (e.g. crossover and mutation operators) and only modify the way solutions are selected for reproduction or replacement by considering several objectives for solution evaluation. Specifically, many of the methods that are proposed for solution ranking in EMO algorithms are based on the Pareto dominance relation. One of the EA classes that has been used for multi-objective optimization is EDAs.

## 8.2 EDAs in Multi-Objective Optimization

In Chapter 2, we discussed some of the main motivations for the use of probabilistic modeling in EAs. In the context of multi-objective optimization, the traditional genetic operators used for solution reproduction in EAs usually provide a good and efficient exploration of the search space. However, the same shortages explained for traditional EAs in single-objective optimization can affect the effectiveness of their search for the optimal solutions of an MOP. Therefore, several multi-objective EDAs have been proposed in the literature [Thierens and Bosman, 2001; Pelikan et al., 2006b; Martí et al., 2010],

which combine the model building in EDAs with a multi-objective solution ranking and selection mechanism to solve MOPs. In the following, we review some of the methods proposed in the literature for multi-objective optimization based on probabilistic model estimation. A summary of the multi-objective EDAs is given in Table 8.1.

### 8.2.1 A Survey of Multi-Objective EDAs

In several multi-objective EDAs proposed in the literature, a BN is estimated as the probabilistic model. Pareto BOA [Schwarz and Očenášek, 2001] integrates the Pareto strength-based solution ranking method [Zitzler and Thiele, 1999] into BOA [Pelikan et al., 1999] for multi-objective optimization. In a similar study, the non-dominated sorting algorithm [Deb et al., 2002a] is used in BOA [Khan et al., 2002]. Bayesian multi-objective optimization algorithm (BMOA) [Laumanns and Očenášek, 2002] uses an $\epsilon$-Pareto dominance based ranking method (a relaxed version of the Pareto dominance relation) to select a subset of solutions for estimating a BN. Decision tree based multi-objective EDA (DT-MEDA) [Zhong and Li, 2007] uses regression trees with Gaussian kernels in their leaves as the probabilistic model. It uses a slightly modified version of the non-dominated sorting algorithm for selecting a subset of solutions in each generation.

Some of the proposed EDAs explicitly estimate a mixture of probability distributions by means of a clustering method to obtain a well-spread approximation of the Pareto solutions. Multi-objective mixture-based IDEA (MIDEA) [Bosman and Thierens, 2002] clusters the selected solutions into several groups in the objective space and learns a separate component for each group of solutions. Probabilistic models with different orders of complexity (e.g. encoding univariate, bivariate or multi-variate dependencies) are tested as the components of the mixture. The proposed algorithm is also further improved by maintaining an $\epsilon$-Pareto archive and introducing adaptive variance scaling to prevent premature convergence in continuous MOPs [Bosman and Thierens, 2007].

Multi-objective Parzen-based EDA (MOPEDA) [Costa and Minisci, 2003] applies a Parzen estimator to learn a mixture of kernel functions in order to reduce the variance of the probability distribution estimation. Both Gaussian and Cauchy kernels are used alternatively during evolution. Voronoi-based EDA (VEDA) [Okabe et al., 2004b] constructs a Voronoi diagram by considering the inferior solutions in addition to good solutions in each generation. It also uses principal component analysis to reduce the dimensionality of the objective space.

In multi-objective hierarchical BOA (mohBOA) [Pelikan et al., 2005] each component of the mixture is a BN. To obtain a well-distributed approximation of the Pareto front, approximately equal shares are allocated to the mixture components during model sampling. In the multi-objective EcGA (meCGA) [Sastry et al., 2005] an MPM is used for each component of the mixture. The algorithm is later improved by using an $\epsilon$-Pareto dominance based clustering and algorithm parameters are dynamically computed during evolution [Soh and Kirley, 2006].

In addition to clustering the solutions, the diversity preserving multi-objective rBOA (dp-MrBOA) [Ahn and Ramakrishna, 2007] decomposes the problem variables by estimating a GBN. It employs a diversity preserving selection method which uses adaptive sharing and dynamic crowding methods. Regularity model-based multi-objective EDA (RM-MEDA) [Zhang et al., 2008] estimates a piece-wise continuous manifold using the

Table 8.1: Summary of the multi-objective EDAs with their ranking methods and probabilistic models used to search in the space of candidate solution.

| Name | Probabilistic Model | Mixture | Hybridized | Ranking Method | External Archive |
|---|---|---|---|---|---|
| Pareto BOA [Schwarz and Očenášek, 2001] | Bayesian network | | | Pareto dominance, Pareto strength, Domination count | • |
| mBOA [Khan et al., 2002] | Bayesian network | | | Pareto dominance, Crowding distance | |
| MIDEA [Bosman and Thierens, 2002] | Univariate distribution, Tree-structured distribution | • | | Domination count, Spread in objective space | |
| SDR-AVS-MIDEA [Bosman and Thierens, 2007] | Multivariate Gaussian distribution | • | | Domination count, Spread in objective space | • |
| MOPEDA [Costa and Minisci, 2003] | Gaussian and Cauchy kernel | • | | Pareto dominance, Crowding distance | |
| VEDA [Okabe et al., 2004b] | Geometric distribution with uniform noise | • | | Pareto dominance, Crowding distance | • |
| mohBOA [Pelikan et al., 2005] | Bayesian network | • | | Pareto dominance, Crowding distance | |
| meCGA [Sastry et al., 2005] | Marginal product model | • | | Pareto dominance, Crowding distance | |
| moPGA [Soh and Kirley, 2006] | Univariate marginal product model | • | | Pareto dominance, Crowding distance | • |
| dp-MrBOA [Ahn and Ramakrishna, 2007] | Product of Gaussian Bayesian networks | • | | Pareto dominance, Domination count, Modified crowding distance | |
| RM-MEDA [Zhang et al., 2008] | Linear combination of principal components with Gaussian noise | • | | Pareto dominance, Crowding distance | |
| MORBM [Tang et al., 2010] | RBM network | • | | Pareto dominance, Crowding distance | |
| MB-GNG [Martí et al., 2011] | Univariate Gaussian distribution | • | | Hypervolume indicator | |
| MARTEAD [Martí et al., 2012] | Univariate Gaussian distribution | • | | Hypervolume indicator estimation | |
| BMOA [Laumanns and Očenášek, 2002] | Bayesian network | | | $\epsilon$-Pareto dominance, Domination count | |
| DT-MEDA [Zhong and Li, 2007] | Gaussian Bayesian network | | | Pareto dominance, Crowding distance | |
| Tabu-BOA [Katsumata and Terano, 2003] | Bayesian network | | • | Pareto dominance | • |
| MOHEDA [Li et al., 2004] | Univariate distribution | • | • | Weighted sum | |
| EDA-PSO [Gao et al., 2010] | Univariate Gaussian distribution | | • | Pareto dominance, Crowding distance | |
| PLREDA [Shim et al., 2013] | RBM network | • | • | Pareto dominance, Crowding distance, Probabilistic dominance | |
| hNSEA, hMOEA/D [Shim et al., 2012b] | RBM network | | • | Pareto dominance, Crowding distance Decomposition with Tchebycheff method | |
| UMEGS, UMSA, UMHC [Shim et al., 2012a] | Univariate distribution | | • | Decomposition with Tchebycheff method | |

local principal component analysis algorithm. Each mixture component is an affine plus a Gaussian noise.

In restricted Boltzmann machine (RBM)-based EDA [Tang et al., 2010] each mixture component is an RBM, a stochastic neural network with hidden neurons. Model building growing neural gas (MB-GNG) algorithm [Martí et al., 2011] uses a specific single-layer neural network, called growing neural gas, to determine the location of mixture components which are Gaussian distributions. The approach is further extended in [Martí et al., 2012] by using the adaptive resonance theory and employing a hypervolume indicator-based selection method [Bader and Zitzler, 2011].

There are also some approaches which have combined the EDA search method with other search heuristics. In multi-objective hybrid EDA (MOHEDA) [Li et al., 2004] an EDA based on a mixture of univariate models is hybridized with a local search method that is applied to the solutions generated from the probabilistic model of EDA. Tabu-BOA [Katsumata and Terano, 2003] uses the Tabu lists maintained in Tabu search to improve multi-objective BOA. Gao et al. [2010] proposed an algorithm which hybridizes an EDA based on univariate distributions with a particle swarm optimization (PSO) algorithm. Very recently, the RBM-based EDA has also been hybridized with PSO for noisy multi-objective optimization [Shim et al., 2013].

While most of the multi-objective EDAs use selection methods based on Pareto dominance, there are also some works which have studied EDAs in other multi-objective optimization frameworks. Shim et al. [2012b] proposed a multi-objective optimization algorithm with dynamic combination of operators in GAs, differential evolution and EDAs during search. The proposed algorithm is tested when using both Pareto dominance-based and decomposition-based selection [Zhang and Li, 2007] methods. They have also used decomposition-based selection in an algorithm which hybridizes an EDA based on univariate distributions with several local search methods [Shim et al., 2012a].

As mentioned before, although in EDAs probabilistic models are used to estimate the values for problem variables and to generate new solutions based on these estimations, probabilistic modeling has also been used for estimating the values of objective functions. For example, Zhang et al. [2010] proposed a decomposition-based MOEA which uses Gaussian processes to estimate surrogate models of the objectives in MOPs with computationally expensive (cost or time) objective functions.

## 8.3 Conclusions

EAs are one of the promising methods for multi-objective optimization, i.e. when there is more than one objective function in the problem. EMO algorithms often only modify the solution ranking and selection mechanism of EAs to account for the existence of several objectives in the problem. Pareto dominance relation is a popular method used in many of these algorithms for solution ranking. As a class of EAs, several multi-objective EDAs have also been proposed in the literature. In the next chapter, we will see that more objectives in the problem can also affect how the new solutions are being generated.

# Chapter 9

# Evolutionary Multi-Objective Optimization with Joint Probabilistic Modeling

## 9.1 Introduction

Although most of the study on multi-objective optimization has been focused on problems with two or three objectives, very often real-world MOPs involve many criteria. Thus, after the initial success of EMO algorithms in solving MOPs with few objectives, several efforts have been made to study the performance of these algorithms on the so called many-objective problems (when there is more than three objectives in the problem) [Deb et al., 2002b; Ishibuchi et al., 2008]. One of the main lines of research in many-objective optimization is the analysis of relationships between objectives in order to simplify the MOP at hand. Different methods, like correlation and principal component analysis [Deb and Saxena, 2005; Goel et al., 2007; López Jaimes et al., 2008; Craft and Bortfeld, 2008; Verel et al., 2011], extending the definition of conflicting objectives [Brockhoff and Zitzler, 2009], and linear programming [Guillén-Gosálbez, 2011] have been proposed for this purpose. These methods reduce optimization complexity by searching for a minimum subset of objectives.

In this chapter, we introduce joint probabilistic modeling in the context of EDAs as a new approach in this direction. The principal idea of this method is to incorporate the objectives into model learning of EDAs when solving an MOP. In this way, the estimated model cannot only capture the interactions between variables, as it is done in other EDAs, but also obtain an approximation of the relationships between objectives and variables. We use BNs for joint modeling which allow to capture more complex patterns of interaction than just linear correlation. Moreover, since the estimated model is used for sampling new candidate solutions, this type of relationships are implicitly considered when exploring the search space, and as shown in this study they are useful for solving MOPs. The approach presented in this chapter is published in [Karshenas et al., 2011b].

## 9.2 Joint Probabilistic Modeling in Multi-Objective EDAs

The common practice in EAs (including most of EDAs) is to only use the variable values for generating new solutions in the search space. The objective values are used for ranking and selecting a subset of solutions, and apart from that, these algorithms ignore the objective information when generating new solutions. Although this scheme of new solution generation usually offers a relatively good exploration of the search space, the objective information of the selected solutions can be exploited in the new solution generation step of EAs for further improvement, so that the new solutions have better or comparable objective values than their parents.

In the case of EDAs, when objective information is incorporated into probabilistic modeling, the estimated model not only encodes the characteristics of the variable values of the selected solutions, but it also encodes preferences concerning objective values of these solutions. This especially applies to MOPs, where, because of the existence of several objectives, more information about the quality of the solutions is available. EDAs try to represent the problem structure by probabilistically approximating the interactions between variables and how their values influence the objective functions. Incorporating the objective values of the solutions in the modeling step of these algorithms will allow them to obtain a probabilistic approximation of the relationships between objectives as well (e.g. based on the expected value of the objectives). Furthermore, the estimated model structure helps to identify redundancy, conditional independence, or other types of relationships between objectives.

We have chosen BNs for the purpose of joint modeling of objectives and variables in this study. As it was discussed in Chapters 1 and 2, due to its inherent properties, this probabilistic model has been employed in many EDAs for both single and multi-objective optimization. When a joint probabilistic model of objectives and variables is learnt with a BN, the conditional probabilities of the nodes corresponding to the variables can be dependent on the objective nodes, according to the BN structure. Therefore, when new solutions are being sampled from the estimated BN, the objective values can directly affect the new values generated for the variables. Figure 9.1 shows an example of joint modeling of objectives and variables encoded in a BN.

Apart from obtaining a decomposition of the problem through the factorization of

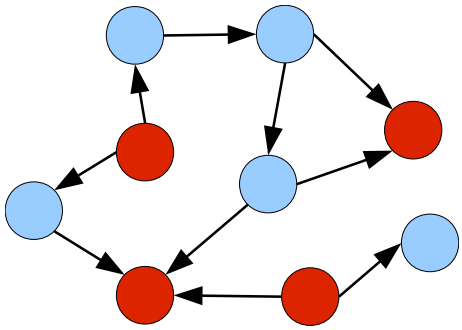


Figure 9.1: An example of a BN structure encoding a joint modeling of 4 objectives and 5 variables.

joint probability distribution, joint modeling performs an implicit variable selection for each of the objectives considering the MB of the objective nodes in BN structure. In continuous domains, assuming a Gaussian distribution for the joint vector of variables and objectives, $(\boldsymbol{X}, \boldsymbol{Q}) = (X_1, \ldots, X_n, Q_1, \ldots, Q_m)$, where $Q_j$ is a random variable taking values from the range of the $j$th objective function, i.e. $Q_j = f_j(\boldsymbol{X})$, a GBN is estimated as the probabilistic model.

### 9.2.1 Discussion

By estimating a probability distribution from a set of selected solutions, EDAs try to model the regions of the search space that are represented by these solutions. In doing so, these algorithms assume that the selected solutions represent promising regions of the search space, and that further exploration and exploitation of these regions (which in EDAs are interchanged automatically based on the distribution of the solutions) will guide the search toward optimal solutions. Using probabilistic models will allow EDAs to both discover and take advantage of the useful regularities in the set of selected solutions for better optimization, as it has been explained in Chapter 2.

In a typical EDA, all of the solutions used for estimating the probabilistic model are equally weighted and are treated in the same way. In other words, the quality of solutions are not taken into consideration in the probabilistic modeling of the search space. Instead, it is the density of the selected solutions in the search space that drives model estimation and therefore the algorithm will be completely blind to the quality of the new candidate solutions sampled from the estimated model.

When the quality information of the solutions is integrated into model estimation, it adds another extent to the probabilistic model concerning the regions of the objective space that correspond to the objective values of the selected solutions. Since the solutions are often selected for model learning according to their objective values, the estimated model encodes best found regions of the objective space. Hence, the new candidate solutions sampled from the joint probabilistic model are more likely to fall in these regions of the objective space, essentially helping the algorithm to find better candidate solutions in each generation. Moreover, isolated solutions with good objective values have a better chance of reproduction when estimating a joint probabilistic model. This is because such a model can encode the relationships between the promising regions in the objective and search spaces, and thus sampling different regions of the search space will be influenced (i.e. controlled) by their approximated qualities.

The employment of similar approaches in a few single-objective EDAs like EBCOA Miquélez et al. [2004, 2006] and DEUM Shakya and McCall [2007] is already shown to help achieving better optimization results. However, with the existence of more than one objective in multi-objective optimization, the joint probabilistic model estimation should account for several, possibly conflicting objective values of the solutions. Using an expressive probabilistic model like BN allows to capture the relationships between different objectives when modeling promising regions of the objective space, and therefore the EDA based on such a model can adaptively combine the distinct quality information available when sampling new candidate solutions from the estimated probabilistic model.

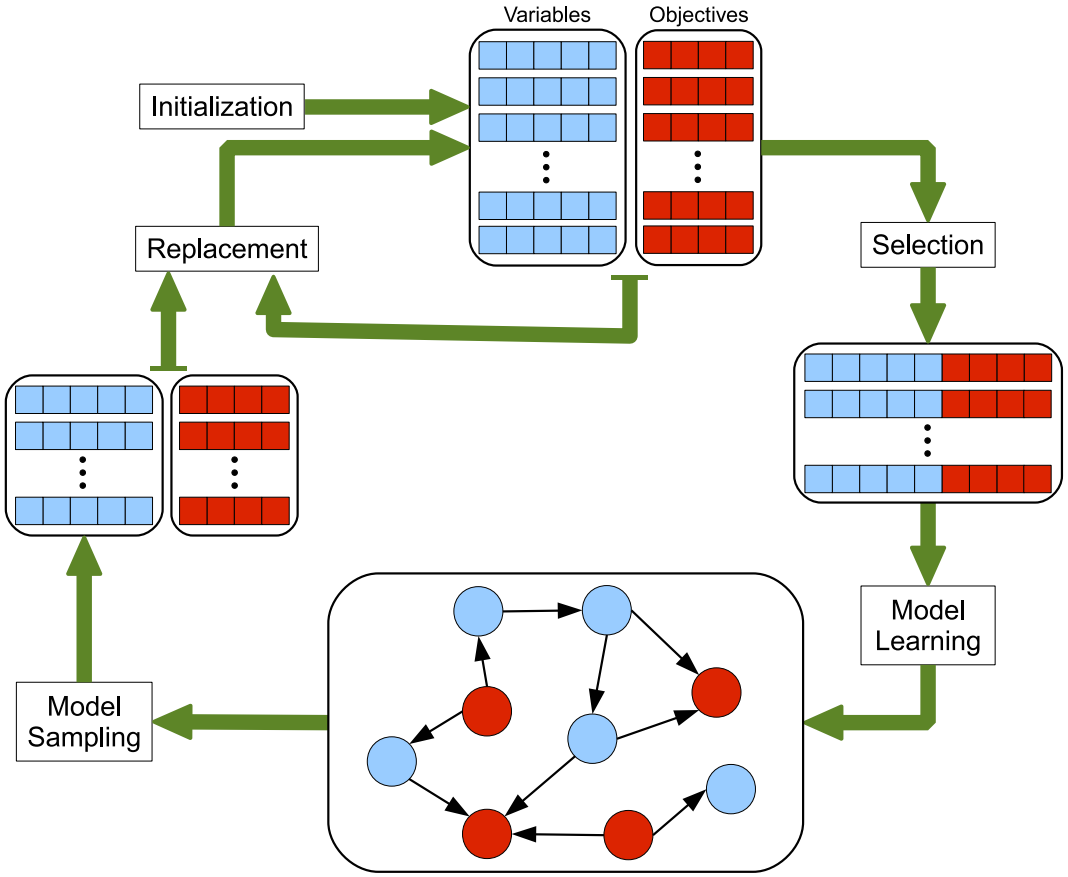



Figure 9.2: An overview of the proposed algorithm

## 9.3 JGBN-EDA for Continuous Multi-Objective Optimization

Figure 9.2 shows the outline of the proposed algorithm based on joint modeling of objectives and variables, called JGBN-EDA. After selecting a subset of solutions in the population, a dataset of joint variable-objective values is formed by appending the objective values of the selected solutions to their variable values. In each generation, a GBN is estimated from this dataset, encoding both objectives and variables, and it is used to sample new values for the variables when generating new solutions. The generated solutions are then evaluated to obtain their true objective values, though approximated objective values for the offspring solutions are also given by the estimated joint model. The solution ranking and selection method as well as the estimation of joint probabilistic model are explained in more detail in the following sections.

### 9.3.1 Solution Ranking and Selection

The extensively used *non-dominated sorting* algorithm [Deb et al., 2002a], based on Pareto dominance relation and crowding distances in the objective space, is adopted in JGBN-EDA for solution ranking. This algorithm first sorts the solutions into different Pareto sets and then within each set the solutions are ordered according to their crowding

distances, favoring solutions that are in less populated regions of the objective space to promote diversity of the approximated Pareto front. The crowding distance measures how close is (using Euclidean distance) each solution to its neighbors with regard to each of the objectives.

After obtaining a total ordering between all solutions with the non-dominated sorting algorithm, a subset of solutions is selected using truncation selection. For this purpose, the Pareto fronts are added to the set of selected solutions one-by-one according to their ranks and starting from the Pareto set with the best rank. When we reach a Pareto set which cannot be added completely anymore, due to the size limit determined by parameter $\tau \in (0, 1)$ of truncation selection, a subset of its solutions are selected according to their crowding distances. Here, we use a slightly modified version of the original non-dominated sorting algorithm for selecting this subset from the partially added Pareto set. The solutions are selected one-by-one for addition according to their crowding distances and after removing each individual, the crowding distance of the solutions remaining in the Pareto set are recomputed [Zhang et al., 2008].

Another sorting algorithm used in the proposed JGBN-EDA is the weighted sum approach. Given a weight vector $\boldsymbol{w} = (w_1, \ldots, w_m)$ showing the importance of each objective function, each solution $\boldsymbol{x}_i$ is assigned a value computed as:

$$\sum_{j=1}^{m} w_j \dot{f}_j(\boldsymbol{x}_i), \tag{9.1}$$

where $\dot{f}_j$ denotes a normalized value of the $j$th objective function considering the objective values in the population. The solutions are then ordered with respect to these weighted sums and the subset with the smallest values (assuming minimization) is selected. Here, we consider an uninformed version of this approach where all objectives are equally weighted.

### 9.3.2 Joint Model Learning and Sampling

We use the algorithm described in Section 6.3 to learn a GBN from the joint dataset of variable-objective values. In comparison to the case when only variable values are used for model estimation, the dimensionality of this dataset has increased while the number of samples is the same. As it was discussed in Chapter 4, this can affect the estimation of MGD parameters needed to compute the parameters of the conditional probabilities in GBN nodes. Therefore, regularization techniques are especially useful for the joint modeling scenario.

In the joint modeling used in JGBN-EDA, all of the values in the given joint dataset are standardized to have a zero mean and a variance of one, simplifying the GBN learning process and reducing the number of parameters encoded in each node [Schmidt et al., 2007]. The standardized joint dataset is then used for estimating the parameters of an MGD. Specifically, we use the shrinkage estimation method [Schäfer and Strimmer, 2005b], with a diagonal target matrix and no shrinkage on the diagonal entries, to obtain a regularized estimation of MGD's covariance matrix. Based on these estimations, the parameters of conditional probabilities in GBN nodes are computed.

The estimated GBN can be sampled like other BNs with PLS algorithm. However, since the model also consists of objective nodes, we consider two different strategies for

sampling. In the first approach, a set of expected or desired objective values are inserted into the model as evidence, when generating new values for the variables. This kind of information can be provided, for example, by a DM or a domain expert. Another possibility, when such information is not available, is to use the objective values of the selected solutions, favoring new solutions with objective values close to those of the selected solutions.

In the second approach, objective nodes are treated the same as variable nodes and new values are generated for both variables and objectives when sampling the estimated model. In this way, the PLS algorithm can take into account the approximated values for objectives, leading to a sampling which is more consistent with the probabilities encoded in the estimated model. Our experimental results have shown better exploration of the search space with this method, and therefore we use the second approach in JGBN-EDA. It should be noted that it is also possible to use a hybrid approach based on a combination of these two methods.

As it is common for the search in continuous spaces with EAs, some of the generated values may not be in the acceptable domain of their corresponding variables. To repair the invalid variable values (values not in the variable domain) of the new offspring solutions, they are reset to a random value in the acceptable domain of the variable. More precisely, to increase the possibility of appropriate value reseting, the invalid value is replaced with a random value generated between the variable's conditional mean (estimated with GBN) and the violated (upper or lower) domain-bound.

Since the number of objectives in MOPs are often of the same order as the number of variables (i.e. $n \simeq m$), according to the time complexity analysis provided in Section 5.3.2, the computational time requirement for estimating a GBN of the joint variable-objective vector is bounded by $O(kN(2n)^2+k(2n)^4)$, which adds to the coefficient of the complexity rather than its degree. In fact, in many MOPs the number of objectives is far smaller than the number of variables, and thus the additional overhead for integrating objectives does not considerably increase the computational time of model estimation or sampling.

## 9.4 Experiments

In this section the proposed algorithm is tested on a set of well-known benchmark MOPs to study its behavior and compare its performance with other EMO algorithms. Especially, to test the influence of learning a joint model in our proposed algorithm when including objectives, we compare it with a version that does not perform joint modeling and only considers variable values for learning the probabilistic model. In fact, this version of the algorithm is very similar to other BN-based EDAs found in the literature (e.g. EGNA [Larrañaga et al., 2000a]). We refer to this version as GBN-EDA to differentiate it from the algorithm based on joint modeling.

The proposed algorithm is also compared with NSGA-II [Deb et al., 2002a] which is considered as a standard reference in many of the studies on EMO algorithms [Zitzler et al., 2003; Coello Coello et al., 2007], and RM-MEDA [Zhang et al., 2008], a multi-objective EDA shown to outperform several EMO algorithms on different benchmark MOPs. NSGA-II is a multi-objective GA which uses simulated binary crossover [Deb and Agrawal, 1995] and polynomial mutation [Deb and Goyal, 1996] as its genetic operators

for search in continuous spaces, and non-dominated sorting algorithm for solution ranking and selection. RM-MEDA assumes a certain type of smoothness for Pareto optimal set and in each generation learns an $m-1$-dimensional piece-wise continuous manifold which is then used to generate new samples. The solutions are ranked using the non-dominated sorting algorithm.

All of the algorithms are implemented in Matlab®. In each generation, all of the algorithms select a subset of solution which are used for reproducing new offspring solutions, either using genetic operators or with estimating and sampling a probabilistic model. However, in contrast to the other algorithms which employ an elitist technique to decide which solutions in the population should be replaced by the newly generated offspring solutions, RM-MEDA just replaces the whole population with the offspring solutions. JGBN-EDA and GBN-EDA were tested with both non-dominated sorting and weighted sum approaches for solution ranking and selection. Since the results obtained with the weighted sum approach were superior to those obtained with the non-dominated sorting, which is also reported by others [Corne and Knowles, 2007; Garza-Fabre et al., 2009], therefore, unless otherwise stated, the results presented here are those obtained with weighted sum approach. Nevertheless, in the replacement step, the non-dominated sorting algorithm is used for elitist selection.

### 9.4.1 WFG Test Problems

Huband et al. [2006] reviewed many of the benchmark MOPs proposed in the literature like ZDT [Zitzler et al., 2000], DTLZ [Deb et al., 2002b] and OKA [Okabe et al., 2004a], and based on the analysis of these problems, they proposed a new set of MOPs called the walking fish group (WFG) problems. These MOPs have a diverse set of properties found in real-world problems and, therefore, can be a great challenge for any multi-objective optimization algorithm. Each objective function $f_j$ of an MOP in this benchmark is defined as

$$\min_{\boldsymbol{z}} \; f_j(\boldsymbol{z}) = D \cdot z_m + S_j \cdot h_j(z_1, \ldots, z_{m-1}), \tag{9.2}$$

where $D$ and $S_j$ are scaling factors and $h_j(\cdot)$ is a shape function, meaning that it will determine the shape of the Pareto optimal front of an MOP (e.g. concave, convex, etc.) together with the shape functions in the definition of other objective functions of that MOP. $\boldsymbol{z} = (z_1, \ldots, z_m)$ is an $m$-dimensional vector of parameters obtained by applying a number of transformation functions, like shifting, biasing or reduction, to the $n$-dimensional input solution $\boldsymbol{x} \in \mathbb{D}$.

The vector of parameters is composed of two parts: the first $m-1$ parameters, $z_1, \ldots, z_{m-1}$, are obtained from the first $k$ variables of the input solution, called position variables (they specify the location of the input solution mapping in the objective space). The last parameter ($z_m$) is obtained from the last $l$ variables of the input solution, called the distance variables (they determine the proximity of the input solution mapping in the objective space to the Pareto optimal front). In all WFG problems we have $n = k + l$, and to simplify the application of transformation functions on the input solution, $k$ is assumed to be a multiple of $m-1$ and $l$ should be an even number.

In all of the 9 MOPs of this benchmark, the number of both objectives and variables can be scaled, with dissimilar domains for the variables and different Pareto optimal

Table 9.1: Experimental settings for the WFG problems

| | | | |
|---|---|---|---|
| No. Objectives | 5 | 10 | 20 |
| No. Variables | 28 | 50 | 50 |
| Population Size | 600 | 1000 | 1500 |
| Max. No. Evaluations | $1.5 \times 10^5$ | $3 \times 10^5$ | $7.5 \times 10^5$ |
| No. Runs | 25 | 25 | 10 |

trade-off magnitudes for the objectives. Except the first three MOPs, the rest of WFG problems have a concave Pareto optimal front. WFG1 has a mixed convex-concave front, WFG2 has a disconnected convex front, and WFG3 has a degenerated one-dimensional linear front. For most of these MOPs, the optimal solution of objectives is shifted away from zero to neutralize any bias of the optimization algorithms towards smaller values of the variables. Moreover, in many of the WFG problems, the objective functions are inseparable, requiring the optimization algorithm to consider the relationships between variables. Other properties like multi-modality and deception are also present in the objective functions of these MOPs.

### 9.4.2 Experimental Design

The quality of the Pareto fronts approximated by each algorithm is evaluated with the inverted generational distance (IGD) [Coello Coello and Cortés, 2005] indicator. This indicator accounts for both the diversity of the approximated front as well as its convergence to the Pareto optimal front. Given a set of points $F^*$, representing a well-distributed sampling of the Pareto optimal front of an MOP, the IGD value for an approximated Pareto front $F$ is computed as:

$$IGD_{F^*}(F) = \frac{\sum_{s \in F^*} \min_{\forall s' \in F} \mathrm{d}(s, s')}{|F^*|} \tag{9.3}$$

where $\mathrm{d}(\cdot, \cdot)$ gives the Euclidean distance between two points. A smaller value for this indicator means a better approximation. The statistical significance in the differences between the results is checked with the Kruskal-Wallis test [Kruskal and Wallis, 1952]. Kruskal-Wallis test performs a non-parametric one-way analysis of variance (ANOVA) to accept or reject the null hypothesis that independent samples of two or more groups come from distributions with equal medians, and returns the $p$-value for that test. The test significance level is set to 0.05 in our comparisons.

We have tested three different numbers of objectives with respectively different number of variables for the WFG problems to keep the computational costs of the experiments in an affordable level. The details of the experimental setup, which are equally set for all of the tested EMO algorithms, are given in Table 9.1. To have an idea of the Pareto fronts approximated by these algorithms, in Figure 9.3 we have shown the final Pareto fronts obtained by the algorithms for some of the tested MOPs (WFG2, WFG4, WFG6 and WFG7) with 3 objectives, 20 variables and a population size of 200. The represented fronts for each algorithm are selected between 25 independent runs using the

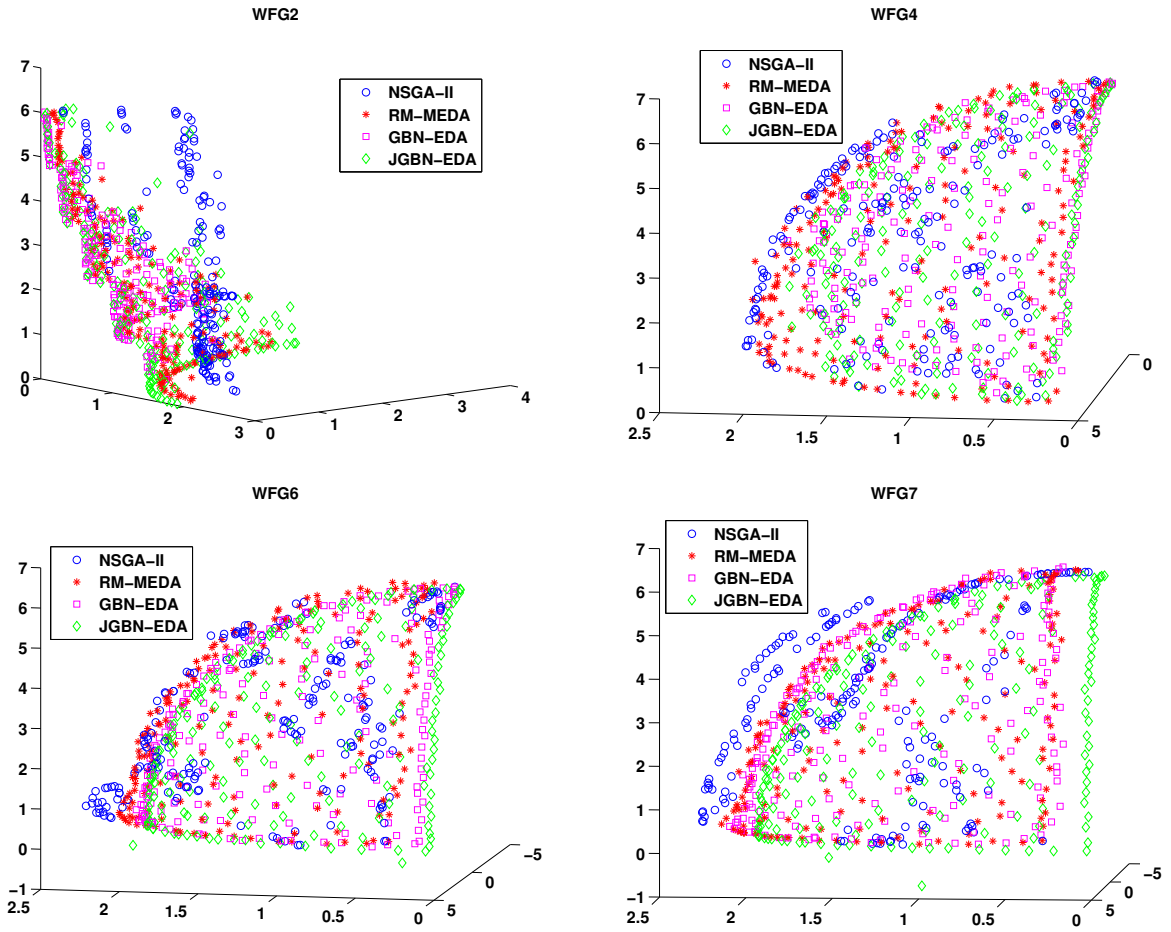



Figure 9.3: Typical Pareto fronts obtained by different algorithms for WFG problems with 3 objectives.

coverage indicator. This indicator computes the percentage of solutions in a Pareto set $B$ dominated by the solutions in another Pareto set $A$

$$A \text{ coverage of } B = \frac{\left|\{\boldsymbol{x} \in B \mid \exists \boldsymbol{y} \in A, \boldsymbol{y} \prec \boldsymbol{x}\}\right|}{|B|}.$$

### 9.4.3 Results

Figure 9.4 shows the results obtained by each of the algorithms for WFG problems with 5 objectives. As it can be seen in the figure, the incorporation of objectives in the modeling of JGBN-EDA enables this algorithm to obtain a (significantly) better performance on most of the problems, according to the IGD indicator. A direct comparison of GBN-EDA and JGBN-EDA shows the effectiveness of joint modeling where except for WFG5 and WFG9 problems, the latter algorithm can obtain significantly better results on the tested MOPs ($p = 0.05$).

WFG5 is composed of separable and deceptive functions while WFG9 consists of a non-separable, biased, multi-modal and deceptive functions. According to the presented

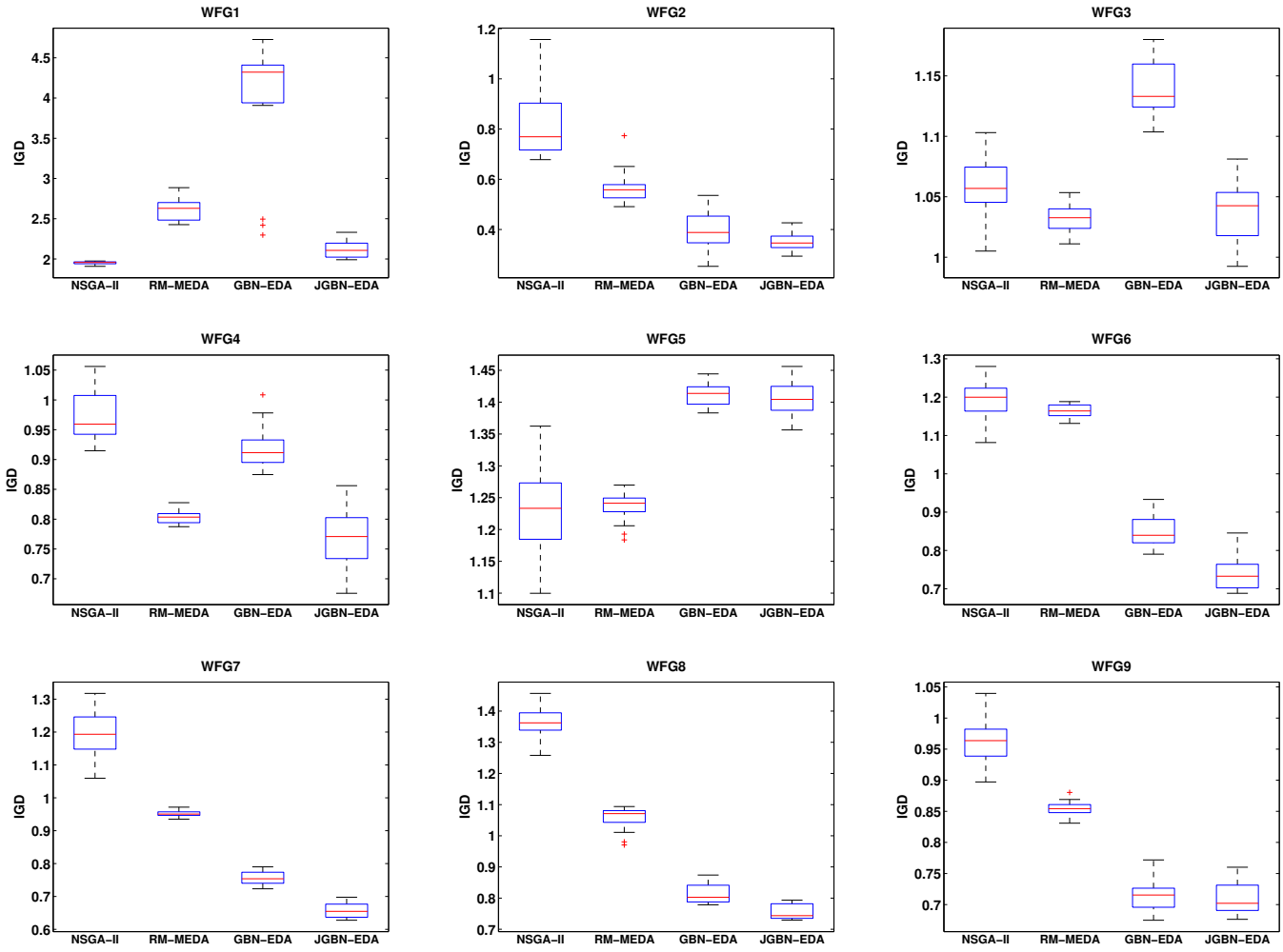



Figure 9.4: Comparison of the performance of different algorithms on WFG problems with 5 objectives.

results, for deceptive functions the information provided by objectives does not have a great impact on the joint probabilistic model estimated in JGBN-EDA for generating better solutions. Also, the algorithm is not able to properly utilize the separability of variables in WFG5 to obtain better fronts. However, the results obtained for WFG9 shows that non-separability and multi-modality in the functions of this problem are completely addressed with the probabilistic modeling used in JGBN-EDA and GBN-EDA, making them better performers in comparison to the other two competitor algorithms for this problem.

As the number of objectives grows to 10 and 20 (Figures 9.5 and 9.6), the performance of the proposed algorithm, measured by IGD indicator, deteriorates in comparison to other algorithms. The other two competitor algorithms also show a diverse behavior on different number of objectives. While the fronts approximated by NSGA-II on 10-objective MOPs are not comparable to those of other algorithms, this algorithm is able to find significantly better results for most of the problems with 20 objectives. RM-MEDA is showing an apposite behavior, obtaining better results on the 10-objective case. For

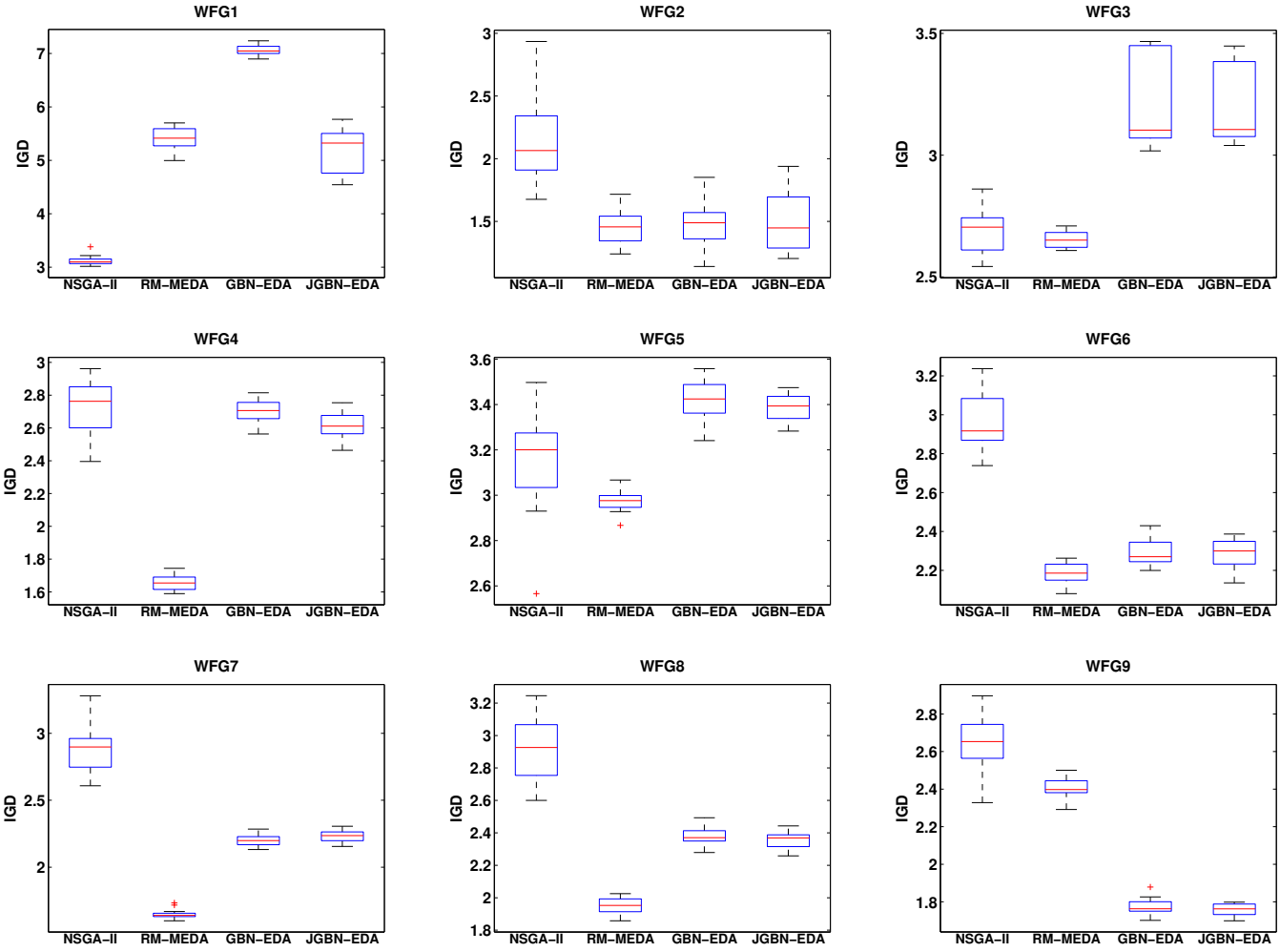



Figure 9.5: Comparison of the performance of different algorithms on WFG problems with 10 objectives.

these high number of objectives the performance of the proposed JGBN-EDA seems to be less varying when compared with other algorithm on both 10- and 20-objective WFG problems.

The recombination operators employed in NSGA-II allow obtaining better approximations for the mixed concave-convex front of WFG1 with some flat regions in the Pareto set. The inclusion of objective values in the modeling of JGBN-EDA has a major influence in improving the performance on this MOP when compared to GBN-EDA. On the other hand, the disconnected front of WFG2 causes a diverse performance of NSGA-II, while the EDAs are almost performing similar when the number of objectives is 10 and 20. The significantly better performance of JGBN-EDA on WFG9 is also repeated for 10 and 20 objectives.

It seems that when the number of objectives increase to a high number (e.g. 20), the probabilistic modeling in EDAs is overfitted for small populations and can not help these algorithms to make progress in the search space. In fact for such a large number of objectives small population sizes may not be sufficient to represent a good approximation

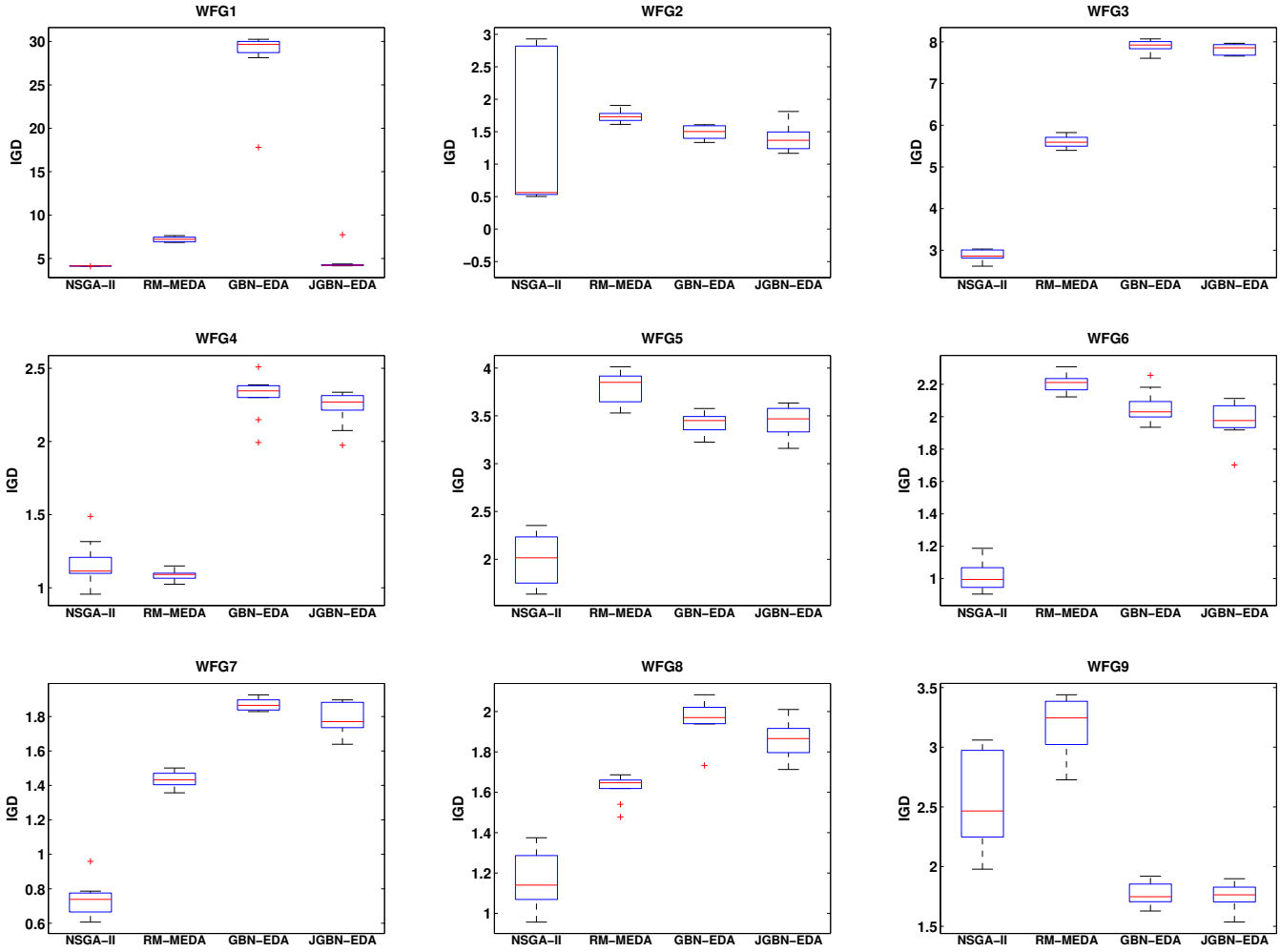



Figure 9.6: Comparison of the performance of different algorithms on WFG problems with 20 objectives.

of the Pareto front. This issue should also be considered for representing the Pareto optimal front with a sample set which is used for computing the IGD quality indicator. Although in this study we have used sample sets with several hundred thousands of points for this purpose, some regions of the Pareto optimal front may still not be covered properly.

## 9.5 Conclusions

A joint modeling of objectives and variables was proposed to be used for multi-objective optimization in this chapter. The proposed EDA, called JGBN-EDA, learns a GBN to encode the relationships between variables and objectives, from a joint variable-objective dataset formed by appending the objective values to the variables values of the solutions. The estimated probabilistic model encodes both variable and objective nodes and thus in the sampling process new values for variables can be generated using the objective values when such dependencies are encountered in the model.

The performance of the algorithm was tested on a number of test MOPs having different properties found in real-world problems. The results of comparing two versions of the algorithm, one utilizing the objectives information and the other not, showed that the incorporation of objectives in the modeling can help the algorithm to obtain better Pareto fronts on some of the problems. The algorithm was also compared with two other algorithms, NSGA-II as a standard EMO algorithm and RM-MEDA which is a competitive multi-objective EDA, and the obtained results indicated the promising performance of the proposed approach to probabilistic modeling in EDAs.

The algorithm was not able to detect the correct search bias for some MOPs with deceptive objective functions using the proposed joint modeling. Moreover, for some of the problems the algorithm was not able to obtain competitive fronts when there are a high number of objectives in the MOP. The effect of proper population sizing on the performance of the algorithm specially for many objectives problems should also be studied in more detail. The information provided by dependencies between the objectives can be further investigated to help revealing their relationships in problems with many objectives. Nevertheless, the proposed algorithm can be seen as an alternative for using probabilistic modeling in multi-objective optimization.

# Chapter 10

# Multi-Dimensional Bayesian Network Modeling for Evolutionary Multi-Objective Optimization

## 10.1 Introduction

In the previous chapter, we studied joint probabilistic modeling of variables and objectives for multi-objective optimization. This type of modeling offered a new way for analyzing the relationships between variables and objectives, and was shown to be advantageous for solution search compared with learning a model of only variables. In this chapter, we extend this study using a specific probabilistic modeling employed in the area of multi-dimensional classification. In this type of problems, each instance or data point is assigned a vector of class-values, and the goal of model learning is to predict the class-value vectors of new instances. On the other hand, in multi-objective EDAs, each solution is assigned a vector of objective values, and the goal of model learning is to generate new solutions with better objective values from the probabilistic model. Clearly, there are similarities between the two problems which motivates the use of similar probabilistic modeling in both.

We adapt multi-dimensional BN classifiers [de Waal and van der Gaag, 2007; Bielza et al., 2011b] for joint modeling of variables and objectives. Using this type of probabilistic model in joint modeling, three distinct types of relationships can be captured: (i) the interactions between variables, as in other EDAs, (ii) the relationships between variables and objectives, and (iii) the relationships between objectives. The clear distinction between these three types of relationships, of which the last two are captured because of joint modeling, allows a better analysis and interpretation of the estimated model structures, and since it is closer to the semantics of MOPs, it provides the DM with an approximation of the MOP structure, i.e. the relationships among variables and objectives of the MOP. Moreover, as it will be shown later, with this type of probabilistic model, EDA is able to find better Pareto fronts for many of the tested MOPs. The proposed method and related discussions are also appeared in [Karshenas et al., 2012b].

As reviewed in Chapter 2, BN classifiers have been previously used as probabilistic models in EDAs for single-objective optimization in EBCOA [Miquélez et al., 2004, 2006]. However, there are several key differences between EBCOA and the algorithm presented

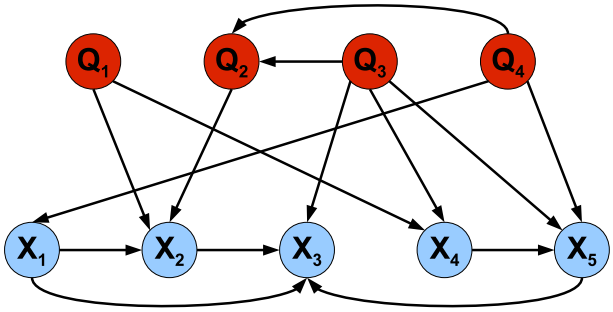



Figure 10.1: An example of a multi-dimensional Bayesian network structure.

here. First, the presence of multiple objectives in an MOP increases the information about the quality of solutions (possibly contradictory) that should be addressed during modeling. Second, in continuous domains, instead of classifying the solutions into disjoint classes which may blur the differences in the quality of the solutions, we directly use the continuous objective values in joint model learning. Third, in contrast to a fixed dependency between the objectives and variables, the algorithm presented here dynamically learns the relationships between the objectives and variables. In this way, the model can select a subset of variables that has more influence on each objective.

## 10.2 Multi-Dimensional Bayesian Network Classifiers

When the classification problem involves more than one class variable, each instance of the dataset is assigned a vector of class-values $\boldsymbol{c} = (c_1, \ldots, c_m)$. Then, to determine the class-value vector of an unlabeled instance given its feature values $\boldsymbol{x} = (x_1, \ldots, x_n)$, a multi-dimensional BN (MBN) can be estimated from a training dataset and used to compute the class-value vector with the highest posterior probability for this instance:

$$\boldsymbol{c}^* = \arg \max_{\boldsymbol{c}} P(\boldsymbol{c} \mid \boldsymbol{x}). \tag{10.1}$$

Figure 10.1 shows an example of an MBN structure. In this type of model, the nodes are organized in two separate layers: the top layer comprises class variables and the bottom layer contains feature variables. The set of arcs in the structure is partitioned into three subsets, resulting in the following subgraphs:

- the class subgraph, containing the class nodes and the interactions between them,
- the feature subgraph, comprising the feature variables and their relations, and
- the bridge subgraph, depicting the *top-down* relationships between class and feature nodes.

Depending on the order of relationships and the structure learnt in the class and bridge subgraphs, different types of MBN classifiers can be considered: without any edges like NB, directed trees like TAN, polytrees or even completely unrestricted.

This probabilistic model can answer several types of queries: the vector of class-values of a given data point, the most probable feature values for a given vector of class-values,

and the most probable values for a subset of features or classes given the value of the others. Considering the similarity between multi-dimensional classification and multi-objective optimization, the respective questions will be: what are the estimated objective values of a given solution, what is the most probable values for the variables of the solution which results in a specific value-setting for the objectives, and, having found the proper values of some objectives or variables, what will be the most probable values of the others.

## 10.3 MBN-EDA: An EDA Based on MBN Estimation

In this section we propose a multi-objective EDA which uses MBNs for joint modeling of variables and objectives, called MBN-EDA. The variables are modeled as feature nodes and objectives as continuous-valued class nodes (regression response nodes). The feature subgraph of MBN encodes the relationships between variables like the models learnt by other BN-based EDAs [Schwarz and Očenášek, 2001; Khan et al., 2002; Laumanns and Očenášek, 2002; Katsumata and Terano, 2003; Pelikan et al., 2005; Ahn and Ramakrishna, 2007]. However, the bridge and class subgraphs, encode new types of relationships as the result of joint modeling of variables and objectives. The bridge subgraph shows the interactions between each objective and the variables (selects a subset of relevant variables for each objective), and the class subgraph represents the relationships directly between objectives. The joint probability distribution encoded in the probabilistic model of MBN-EDA can be represented as

$$\rho\left(x_1,\ldots,x_n,q_1,\ldots,q_m\right)=\prod_{i=1}^{n}\rho(x_i|\boldsymbol{pa}_i)\cdot\prod_{j=1}^{m}\rho(q_j|\boldsymbol{pa}'_j), \tag{10.2}$$

where $\boldsymbol{Pa}_i\subseteq\left\{\{\boldsymbol{X}\setminus X_i\}\cup\boldsymbol{Q}\right\}$ and $\boldsymbol{Pa}'_j\subseteq\{\boldsymbol{Q}\setminus Q_j\}$ are the parents of each variable and objective, respectively, according to the MBN structure, and $\boldsymbol{pa}_i$ and $\boldsymbol{pa}'_j$ represent one of their possible value-settings. $\boldsymbol{q}=(q_1,\ldots,q_m)$ denotes a possible value-setting for the objective variables $\boldsymbol{Q}=(Q_1,\ldots,Q_m)$ corresponding to solution $\boldsymbol{x}=(x_1,\ldots,x_n)$.

The general framework of MBN-EDA is the same as the one shown for joint model learning in Figure 9.2 of the previous section. However, the solution ranking methods and the probabilistic model used in this algorithm are different and are discussed in the following sections.

### 10.3.1 Solution Ranking and Selection

In contrast to single objective optimization, where the objective values can be used directly to rank solutions, the existence of multiple objectives in MOPs necessitates the application of an intermediate function of the form $g:\mathbb{F}\subseteq\mathbb{R}^m\mapsto\mathbb{F}'\subseteq\mathbb{R}$, so that its output can be used to rank the solutions. In the previous chapter, the commonly used non-dominated sorting algorithm was employed to sort the solutions for selection. However, it has been shown that the effectiveness of this ranking method decreases as the dimensions of the objective space increase [Hughes, 2005; Ishibuchi et al., 2008; Aguirre

and Tanaka, 2010]. Finding efficient ranking methods for many-objective optimization is the topic of ongoing research, and several methods have been proposed so far in the literature. In this study, we adopt some of the methods proposed for this purpose, which were shown to result in good performance for evolutionary many-objective optimization [Corne and Knowles, 2007; Garza-Fabre et al., 2009, 2010a,b].

Let $\boldsymbol{x}$ and $\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{D}$, be two solutions of a population of solutions $\mathcal{D}$ to a given MOP. Then, beside the weighted sum approach defined in Section 9.3.1, denoted as $g_{\text{WS}}$, the following three ranking methods are employed in MBN-EDA:

- Distance to the best objective values $\boldsymbol{f}^* = (f_1^*, \ldots, f_m^*)$ for each solution, using some distance measure $\mathrm{d}(\cdot, \cdot)$ in the objective space (e.g. Euclidean distance):

$$g_{\text{DB}}(\boldsymbol{x}) = \mathrm{d}\big(\boldsymbol{f}^*, \dot{\boldsymbol{f}}(\boldsymbol{x})\big). \tag{10.3}$$

  When the best objective values are not known beforehand (which is usually the case), the best objective values achieved so far (considering each objective separately) in the current population can be used, i.e. the best value $f_j^*$ for the $j$th objective function is

$$f_j^* = \min_{\boldsymbol{x} \in \mathcal{D}} \ \dot{f}_j(\boldsymbol{x}).$$

- Global detriment or the total gain lost by each solution against other solutions of the population:

$$g_{\text{GD}}(\boldsymbol{x}) = \sum_{\forall \boldsymbol{y} \in \mathcal{D}, \boldsymbol{y} \neq \boldsymbol{x}} \text{gain}(\boldsymbol{y}, \boldsymbol{x}), \tag{10.4}$$

  where the function $\text{gain}(\cdot, \cdot)$ computes the gain of a solution over another in the objective space (assuming minimization):

$$\text{gain}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \max\{0, \dot{f}_j(\boldsymbol{y}) - \dot{f}_j(\boldsymbol{x})\}. \tag{10.5}$$

- Profit of the gain obtained from each solution against other solutions of the population:

$$g_{\text{PG}}(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \mathcal{D}, \boldsymbol{y} \neq \boldsymbol{x}} \ \text{gain}(\boldsymbol{x}, \boldsymbol{y}) - \max_{\boldsymbol{y} \in \mathcal{D}, \boldsymbol{y} \neq \boldsymbol{x}} \ \text{gain}(\boldsymbol{y}, \boldsymbol{x}). \tag{10.6}$$

When any of the above $g_{\text{XX}}$ functions is applied to the solutions in a population $\mathcal{D}$, a single value is assigned to each of the solutions served for their sorting. Given a sorted list of solutions, any selection mechanism can be used for selecting a subset. Similar to the previous chapter, a truncation selection mechanism is used in MBN-EDA.

### 10.3.2 MBN Learning and Sampling

Here, we explain probabilistic modeling in MBN-EDA for continuous multi-objective optimization, assuming a Gaussian distribution for the joint vector of variables and objectives. In continuous domains, MBN is implemented with a GBN. Instead of learning the structure of each of the MBN subgraphs separately, we use a learning procedure very similar to the one explained in Section 9.3.2 of the previous chapter. Since the structure

of the bridge subgraph is constrained to only include top-down interactions between variable and objective nodes, the greedy search is performed in a restricted DAG space, and the corresponding sub-matrix of the covariance matrix is explicitly set to zero. The BIC score of an MBN structure based on log-likelihood and computed from a joint dataset of variable-objective values $\Big\{\big(\boldsymbol{x}_1, \boldsymbol{f}(\boldsymbol{x}_1)\big), \ldots, \big(\boldsymbol{x}_N, \boldsymbol{f}(\boldsymbol{x}_N)\big)\Big\}$ is

$$\sum_{k=1}^{N}\Big(\sum_{i=1}^{n}\log\big(p(\boldsymbol{x}_{k<X_i>} \mid \boldsymbol{x}_{k<\boldsymbol{Pa}_i>})\big) + \sum_{j=1}^{m}\log\big(p(f_j(\boldsymbol{x}_k) \mid \boldsymbol{x}_{k<\boldsymbol{Pa}'_j>})\big)\Big) - \frac{1}{2}\log(N)\Big(\sum_{i=1}^{n}|\boldsymbol{Pa}_i| + \sum_{j=1}^{m}|\boldsymbol{Pa}'_j| + 2(n+m)\Big), \quad (10.7)$$

where $\boldsymbol{x}_{k<\boldsymbol{Pa}_i>}$ and $\boldsymbol{x}_{k<\boldsymbol{Pa}'_j>}$ are the value-settings for the parents of respectively the $i$th variable and $j$th objective nodes, according to the MBN structure, in the $k$th joint variable-objective individual of the dataset. $|\boldsymbol{Pa}_i| \leq n+m-1$ and $|\boldsymbol{Pa}'_j| \leq m-1$ show the number of parents for the $i$th variable and $j$th objective, according to the MBN structure.

As before, the PLS algorithm is used to sample the estimated MBN for generating new solutions. Due to the restrictions imposed on the bridge subgraph in the learning process, all objective nodes appear before variable nodes in the topological ordering obtained for an MBN. This decreases the number of inconsistent values generated for the variables when a set of objective values are inserted as evidence during model sampling [Koller and Friedman, 2009]. The second approach to joint model sampling, explained in Section 9.3.2, is employed in MBN-EDA to take into account the approximated characteristics of the objective values (collected from the selected solutions) encoded in the model.

## 10.4 Experiments on WFG Test Problems

Following the experimental setup of the previous chapter, in this section we examine the performance of MBN-EDA and its joint modeling for continuous multi-objective optimization of WFG problems. Figure 10.2 shows the final Pareto fronts approximated by NSGA-II, RM-MEDA and MBN-EDA for some of the WFG problems (WFG2, WFG3, WFG4, WFG6, WFG7) with 3 objectives, 20 variables and a population size of 200. These fronts are selected, according to the coverage indicator, between 25 independent runs of each algorithm when using non-dominated sorting for solution ranking and selection.

To compare the joint probabilistic modeling of variables and objectives performed in MBN-EDA with the one done in JGBN-EDA of the previous chapter, we have compared their optimization performance on WFG problems with an increasing number of objectives. On some of these problems, the quality of the final Pareto fronts obtained by the two algorithms, evaluated with IGD indicator, are very similar. However, for some of the WFG problems, there are considerable differences in the results obtained by these algorithms. For example, Figures 10.3 and 10.4 show the IGD indicator values of the final Pareto fronts obtained by these two algorithms, when using the non-dominated sorting method for solution ranking and selection, in 25 independent runs for respectively

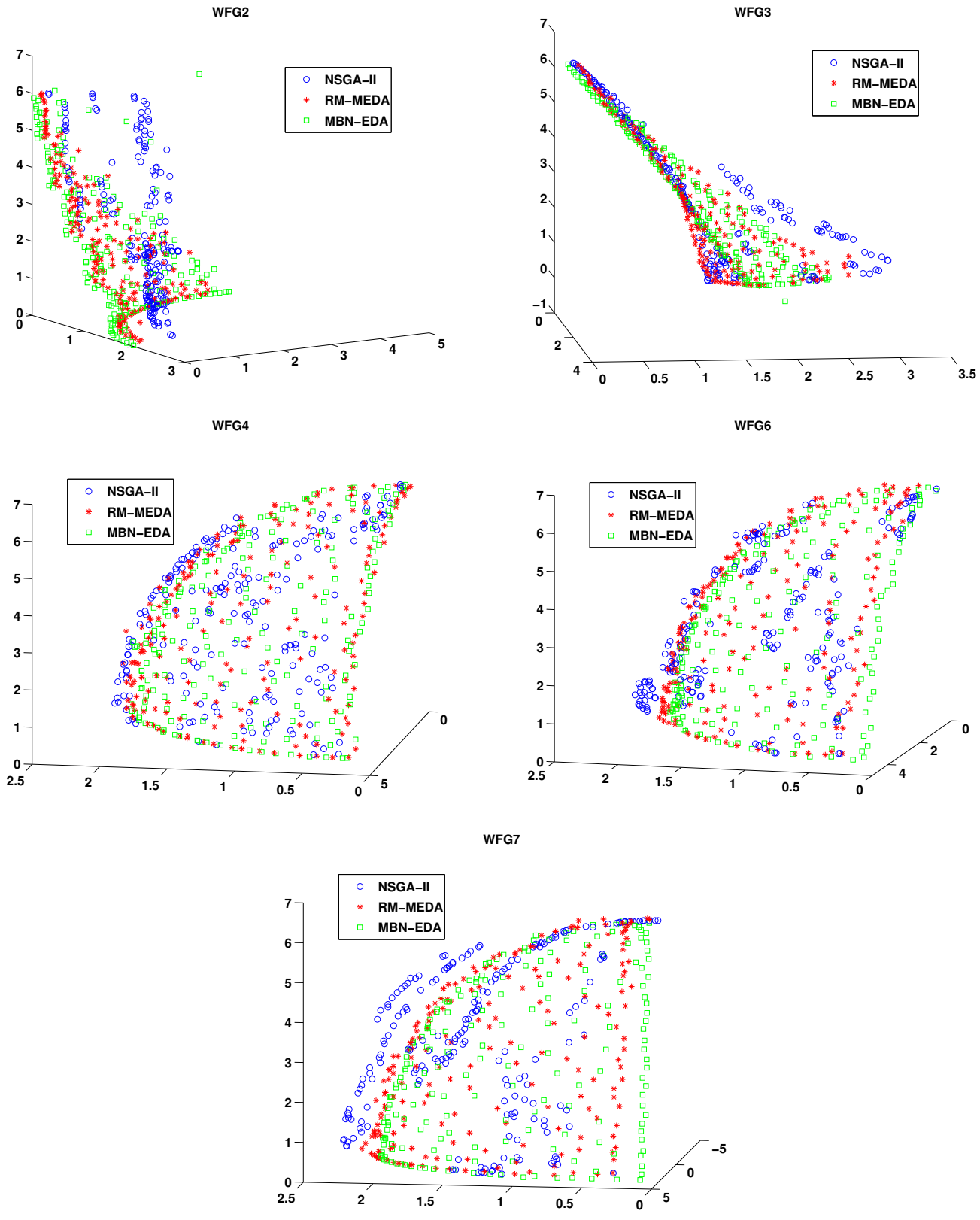
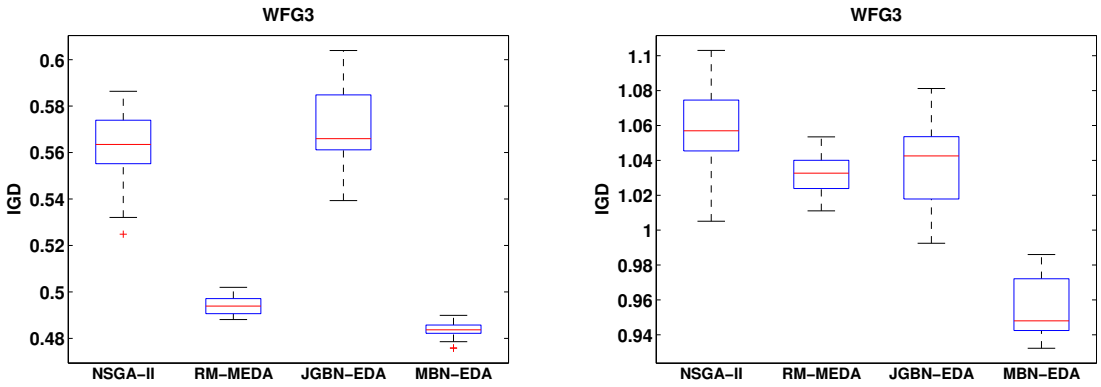


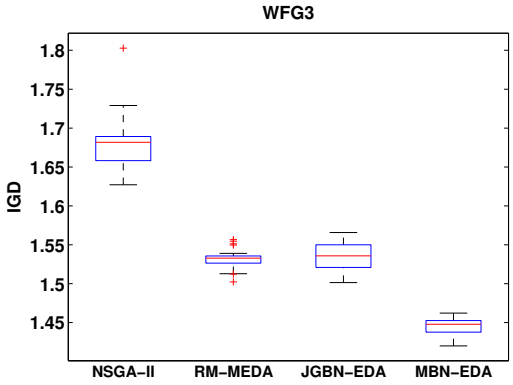

Figure 10.2: The Pareto fronts obtained by different algorithm for WFG problems with 3 objectives.



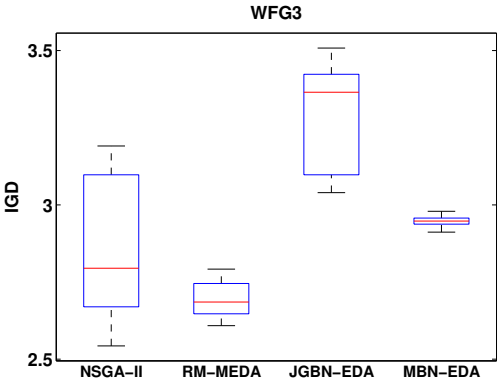

(a) 3 objectives




(b) 5 objectives




(c) 7 objectives



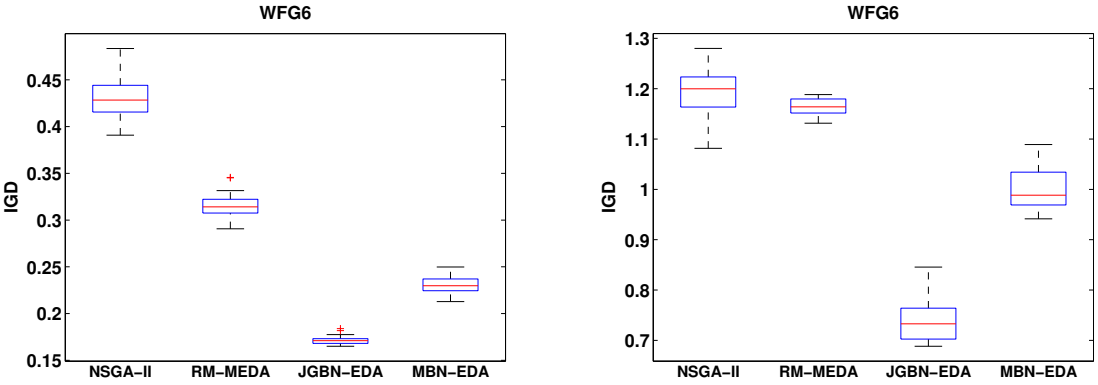

(d) 10 objectives

Figure 10.3: IGD values of the approximated Pareto fronts for WFG3 problem with different number of objectives.

WFG3 and WFG6 problems with different number of objectives. The figures also include NSGA-II and RM-MEDA for comparison. It can be seen that the fronts approximated by MBN-EDA for WFG3 are significantly better, whereas JGBN-EDA is able to find considerably better fronts for WFG6 with respect to the IGD indicator. This also suggests that certain types of structure restrictions in joint probabilistic modeling can improve solution search for problems with specific properties (e.g. degeneration in WFG3).

### Implementation Details and Experimental Design

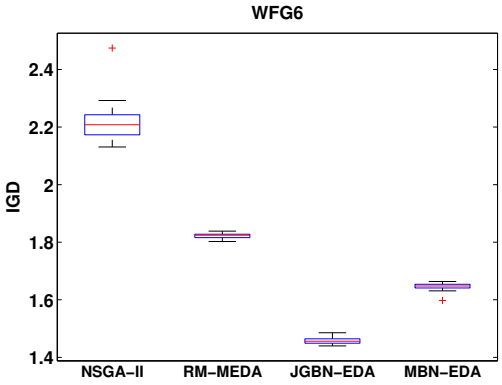
In the following experiments, the performance of NSGA-II, RM-MEDA and MBN-EDA are compared on WFG problems. The number of objectives considered in the experiments are 3, 5, 7, 10, 15 and 20, whereas the number of variables is set to 16 (with some exceptions). In this way, we will be able to investigate the performance of the algorithms against an increasing number of objectives while the size of solution space is unchanged. The four ranking methods described in Section 10.3.1 are implemented within an individual selector engine which is plugged into each of these algorithms. Therefore, since NSGA-II will not use the non-dominated sorting algorithm for solution ranking anymore (which is the cause for its name), we will simply call it multi-objective EA (MOEA) henceforth.



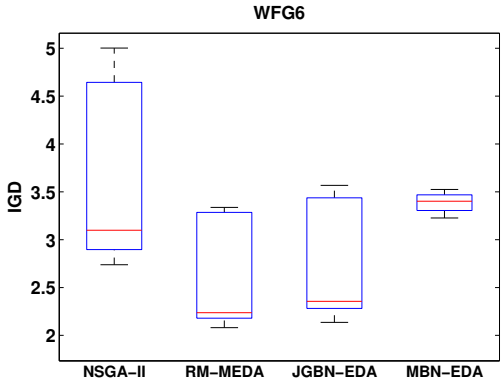

(a) 3 objectives




(b) 5 objectives




(c) 7 objectives




(d) 10 objectives

Figure 10.4: IGD values of the approximated Pareto fronts for WFG6 problem with different number of objectives.

Each algorithm with each ranking method is separately applied to each WFG problem with different numbers of objectives. Therefore, there will be $3 \times 4 \times 9 \times 6$ possible combinations for the experiments. All of the algorithms stop after reaching a maximum number of generations, which is set to 300. The population size is equal for all algorithms and is gradually incremented as the number of objectives increases according to Table 10.1. In each generation, 50% of the solutions in the population are selected for reproduction (i.e. $\tau = 0.5$).

The additive epsilon indicator [Zitzler et al., 2003, 2008] is used to measure the quality of the results obtained by each of the algorithms because of its tractable computational complexity for many-objective problems. This indicator is based on the notion of epsilon efficiency [Helbig and Pateva, 1994], and the corresponding relation of *epsilon dominance*

Table 10.1: The population size used for different number of objectives and variables.

| No. Objectives | 3 | 5 | 7 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| No. Variables | 16 | 16 | 16 | 15 | 16 | 21 |
| Population Size | 50 | 100 | 150 | 200 | 250 | 300 |

that is defined as

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{D},\ \boldsymbol{x} \preceq_{\epsilon+} \boldsymbol{y} \quad \Longleftrightarrow \quad \forall f_i \in \mathcal{F} \quad f_i(\boldsymbol{x}) \leq \epsilon + f_i(\boldsymbol{y}). \tag{10.8}$$

The additive epsilon indicator between two Pareto set approximations $A$ and $B$ is defined as the smallest epsilon value that allows all the solutions in $B$ to be '$\epsilon$+'-dominated by at least one solution in $A$:

$$I_{\epsilon+}(A, B) = \max_{\epsilon \in \mathbb{R}^+} \{A \preceq_{\epsilon+} B\}, \tag{10.9}$$

where

$$A \preceq_{\epsilon+} B \quad \Longleftrightarrow \quad \forall \boldsymbol{y} \in B, \exists \boldsymbol{x} \in A \mid \boldsymbol{x} \preceq_{\epsilon+} \boldsymbol{y}.$$

According to this definition, the additive epsilon indicator for a Pareto set approximation $A$ is obtained using a reference set $R$

$$I_{\epsilon+}(A) = I_{\epsilon+}(A, R). \tag{10.10}$$

This definition implies that smaller values of the epsilon indicator are better. A good choice for the reference set $R$ is an approximation of the Pareto optimal set. However, the size of a good approximation of the Pareto optimal set should increase exponentially with the number of objectives in the MOP to offer a good coverage of the Pareto optimal front. Therefore, this choice of reference set is impractical for many-objective problems. The reference set considered here is composed of the endpoint solutions, obtained by setting one of the objectives to its minimum value and the others to their maximum values, plus the solution representing an approximate compromise between the values of all objectives (e.g., the mean value in the objectives range). The size of this reference set grows only linearly with the number of objectives, and the inclusion of endpoints favors those Pareto set approximations that result in a more scattered Pareto front.

### Results

Figures 10.5–10.7 show the epsilon indicator value obtained for the Pareto sets approximated by each of the algorithms when using different ranking methods, averaged over 20 independent runs. The statistical analysis of the results on each of the MOPs with different numbers of objectives is shown in Table 10.2. The non-parametric Friedman test is used to check for the statistical differences of the algorithms performance [Derrac et al., 2011]. When the null hypothesis that all the algorithms have an equal average rank is rejected for a specific problem configuration with a p-value less than 0.05, the entry related to the algorithm with the best Friedman rank is shown in bold. The numbers in parentheses show the results of pairwise comparisons using Bergmann-Hommel's post-hoc test with a significance level of $\alpha = 0.05$. The first number shows how many algorithms are significantly worse than the algorithm listed in this column, and the second number shows how many algorithms are significantly better.

The objectives in WFG1 are unimodal and biased for specific regions of their input. For this problem, MBN-EDA is able to obtain significantly better Pareto set approximations than the other two algorithms. The performance of the algorithm is very similar when using the different ranking methods tested in these experiments (Figure 10.5, left

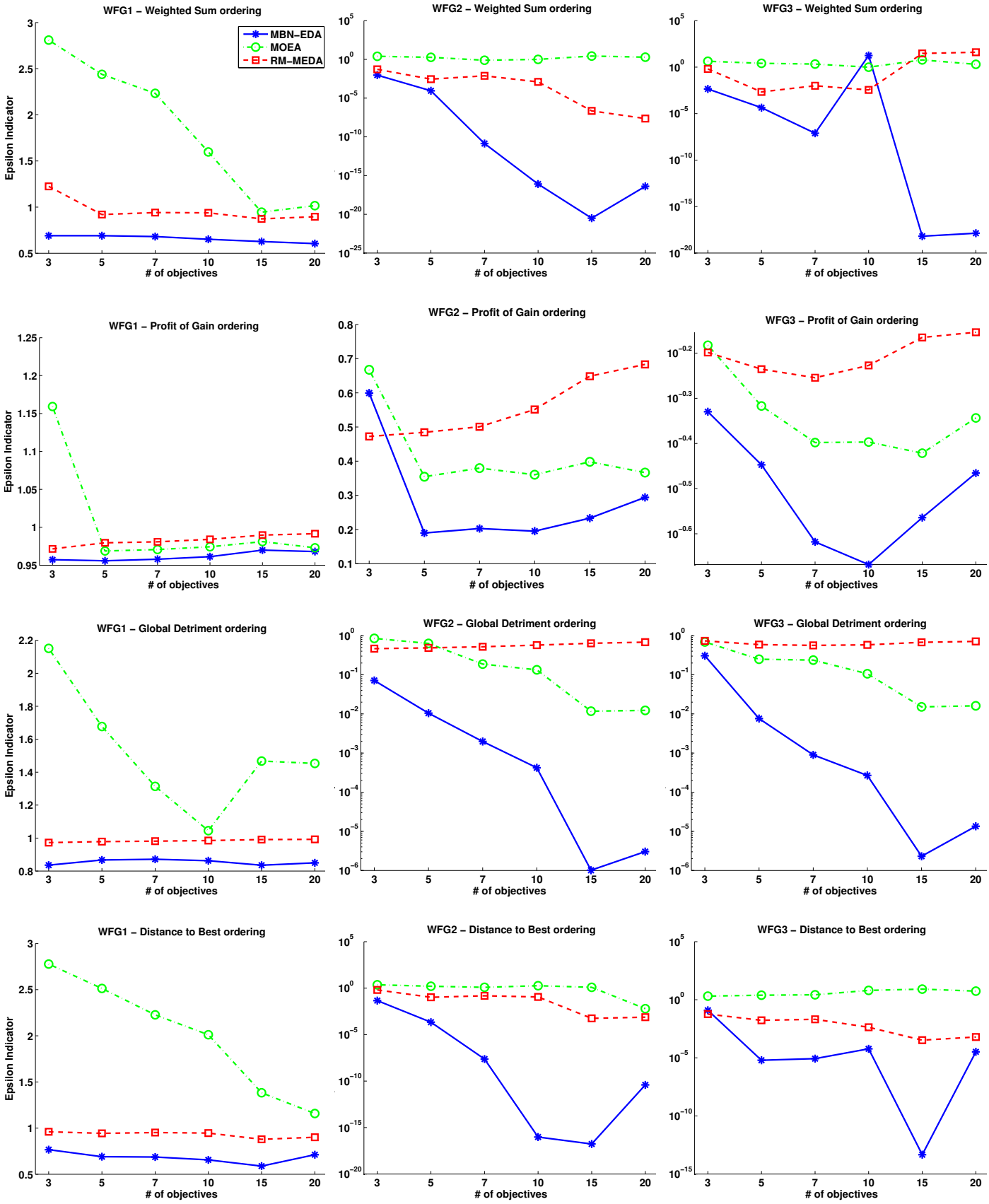



Figure 10.5: The average epsilon indicator values for WFG1 (left column), WFG2 (middle column) and WFG3 (right column) problems.

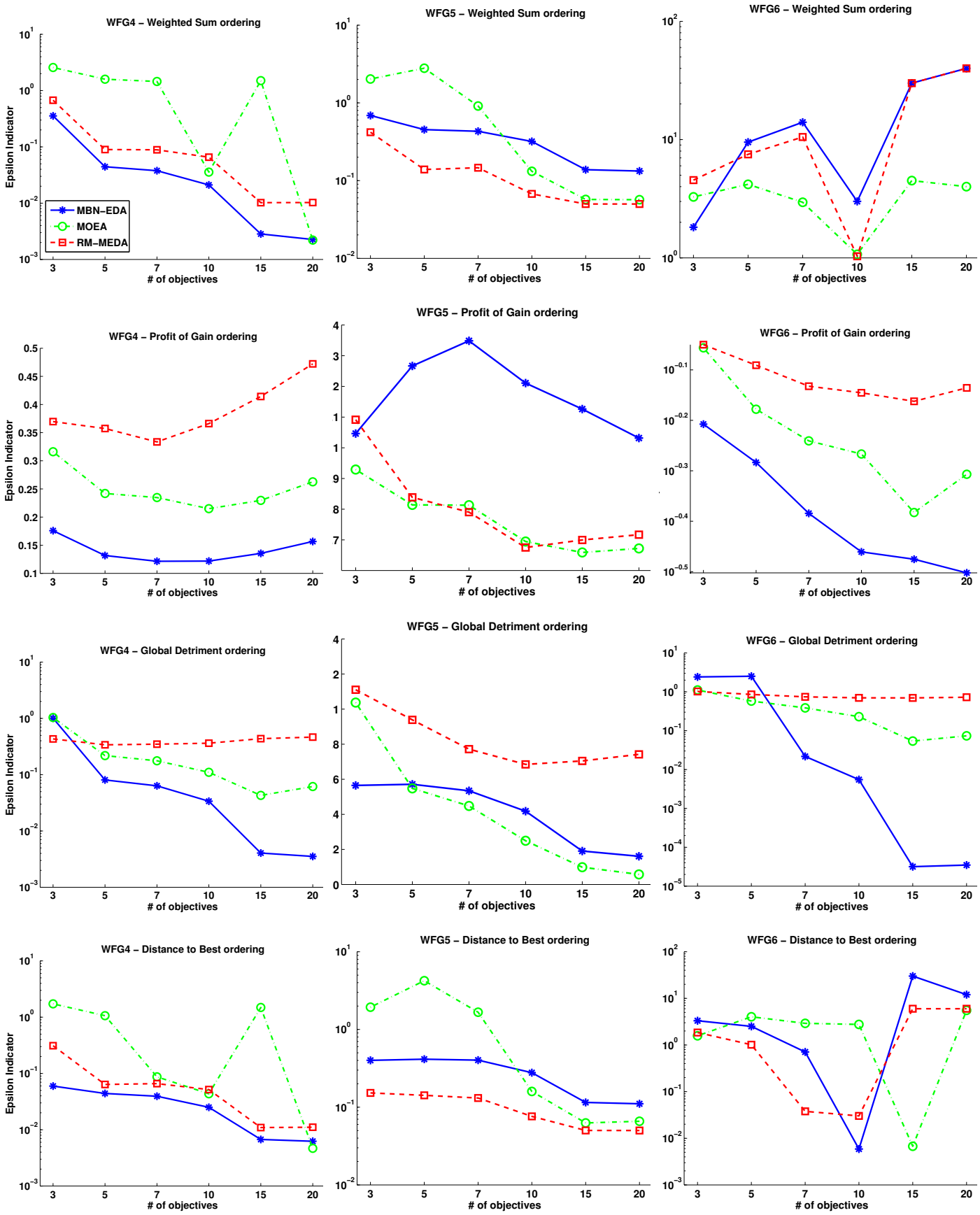



Figure 10.6: The average epsilon indicator values for WFG4 (left column), WFG5 (middle column) and WFG6 (right column) problems.

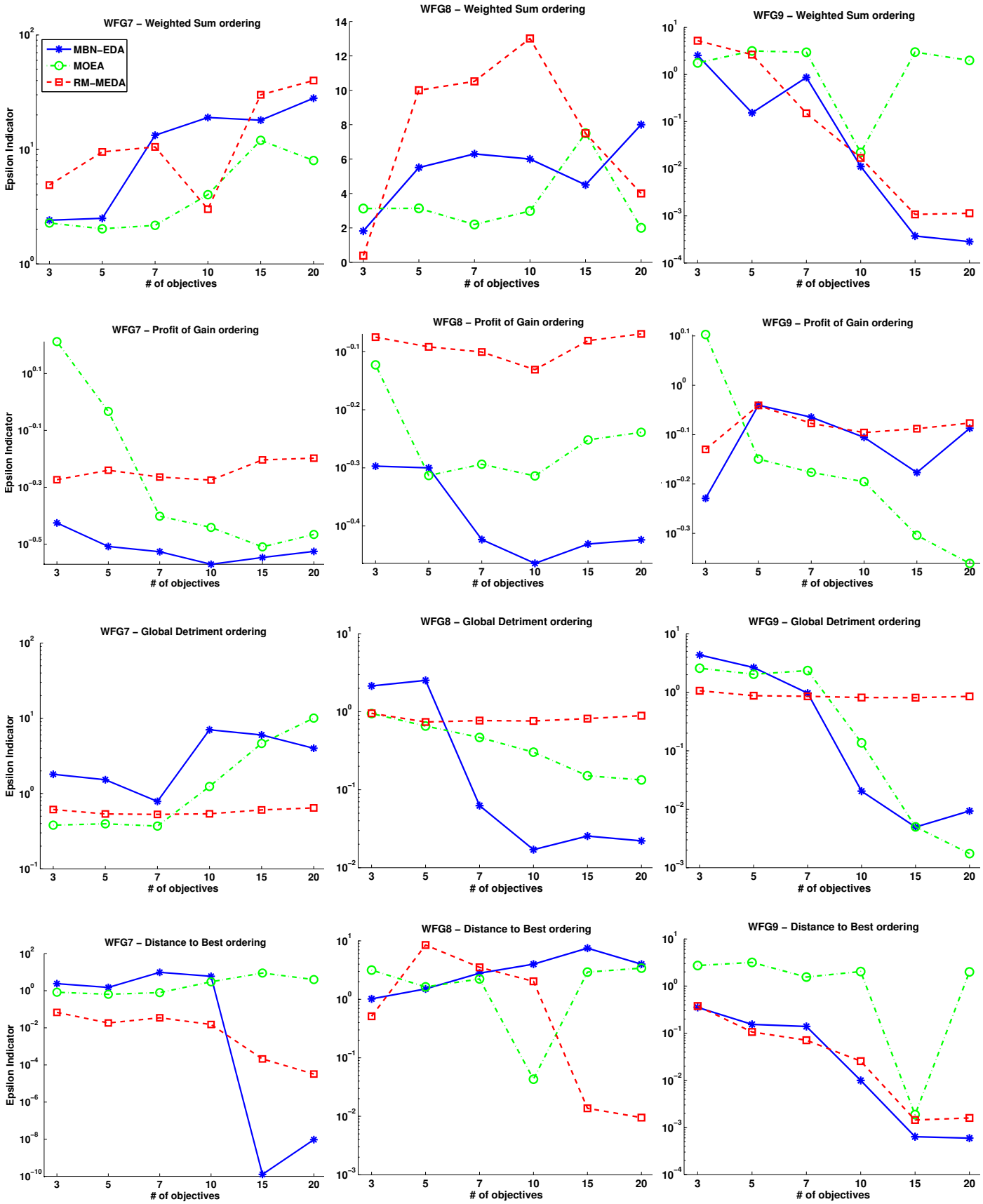



Figure 10.7: The average epsilon indicator values for WFG7 (left column), WFG8 (middle column) and WFG9 (right column) problems.

column). Even though there are more interdependencies between the variables in WFG2, MBN-EDA is able to obtain significantly better results for this problem, evidencing the advantage of its probabilistic model for guiding the solution space search. The difference in the optimal front of the MOP problems does not significantly affect MBN-EDA's optimization ability as observed for WFG3, which is very similar to WFG2 except for the shape of the Pareto optimal front. Moreover, approximating the degenerated front in this problem requires good search process exploitation, which, according to the results for this problem, MBN-EDA is better able to do than the other two algorithms.

For WFG4 (Figure 10.6, left column), where all of the objectives are multi-modal, the optimization results obtained by MBN-EDA with different ranking methods are significantly better in most of objective space dimensions. For some of the ranking methods, MOEA and MBN-EDA performances are comparable as the number of objectives increases, suggesting the usefulness of genetic operators if there are a large number of optima in a problem. When objective function multi-modality is combined with deception, as in WFG5, MBN-EDA performance significantly deteriorates. In fact this algorithm has the worst optimization performance compared with the other two algorithms for this problem, where it obtains significantly worse Pareto set approximations with all the tested ranking methods, and specially for larger objective space dimensions. A possible explanation for this behavior is that the relationships between deceptive objectives do not provide sufficient information to help the algorithm generate good trade-off solutions in the search space.

The interdependencies between variables in WFG6 are more complex than in the WFG2 and WFG3 problems. Therefore, we find that the choice of the ranking method used in solution selection, which provides the training data for model estimation, will play a major role. In this MOP, the results obtained by MBN-EDA with the $g_{PG}$ and $g_{GD}$ ranking methods are significantly better than for the other algorithms, whereas the results are comparable or significantly worse with the other two ranking methods when the number of objectives is increased. This also shows that the gain function defined in Equation (10.5) can be a good measure of solutions superiority in this type of MOP.

In WFG7 and WFG8, the optimum value of each variable is biased, based on the values of other variables. All of the algorithms find it very difficult to deal with this property of the problem. Again, we see (Figure 10.7, left and middle columns) that the choice of ranking method has a significant influence on algorithm performance. With some of the ranking methods (e.g., $g_{PG}$ and $g_{GD}$), MBN-EDA is able to obtain significantly better approximations of the Pareto optimal set for these two problems according to the quality indicator values. The last problem (WFG9) combines many of the properties found in the previous WFG problems. Specifically, apart from variable optimal values being biased, many of the objectives are deceptive as in WFG5. As described previously for WFG5, this prevents MBN-EDA from being able to perform considerably better than the other two algorithms, despite the additional information it collects from data. Nevertheless, the performance of MBN-EDA for this problem is comparable to the other two algorithms, and even, with some ranking methods ($g_{WS}$ and $g_{DB}$), it is significantly better.

In general, the results suggest that there are several factors affecting the optimization performance of the tested algorithms on the selected set of MOPs. According to the employed quality indicator, MBN-EDA is able to obtain better approximations of the Pareto set than the other two algorithms for many of the tested MOPs featuring different

properties and on different objective space dimensions. MBN-EDA finds some specific MOP properties, like deception in the variable values, difficult to deal with.

For some of the tested MOPs, the ranking method plays a crucial role in algorithm performance and some algorithms tend to be more compatible with specific ranking methods. For example, MBN-EDA performed better than the other algorithms for most MOPs when using $g_{<PG>}$ as the ranking method. Since the solution selection mechanism is similar in all algorithms (as they use similar ranking methods), a significant difference in the algorithms performance can be attributed to their solution reproduction mechanism. Therefore, the better optimization results for MBN-EDA may be related to its model estimation and sampling, as it concerns both objectives and variables. Moreover, although the choice of probabilistic model in an EDA is important, it should be noted that the difference between MBN-EDA and RM-MEDA performance is not only due to the difference in their probabilistic models. We showed (in the previous chapter) influence of incorporating objectives into the probabilistic model when comparing GBN-EDA and JGBN-EDA, where the latter obtained significantly better results.

In some of the problem instances (e.g., WFG6 with $g_{PG}$), with the increase in the objective space dimension, the algorithms seem to obtain better Pareto set approximations, resulting in lower quality indicator values. Note, however, that like the algorithms, the computation of quality indicator values is also affected by the increase in the objective space dimension. In larger objective spaces, the Pareto set approximations obtained by the algorithms will become sparser, as they are using small populations. Also since a small reference set is used to evaluate the algorithms, the differences in the performance of an algorithm in different objective space dimensions will not be clear. However, since an equal reference set is used for each specific objective space dimension, the indicator values can be used to compare the performance of different algorithms in the same dimension.

## 10.5 Experiments on CEC09 Test Problems

In this section we compare the performance of MBN-EDA on the 13 unconstrained problems of the CEC09 benchmark [Zhang et al., 2009b] with two other state-of-the-art EMO algorithms. The first 7 problems of this benchmark are bi-objective, the next three are three-objective and the last three contain five objectives. Two of the five-objective problems are modified versions of DTLZ2 and DTLZ3 problems [Deb et al., 2002b], and the other five-objective problem is the previously studied WFG1. In all of the CEC09 benchmark problems, solutions are considered to be of size $n = 30$ (Table 10.3).

### Implementation Details and Experimental Design

The EMO algorithms used for comparison in this section are the decomposition-based multi-objective EA (MOEA/D) [Zhang and Li, 2007] and a hypervolume indicator-based EMO algorithm [Bader and Zitzler, 2011] which we call MOEA-HypE in this paper. In MOEA/D the MOP at hand is decomposed into several single-objective problems using a number of weight vectors and a decomposition method. Here, the Tchebycheff method is used for MOP decomposition in MOEA/D which has been reported to give better results. The choice of weight vectors which should be given in advance to the algorithm is very

Table 10.2: The results of statistical difference tests for different WFG problems with different number of objectives and four different ranking methods. The bold entries show the algorithm obtaining the best ranking according to the statistical test. The numbers in the parentheses shows the number of algorithms that are significantly worse and better than each algorithm, respectively, considering a 0.05 significance level (refer to the text for more discussion of the statistical test).

| | | Weighted Sum | | | Profit Gain | | | Global Detriment | | | Distance To Best | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MBN-EDA | MOEA | RM-MEDA | MBN-EDA | MOEA | RM-MEDA | MBN-EDA | MOEA | RM-MEDA | MBN-EDA | MOEA | RM-MEDA |
| WFG1 | 3 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | (1, 1) |
| | 5 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | (1, 1) |
| | 7 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | (1, 1) |
| | 10 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) |
| | 15 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) |
| | 20 | **(1, 0)** | (0, 2) | (0, 2) | **(2, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | (1, 1) |
| WFG2 | 3 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | (1, 1) |
| | 5 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 7 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |
| | 10 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 15 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 20 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| WFG3 | 3 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | **(1, 0)** |
| | 5 | **(2, 0)** | (0, 2) | (1, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 7 | **(2, 0)** | (0, 2) | **(2, 0)** | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 10 | **(2, 0)** | (1, 1) | **(2, 0)** | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 15 | **(2, 0)** | (1, 1) | **(2, 0)** | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 20 | **(2, 0)** | (1, 1) | **(2, 0)** | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| WFG4 | 3 | **(1, 0)** | (0, 2) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 2) | (1, 1) |
| | 5 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 7 | **(2, 0)** | (0, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |
| | 10 | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |
| | 15 | **(1, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |
| | 20 | (1, 1) | **(1, 0)** | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(1, 0)** | (1, 1) | (0, 2) | **(1, 0)** | (0, 2) | (0, 2) |
| WFG5 | 3 | **(2, 0)** | (0, 1) | **(1, 0)** | **(2, 0)** | (0, 0) | (0, 1) | **(1, 0)** | (0, 0) | (0, 2) | **(2, 0)** | (1, 1) | **(2, 0)** |
| | 5 | **(2, 0)** | (0, 2) | **(2, 0)** | **(2, 0)** | (0, 2) | (1, 0) | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (0, 1) | **(2, 0)** |
| | 7 | **(2, 0)** | (0, 2) | **(2, 0)** | **(2, 0)** | (0, 2) | (1, 0) | **(1, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | **(2, 0)** |
| | 10 | **(2, 0)** | (1, 1) | **(2, 0)** | **(2, 0)** | (0, 2) | (1, 0) | **(1, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | **(2, 0)** |
| | 15 | **(2, 0)** | (0, 2) | **(2, 0)** | **(2, 0)** | (0, 2) | (1, 0) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | **(2, 0)** |
| | 20 | **(2, 0)** | (1, 1) | **(2, 0)** | **(2, 0)** | (0, 2) | (1, 0) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | **(2, 0)** |
| WFG6 | 3 | **(2, 0)** | (0, 2) | (0, 1) | **(2, 0)** | (0, 1) | (0, 2) | **(1, 0)** | (0, 0) | (0, 2) | **(1, 0)** | (0, 0) | (1, 0) |
| | 5 | **(2, 0)** | (0, 0) | (0, 0) | **(2, 0)** | (1, 0) | (0, 2) | **(1, 0)** | (1, 0) | (0, 2) | **(1, 0)** | (0, 2) | (1, 0) |
| | 7 | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 10 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 0) | **(1, 0)** |
| | 15 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 2) | **(1, 0)** |
| | 20 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(1, 0)** | (0, 1) | **(1, 0)** |
| WFG7 | 3 | **(2, 0)** | (1, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) | **(1, 0)** | (0, 1) | (0, 1) | **(1, 0)** | (0, 2) | **(1, 0)** |
| | 5 | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(1, 0)** | (0, 1) | **(2, 0)** |
| | 7 | **(1, 0)** | (0, 1) | (0, 0) | **(2, 0)** | (1, 1) | (0, 2) | **(1, 0)** | (0, 1) | (0, 2) | **(1, 0)** | (0, 2) | (1, 1) |
| | 10 | (0, 0) | (0, 0) | (0, 0) | **(2, 0)** | (1, 1) | (0, 2) | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 15 | (0, 0) | **(1, 0)** | (0, 1) | **(1, 0)** | (1, 0) | (0, 2) | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| | 20 | (0, 0) | **(1, 0)** | (0, 1) | **(2, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (0, 2) | (1, 1) |
| WFG8 | 3 | **(1, 0)** | (0, 0) | (1, 0) | **(2, 0)** | (1, 0) | (0, 2) | **(1, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) |
| | 5 | **(1, 0)** | (0, 0) | (0, 0) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) |
| | 7 | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) |
| | 10 | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 1) | (0, 1) |
| | 15 | (0, 0) | (0, 0) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(2, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (0, 1) | (0, 2) |
| | 20 | (0, 0) | (0, 0) | (0, 1) | **(2, 0)** | (1, 1) | (0, 2) | **(1, 0)** | (1, 0) | (0, 2) | **(2, 0)** | (1, 1) | (0, 2) |
| WFG9 | 3 | **(1, 0)** | (0, 1) | (0, 1) | **(1, 0)** | (0, 1) | (0, 1) | **(1, 0)** | (0, 0) | (0, 1) | **(1, 0)** | (0, 2) | (1, 0) |
| | 5 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 0) | (0, 1) | **(1, 0)** | (0, 1) | (0, 1) | **(1, 0)** | (0, 2) | (1, 0) |
| | 7 | **(2, 0)** | (0, 2) | (1, 0) | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (0, 1) | (0, 2) | **(2, 0)** | (0, 1) | **(1, 0)** |
| | 10 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (0, 0) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |
| | 15 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (0, 0) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |
| | 20 | **(2, 0)** | (0, 1) | (0, 1) | **(2, 0)** | (0, 0) | (0, 1) | **(2, 0)** | (0, 0) | (0, 2) | **(2, 0)** | (0, 1) | (0, 1) |

Table 10.3: Characteristics of the unconstrained MOPs used in CEC09 benchmark.

| Name | No. Objectives | No. Variables | $F^*$ Size |
|---|---|---|---|
| UF1 | 2 | 30 | 1000 |
| UF2 | 2 | 30 | 1000 |
| UF3 | 2 | 30 | 1000 |
| UF4 | 2 | 30 | 1000 |
| UF5 | 2 | 30 | 21 |
| UF6 | 2 | 30 | 1000 |
| UF7 | 2 | 30 | 1000 |
| UF8 | 3 | 30 | 10000 |
| UF9 | 3 | 30 | 10000 |
| UF10 | 3 | 30 | 10000 |
| DTLZ2 | 5 | 30 | 5000 |
| DTLZ3 | 5 | 30 | 5000 |
| WFG1 | 5 | 30 | 5000 |

important and can influence its performance. For two- and three-objective MOPs, we have used the weight vector generation method proposed in the original paper by setting the number of different weight levels to respectively $H = 99$ and $H = 19$, resulting in population sizes of $N = 100$ and $N = 210$, respectively. For five-objective MOPs, the method proposed in [Zhang et al., 2009a] is used to generate a well distributed set of weight vectors for a population size of $N = 300$ and a random pool of 15000 weight vectors. The rest of algorithm parameters are set to their default values according to the original paper.

MOEA-HypE belongs to the group of indicator-based EMO algorithms [Zitzler and Künzli, 2004]. These algorithms use the quality indicators values, usually used to evaluate the final Pareto set approximations, to rank and select solutions in the course of evolution. One of the quality indicators used for this purpose is the hypervolume indicator. Given a Pareto set approximation $A$, its hypervolume indicator value with respect to a set $R$ of reference points in the objective space, denoted as $I_h(A, R)$, is given as

$$I_h(A, R) = \bigcup_{\boldsymbol{x} \in A} H(\boldsymbol{f}(\boldsymbol{x}), R), \tag{10.11}$$

where $H(\boldsymbol{a}, B)$ indicates the (hyper-)volume between point $\boldsymbol{a}$ and the points of set $B$ in the objective space. A higher value for this indicator implies a better Pareto set approximation. Computing the hypervolume indicator is very time consuming and its computational complexity is exponential in the number of objectives.

To simplify this computation, in MOEA-HypE a set of points are randomly sampled in the objective space and is used to obtain an estimation of the hypervolume dominated by each solution. These estimated hypervolume values are then used to rank the solutions and select a subset as parents with a tournament selection strategy (with a tournament size of 5) for offspring generation. In the replacement step, population and offspring

solutions are aggregated and sorted into a number of Pareto sets, using the non-dominated sorting algorithm. However, instead of computing the crowding distances of solutions within each Pareto set, the estimated hypervolume values of the solutions are used to select the elite solutions in each set for forming the population of the next generation. In this study we use a sample of 10000 points for hypervolume estimation as it is suggested in the original paper of this algorithm. The rest of algorithm parameters are set to their default values.

Both of these algorithms use the same genetic operators as the ones used in NSGA-II (simulated binary crossover and polynomial mutation) for generating new solutions. However, MOEA-HypE generates $N/2$ new offspring solutions in each generation, whereas MOEA/D generates $N$ new offspring solutions by evolving all of the solutions in the population. This means that during optimization, MOEA/D processes twice the number of solutions that is searched by MOEA-HypE. Therefore, to have a fair comparison between the algorithms, we impose a similar maximum number of fitness evaluations for both of the algorithms by setting the population size of MOEA-HypE to twice the size of population in MOEA/D.

MBN-EDA is tested with three different ranking methods in these experiments. From the ranking methods introduced in Section 10.3.1, we have selected $g_{DB}$ and $g_{GD}$. In order to extend the diversity of solutions when using $g_{DB}$, a tournament selection strategy similar to that used by MOEA-HypE is adopted for this ranking method. In addition to these two ranking methods, MBN-EDA is also tested when using HypE method for ranking and selecting the solutions, in order to compare joint modeling in MBN-EDA with genetic operators in the other two EMO algorithms. Other parameters of MBN-EDA, like population size, are set similar to those in MOEA-HypE.

The quality indicator suggested in CEC09 benchmark for comparing the approximated fronts is IGD indicator. The size of the sampling set provided for the Pareto optimal front of each of the MOPs in CEC09 benchmark is shown in Table 10.3. In addition to this quality indicator, we have also evaluated the approximated Pareto sets with epsilon and hypervolume indicators. The sampling of the Pareto optimal front is also used as the reference set when computing the epsilon indicator. For the hypervolume indicator, a point in the objective space having the worse values for all of the objective functions, usually referred to as the nadir point, is used as the reference. This point is equally set for all of the algorithms to 10000 in all of the objective dimensions and for all of the problems.

### Results

Figures 10.8–10.11 show the values of the quality indicators for the final Pareto sets approximated by each of the algorithms on each of the problems, in 20 independent runs. The maximum number of generations is equally set for all of the algorithms to 300.

The results show that the algorithms have different behaviors on each of the tested MOPs. In general it seems that the problems in CEC09 benchmark are better optimized when using the decomposition method in MOEA/D for guiding the search. With this method, each of the subproblems will guide the search into a different subspace of the objective space, depending on the weight vector which is used for decomposition. Therefore the algorithm is able to search different regions of the objective space even until the

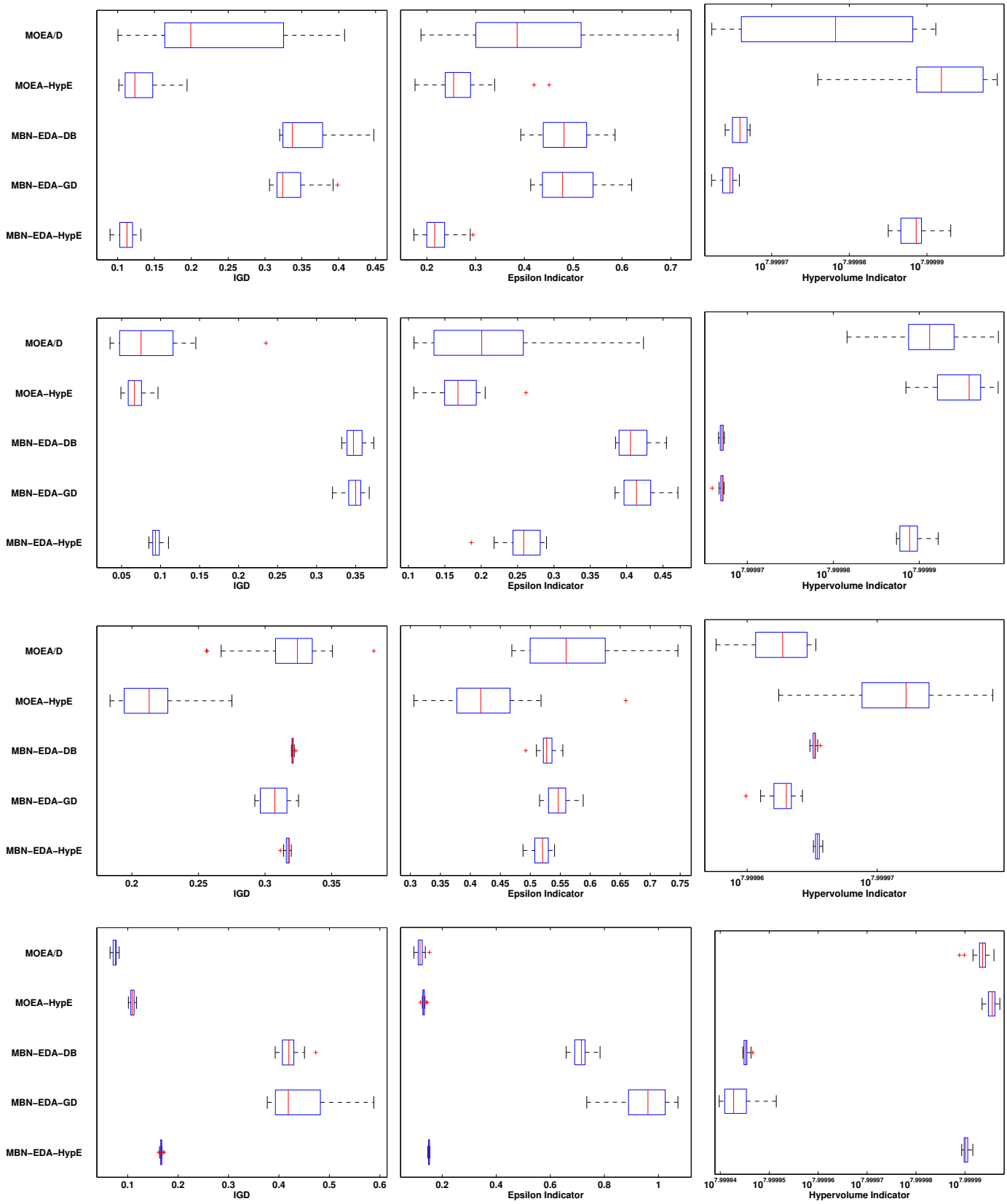



Figure 10.8: The IGD, epsilon and hypervolume indicators values for UF1–UF4 problems of the CEC09 benchmark. Each row corresponds to one of the problems with a top-down order.

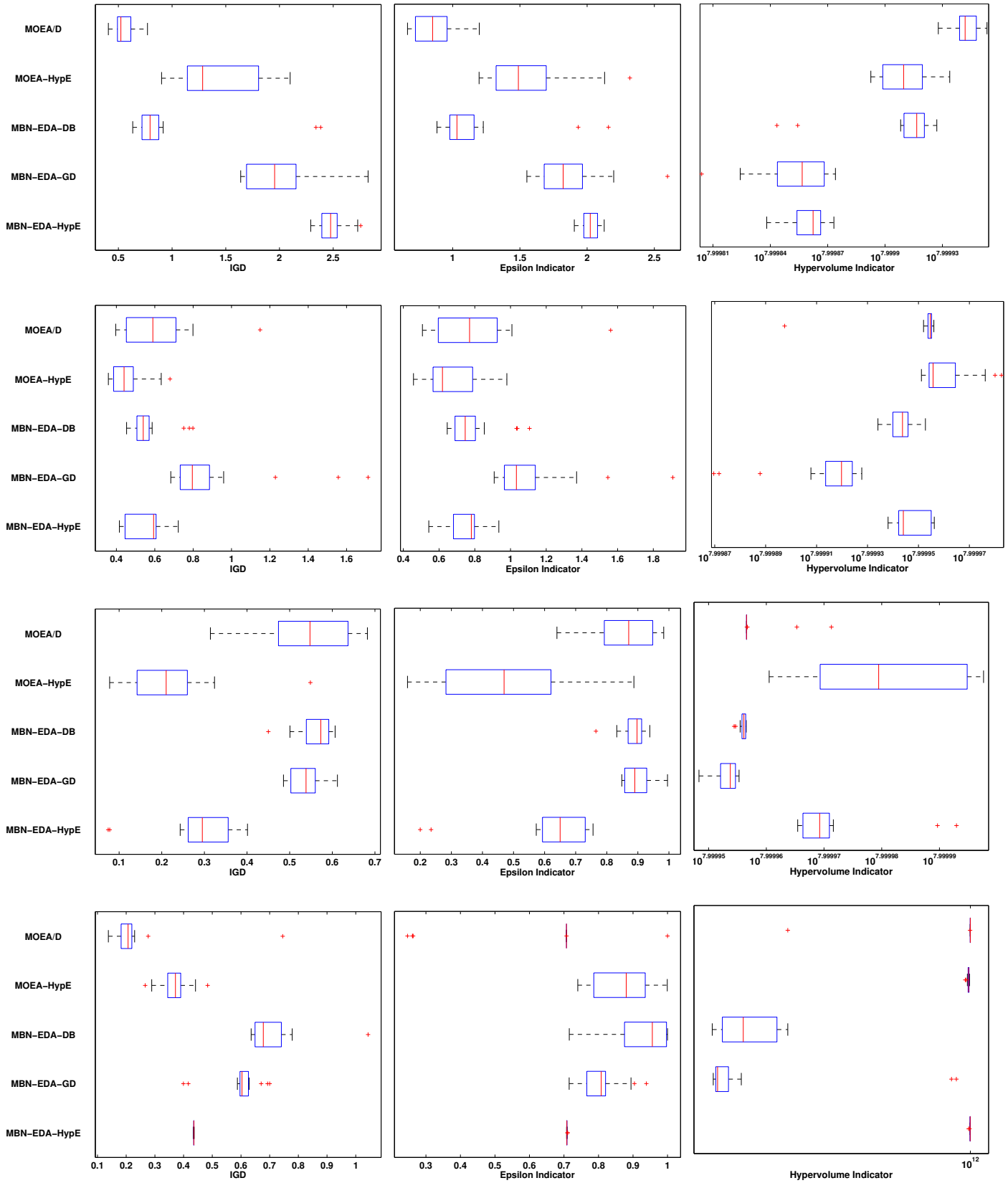



Figure 10.9: The IGD, epsilon and hypervolume indicators values for UF5–UF8 problems of the CEC09 benchmark. Each row corresponds to one of the problems with a top-down order.

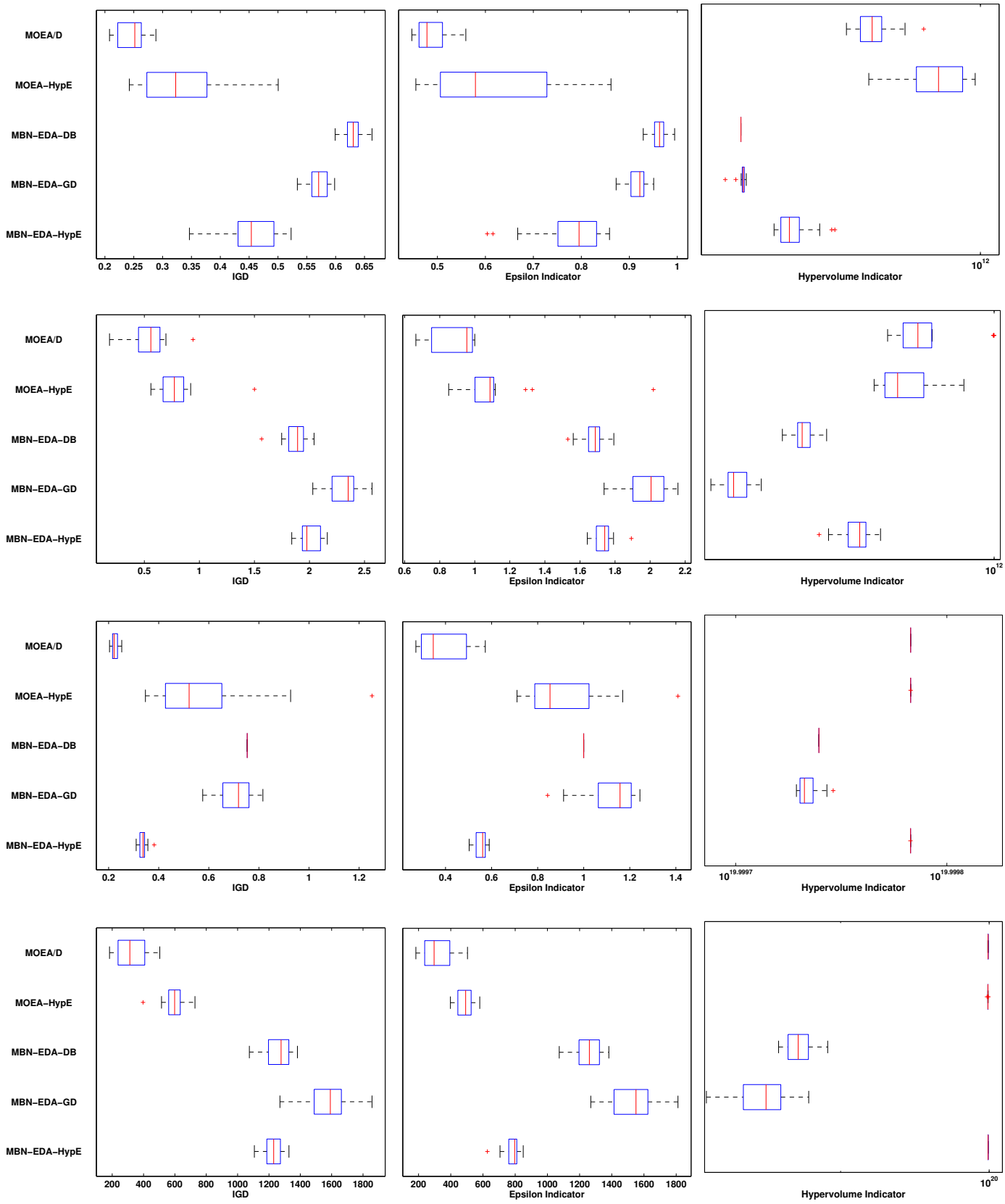



Figure 10.10: The IGD, epsilon and hypervolume indicators values for UF9, UF10, DTLZ2 and DTLZ3 problems of the CEC09 benchmark. Each row corresponds to one of the problems with a top-down order.

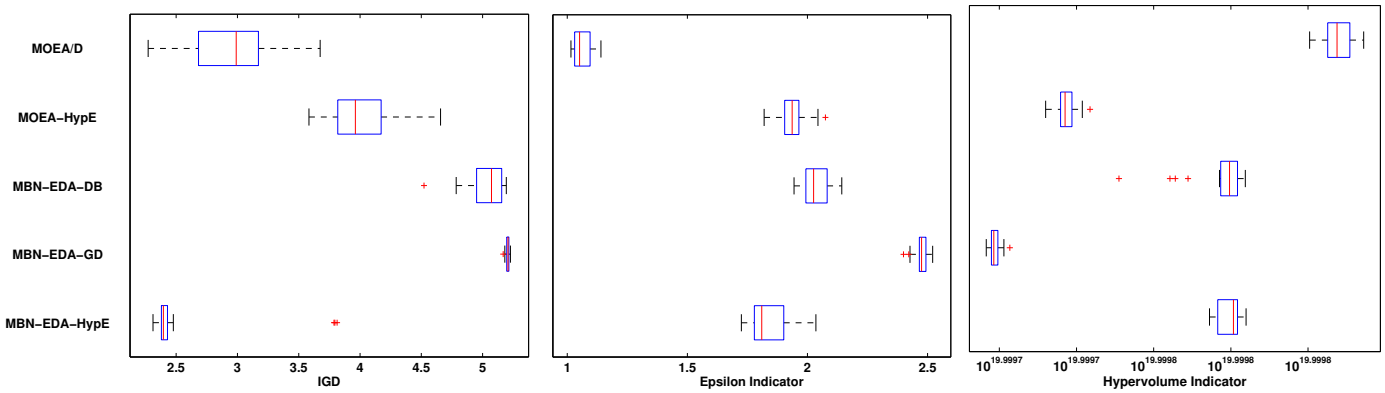



Figure 10.11: The IGD, epsilon and hypervolume indicators values for WFG1 problem of the CEC09 benchmark.

final stages of the evolution.

However, for some of the MOPs in this benchmark, like UF1, UF3, UF6 and UF7, where the optimal Pareto set and its corresponding optimal Pareto front have almost similar geometry, the other dominance-based algorithms are able to obtain better approximations of the Pareto optimal front. Especially on problems UF1 and UF7, MOEA/D is outperformed by the HypE method for solution ranking, whether using genetic operators or joint model estimation for generating new solutions. An explanation for this behavior is that using decomposition on some of the MOPs causes the algorithm to miss certain information about the promising areas of the search space, thus reducing the effectiveness of this method.

It can be observed that the overall performance of MBN-EDA with HypE selection method is superior to the performance observed with the other two ranking methods, according to the values of quality indicators. This suggests that the hypervolume indicator estimation provides a better solution ranking than $g_{DB}$ and $g_{GD}$ methods. A closer look at the quality of the approximated fronts along different generations of the evolution has also revealed that HypE is less affected by the specific geometry of the search space. Again, there are some problem instances like UF5, UF6 and UF10 for which $g_{DB}$ provides a better solution ranking than HypE, resulting in better Pareto set approximations according to the indicator values.

The comparison between MOEA-HypE and MBN-EDA-HypE indicates that for some of the tested MOPs like UF1, UF8, DTLZ2 and WFG1, which cover MOPs with different number of objectives, the Pareto sets approximated by MBN-EDA-HypE are better according to the indicator values. Since both of these algorithms are using similar selection methods, this improvement in the results can be directly attributed to the better solution search in MBN-EDA with its joint modeling of variables and objectives. Our investigation of the populations evolved in MOEA-HypE and MBN-EDA-HypE suggests that joint modeling in MBN-EDA allows the algorithm to rapidly improve its approximated front in early generations of the search, whereas with the genetic operators in MOEA-HypE usually the improvement of the approximated front is slower. However, the diversity of the population in MBN-EDA may not be preserved very well during evolution and the algorithm can enter a stagnation state. On the other hand, with genetic operators in MOEA-HypE more diverse populations are generated during search and except for some

of the MOPs, the algorithm can constantly improve its approximated front.

A point that should be noted here is that when HypE selection method is used, the algorithms are directly optimizing the hypervolume indicator. Thus, using the same indicator to evaluate their achieved results may not properly reflect their performances, especially because the hypervolume indicator may overrate certain regions of the approximated fronts [Zitzler and Thiele, 1999]. For the algorithms with HypE selection method, the results provided by IGD and epsilon indicators are a better reference of their performance.

## 10.6 Problem Structure Estimation

One of the advantages of EDAs is that apart from finding solutions to the optimization problem, they estimate a probabilistic model which captures certain regularities of the problem search space and are common among the solutions. This kind of meta information is especially useful when the intrinsic properties of the problem at hand are unknown in advance. Especially, in multi-objective optimization, the estimated model encodes common properties of the approximated Pareto set which, in decision making, can be used together or even instead of the non-dominated solutions, e.g. when the size of the approximated Pareto set is very large.

Recently, several works have studied the use of data mining techniques to obtain new knowledge from the approximated Pareto sets in EMO algorithms after optimization [Bandaru and Deb, 2011; Deb et al., 2012]. However, multi-objective EDAs already obtain this kind of information during optimization, depending on the probabilistic model they use. Specifically, the type of joint modeling of both variables and objectives used in MBN-EDA offers a systematic way for estimating the structure of an MOP.

A major concern of our study in this chapter is to analyze the MOP structures estimated by MBN-EDA in the course of evolution. These structures are important not only because they can improve optimization by providing information about different types of (in)dependencies existing in the problem, but also because they can give DMs more control over the selection of the desired information from the Pareto set approximations [Ulrich et al., 2008], and better insight into how different variables influence the objectives or the way objectives interact [Chan and Chung, 2004]. To examine MBN-EDA's ability to retrieve the MOP structure, we have analyzed the structures estimated for WFG1 problem with five objectives and 16 variables in a number of different case studies. To include the factor of different training data used for estimating MOP structure in the analysis, MBN-EDA is tested with two of the ranking methods described in Section 10.3.1 in this study, namely $g_{PG}$ and $g_{DB}$.

### Selection of Relevant Variables

In the first case study, nine irrelevant variables are added to the problem and uniformly distributed among other variables. These variables do not affect the outcome of objective functions of the MOP. Figure 10.12 shows the absolute weight of the links encoded in MBN's bridge substructure between objectives and variables along the evolution path of MBN-EDA. The weights are averaged over 20 independent runs. We can see that MBN-EDA is able to clearly distinguish between relevant and irrelevant variables in the studied

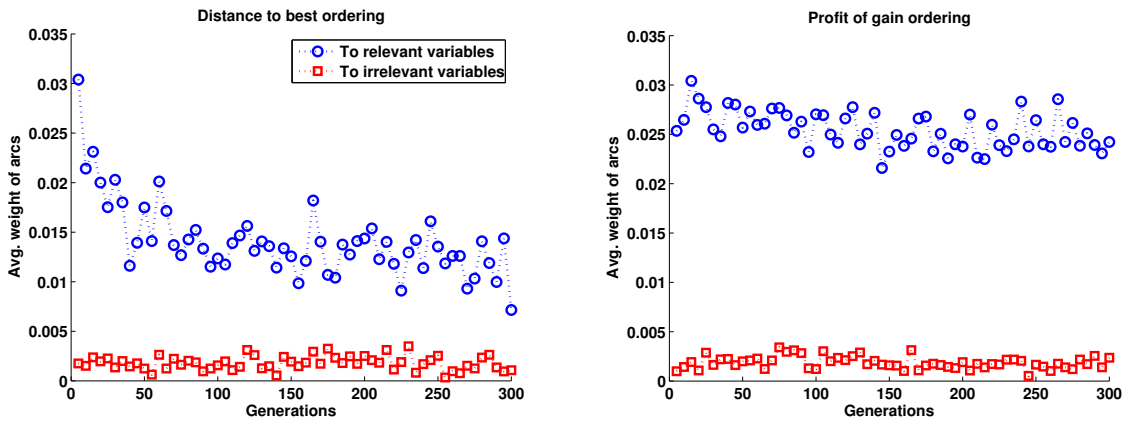



Figure 10.12: The average weight of the links from objectives to variables in the 5-objective WFG1 problem which includes irrelevant variables.

MOP. The low weight of the links between objectives and irrelevant variables in the estimated MBNs is either because the objectives and these variables have been encoded as conditionally independent given other variables and objectives, or any existing link has a very small weight, allowing the algorithm to bypass the noise introduced by these variables to the problem. Although the models are estimated from different initial populations in the each of the runs, the structural information encoded between objectives and variables is very similar. It is also shown that the populations selected according to the $g_{PG}$ ranking method help to better distinguish between relevant and irrelevant variables especially in the final generations where the algorithm focuses on specific regions of the search space.

### Identifying Redundant Objectives

The second case study analyzes the structures estimated for an eight-objective WFG1 problem with three pairs of similar objectives. Figure 10.13 compares the absolute weight of the arcs between similar objectives with those between dissimilar objective pairs, encoded in the class substructure of MBN in different generations of MBN-EDA. The results are averaged over 20 independent runs. The relatively high weights between similar objectives show that MBN-EDA is correctly encoding a strong dependency between these objectives compared with the other relationships in the class subgraph. Note that a closer inspection of the models estimated in different runs with different initial populations has revealed that such a dependency between similar objectives is encoded in the models estimated in all of the runs. We can also see that the information about the similarity of objectives in the MBN's class subgraph is better captured from the populations selected according to the $g_{DB}$ ranking method.

### Estimated Structures

Based on the observations from the above two case studies, the third case study directly inspects the structures estimated by MBN-EDA for the WFG1 problem. In the 5-objective WFG1 problem considered in this section, the number of position variables is set to $k = 4$ and the number of distance variables is set to $l = 12$ (see Section 9.4.1). A

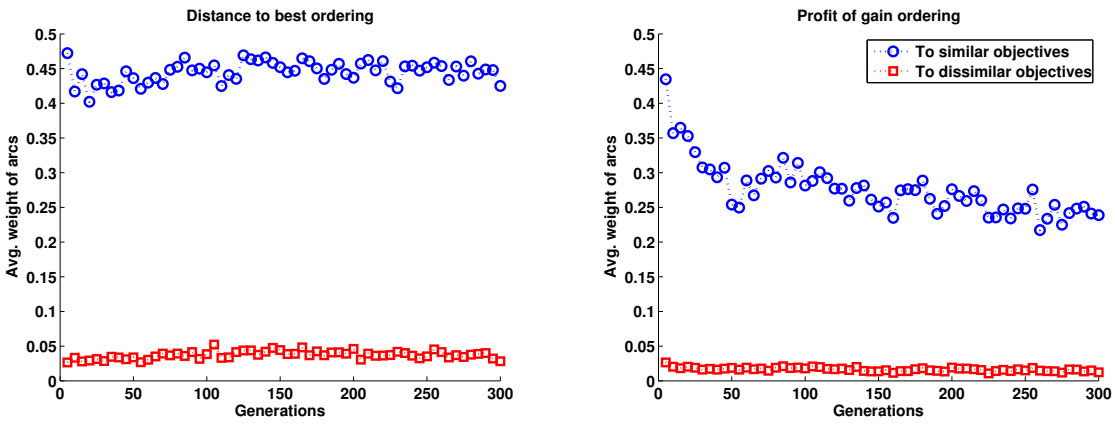



Figure 10.13: The average weight of the links between objectives in the 8-objective WFG1 problem with three pairs of similar objectives.

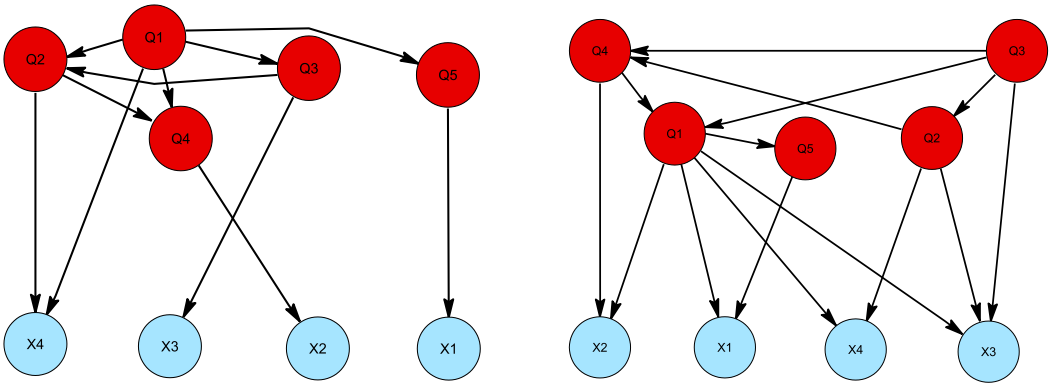
simplified definition of the five objective functions in this problem can be given as follows [Huband et al., 2006]:

$$\begin{aligned} f_1(\boldsymbol{x}) &= a + 2 \cdot h_1\big(g_2(x_1), g_2(x_2), g_2(x_3), g_2(x_4)\big) \\ f_2(\boldsymbol{x}) &= a + 4 \cdot h_2\big(g_2(x_1), g_2(x_2), g_2(x_3), g_2(x_4)\big) \\ f_3(\boldsymbol{x}) &= a + 6 \cdot h_3\big(g_2(x_1), g_2(x_2), g_2(x_3)\big) \\ f_4(\boldsymbol{x}) &= a + 8 \cdot h_4\big(g_2(x_1), g_2(x_2)\big) \\ f_5(\boldsymbol{x}) &= a + 10 \cdot h_5\big(g_2(x_1)\big), \end{aligned} \tag{10.12}$$

where $a = g_1(x_5, \ldots, x_{16})$, and $g_1(\cdot)$ and $g_2(\cdot)$ represent a composition of transformation functions on the input variables.

Figure 10.14 shows part of the structures learnt for this problem, consisting of the significant arcs and their corresponding nodes with an average absolute weight value greater than a threshold set to $w \geq 0.1$ (constituting about 7% of the most significant arcs). While there are many links capturing the obscure relationships between variables (not depicted here), it is evident that MBN-EDA attaches more importance to the links between objectives in the class subgraph, and between the objectives and the first four variables in the bridge subgraph. Moreover, these dependencies conform to the function definitions given in Equation (10.12). For example, the link between objective nodes $Q_2$ and $Q_4$, which is captured using both of the tested ranking methods, is supported by the fact that $h_2(\cdot)$ is actually a multiplication of $h_4(\cdot)$ and two other factors obtained from variables $X_3$ and $X_4$. Another example is the relationship between objective node $Q_1$ and the nodes corresponding to the first four variables, either directly or through the relationships with other objectives, since all of these variables influence the value of the first objective function.

An important point to note here is the significance of the information provided by the dependencies between objectives, and between objectives and variables in multi-objective optimization from MBN-EDA point of view. There are some studies in the literature that analyze how the dependencies between variables are represented in probabilistic models [Santana et al., 2009a]. But, to the best of our knowledge, the importance of




(a) Distance to best ordering

(b) Profit of gain ordering

Figure 10.14: Part of the structure learnt for the 5-objective WFG1 problem showing the most significant edges and their corresponding nodes.

the interactions involving objectives have not been considered so far in other EDAs used for multi-objective optimization. Such dependencies allow the proposed MBN-EDA to approximate how the variables can affect objective values, which is used to generate new solutions with better objective values.

## 10.7 Conclusions

The similarity between multi-dimensional classification and multi-objective optimization motivates the use of MBNs in the context of EDAs to solve MOPs. A new modeling approach in multi-objective EDAs was proposed that uses MBN estimation to learn a joint model of objectives and variables while at the same time distinguishes their role in the model. The estimated joint model encodes three types of relationships between variables and objectives in three different substructures.

The proposed joint modeling was coupled with different ranking methods introduced in the literature for many-objective optimization and tested on two sets of benchmark problems with different number of objectives. The exhaustive experiments comparing MBN-EDA with several state-of-the-art EMO algorithms show promising performance when using the proposed joint modeling for guiding the search in the space of candidate solutions. According to the epsilon quality indicator and compared with a standard EMO algorithm and a competitive EDA, MBN-EDA was able to obtain significantly better approximations of the Pareto set for many of the WFG problems, with a significance level of $\alpha = 0.05$. We found that the choice of ranking methods has a major influence on the performance of the algorithms for some of the MOPs, as they determine the population used for model estimation and offspring generation. The results also show that the proposed MBN-EDA was unable to satisfactorily deal with some MOP properties, like deception in the objective values. The results of the second set of experiments on the CEC09 unconstrained problems show that on some of the MOPs in this benchmark, the joint modeling in MBN-EDA allows to find considerably better fronts according to three different quality indicators.

The proposed joint modeling approach also obtains the MOP structure which can be used for decision making. An analysis of the structures estimated by MBN-EDA along the evolution path showed that the proposed algorithm is able to distinguish between relevant and irrelevant variables, performing a type of variable selection for the objectives encoded in the model. It can also capture stronger dependencies between similar objectives which helps to identify redundant objective functions. The analysis of the specific structures estimated for the 5-objective WFG1 problem shows that MBN-EDA is able to obtain a very good approximation of this MOP structure. We saw that the information provided by the relationships between variables and objectives and the inter-objectives dependencies, the two types of interactions completely overlooked by other EDAs, can be very important for multi-objective optimization.

# Chapter 11

# Interval-Based Ranking in Noisy Evolutionary Multi-Objective Optimization

## 11.1 Introduction

A feature of many real-world problems is the existence of noise, which can appear as variable or environmental change, or/and as uncertainty in the objective values. When EMO algorithms based on Pareto dominance relation are used for solving these problems, the noise in objective values can mislead the algorithm by selecting inferior solutions that are considered good because of noise, and discarding good solutions that are necessary for directing the search to promising areas.

One of the ways to deal with noisy objectives is to assume each objective returns an interval of values for a solution. This interval can be obtained by considering the range of error or amount of noise in the system, or from a set of values obtained from multiple reevaluations of a solution in different conditions. Considering an interval of values instead of a single value allows the EMO algorithm to take into account the extent of noise in the objectives when selecting solutions for recombination.

In this chapter we present a new solution ranking, called $\alpha$-degree Pareto dominance, for EMO algorithms applied to noisy problems when objective functions return intervals. This new relation for ranking the solutions allows the intervals to overlap each other, and determines the dominance among solutions based on the extent to which their corresponding intervals are better than other solutions. The method can also be extended to the case where only some of the objective functions of the MOP at hand are noisy. This solution ranking method is then integrated into MBN-EDA, proposed in Chapter 10, for multi-objective optimization of noisy MOPs by adapting the joint variable-objective modeling method of the algorithm. The contents of this chapter are taken from [Karshenas et al., 2013a].

## 11.2 A Survey of Evolutionary Multi-Objective Optimization with Noise

The topic of EMO under uncertainty and noise has recently gained a lot of attention, and many studies and methods are reported in the literature. In general, as explained by Jin and Branke [2005], three different types of noise handling can be identified in EAs. First, the population-based search in these algorithms, by itself, implicitly deals with certain levels of noise in objective values. The average quality of a population or subgroups of the population, presenting certain subspaces, is less susceptible to noise. Therefore, the larger the population is, the better the algorithm can overcome the noise in the quality of solutions for finding the optimal solutions. However, the existence of several objectives in a multi-objective scenario reduces the effectiveness of EAs in noise handling [Tan and Goh, 2008]. Second approach is to explicitly reevaluate each solution of the population several times to obtain a better estimation of its objective values. Although this approach greatly increases the computational cost of optimization, for some of the problems where the objective values are obtained as the result of simulations, it is inevitable. A third approach is to consider the noise in objective values in the selection step of EAs. This is usually done by modifying the solution ranking method, assuming a level of uncertainty in the objective values.

As explained before, noise can exist in the values of the input variables, the outputs of the objective functions or even both. There are some works on EMO algorithms considering the former type of noise, i.e. noise in the input values, which is sometimes referred to as robust optimization and its aim is to find solutions with the highest stability in their objective values. Soares et al. [2009] optimized the worst noisy objective values of the solutions (in a min-max formulation) using interval analysis. To decrease the amount of uncertainty in the intervals they propose to recursively divide the intervals into halves, resulting in a grid which is placed on the objective space and is used to compute the worst objective values of the solutions. The grid also serves as a niching method, penalizing the solutions that are very close in the objective space. The noise is introduced in the input variables using an uncertainty vector which is incorporated in the definition of the objective functions. Goh et al. [2010] offered a classification of the noisy MOPs depending on the effect of noise on the Pareto front, Pareto set and landscape of the problem. Based on this classification, they proposed some guidelines for designing challenging MOPs for robust optimization, and introduced a Gaussian landscape generator using a number of basis functions for this purpose.

Most of the algorithms proposed for EMO in noisy environments consider the second type of noise, i.e. noise in the objective values, since this type of noise is harder to deal with. In this type of MOPs it is usually assumed that the objective values are distorted by an additive noise value. Almost all EMO algorithms reviewed in this section consider a Gaussian noise model for the objective values of the tested MOPs, unless otherwise mentioned.

One of the earliest and well-known approaches to deal with noisy objectives when comparing two solutions in multi-objective optimization is to use probabilistic dominance, considering the probability that one solution dominates another [Teich, 2001; Hughes, 2001]. This approach has been successfully used for optimization in some of the real world problems like designing the shape of acoustic noise attenuation barriers with several

receiver points [Greiner et al., 2009]. Since this method is considered as a main reference in many later works on noisy objectives, we explain here it in more detail.

**Definition 11.1. (Dominance Probability)** ***[Teich, 2001; Hughes, 2001]*** *Let $\boldsymbol{U} = \boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{V} = \boldsymbol{f}(\boldsymbol{y})$ be two vectors of random variables representing the objective values returned for two solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ of a noisy MOP in decision space $\mathbb{D}$. Then the probability that $\boldsymbol{x}$ dominates $\boldsymbol{y}$ is given by:*

$$P(\boldsymbol{x} \prec \boldsymbol{y}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \rho_{\boldsymbol{U}}(\boldsymbol{t})\big(1 - \Psi_{\boldsymbol{V}}(\boldsymbol{t})\big)d\boldsymbol{t}, \tag{11.1}$$

*where $\rho_{\boldsymbol{U}}(\cdot)$ denotes the joint probability distribution for the output of objective functions given solution $\boldsymbol{x}$, and $\Psi_{\boldsymbol{V}}(\cdot)$ is the cumulative probability distribution for the output of objective functions given solution $\boldsymbol{y}$.*

Based on the probability of dominance, ranking strategies like the mean dominance probability [Teich, 2001] is used to order the solutions with noisy objective values. Hughes [2001] also tested this approach when the noise is introduced to the input values, showing the ability of this method for handling noise in both input variables and output objective functions. Bui et al. [2005a] adopted probabilistic dominance in NSGA-II [Deb et al., 2002a] and compared it with a version of this algorithm which is based only on reevaluation of objectives. They also proposed a fitness inheritance method to reduce the complexity of reevaluating the objectives. In a separate work [Bui et al., 2005b] they performed a comparative study of the noise handling capability of the original versions of NSGA-II and SPEA2 [Zitzler et al., 2001] in the presence of different noise levels, using several performance measures. Fieldsend and Everson [2005] proposed a Bayesian method for obtaining an estimation of the Gaussian noise variance when computing the probability of dominance between two solutions. They considered different scenarios like unknown, constant and variable noise for the objective values.

Many of the EMO algorithms proposed for noise handling try to extend the existing solution ranking and selection methods by considering noise in the values of the objectives. Büche et al. [2002] proposed a noise-tolerant EA using the Pareto strengths-based solution ranking method [Zitzler and Thiele, 1999] assigning a lifetime (number of generations) to the solutions in the archive depending on their strengths. This lifetime is used to determine which solutions are to be reevaluated or to be used for updating the archive in order to limit the influence of noisy objective values and outliers. They claim that elitism does not necessarily result in faster convergence when noise is present in multi-objective optimization. The proposed algorithm is applied to find an optimal flow of fuel in the burner of a gas turbines.

Babbar et al. [2003] modified NSGA-II to include neighbors of the non-dominated solutions in the first Pareto front. Neighboring solutions are determined using the mean and variance of the objective values for each solution, estimated from the reevaluation of the solutions. Solutions that are reevaluated less than a predetermined number of times during the evolution are considered as outliers and removed from the final Pareto set. Goh and Tan [2007] proposed three techniques for noise handling in an EMO algorithm which uses a two-part, discrete-continuous, representation for the solutions. The first technique consists of incorporating the direction of population movement along evolution into the generation of new solutions. The second technique is stretching or shrinking

the search domain of each variable depending on the behavior of the algorithm in the current phase of evolution. The third technique is to assume objective values are given as fuzzy numbers and then use two newly proposed dominance relations, called necessity and necessity-possibility, to update the archive of non-dominated solutions maintained by the algorithm.

Eskandari and Geiger [2009] considered the expected values of the objective functions and proposed the stochastic dominance relation for ranking the solutions. In the selection process, the solutions are divided into two sets depending on whether they are stochastically dominated. The set of stochastically dominated solutions are further ordered by considering the expected strength of each solution depending on both how many solutions it dominates and by how many solutions it is dominated. To detect the algorithm convergence in noisy environments they proposed to monitor the rate of hypervolume indicator growth. Bui et al. [2009] proposed the use of adaptive non-overlapping hyper-spheres which are locally moved in the search space to reduce the effect of noise. The motivation is that the average objective values of the solutions in a neighborhood of the search space provide a better approximation of the direction of movement during evolution. A PSO inspired algorithm is used to update the center and radius of the hyper-spheres in the search space. The algorithm also deploys an archive of solutions and its corresponding hyper-sphere to represent the global behavior of the population.

Syberfeldt et al. [2010] proposed a method to increase the efficiency of solution reevaluations. When comparing two solutions in the non-dominated sorting algorithm [Deb et al., 2002a], if the confidence in the differences between their objective values are less than a specific level, then one of them is reevaluated more to increase the level of confidence. Different Pareto sets are assigned different values of minimum confidence level, and since during evolution the rank of solutions in the population changes, implicit dynamism is introduced during evolution as each solution is reevaluated. The proposed method is also applied to two real-world problem related to optimization in engine manufacturing lines. Instead of discarding old evaluations of a solution in a noisy problem, Park and Ryu [2011] proposed an accumulating approach which combines old evaluations with the new reevaluation of a solution to improve the expected value estimation of objective functions.

Another approach taken by some of the proposed methods is to use DM provided information for EMO in the presence of noise. Mehnen et al. [2007] used desirability functions to include both the preferences of DM and to reduce the effect of noise. Instead of performing the search in the objective space, they optimize the expected value of the desirability functions computed from noisy objectives. The final Pareto front is obtained by either applying the inverse desirability functions on the approximated front or computing the objective values of the final Pareto set. The method is used to find the working parameters of an industrial cutting tool. Woźniak [2007] proposed to use a number of reference points provided by DM, each accompanied with a weight vector showing the importance of the objectives, to select fitter solutions in the search process. To maintain the population diversity only one of the solutions in each neighborhood, defined by a predefined neighborhood radius, is selected. The method then is applied for the design of a motor speed controller.

In addition to the noise in objective values, Kaji et al. [2009] studied the effect of noise in constraint functions of a constrained MOP, trying to reduce the number of infeasible solutions selected in the Pareto set. For this purpose they form a history of solutions

Table 11.1: Summary of the EMO methods for noisy environments.

| | Method | Noise Type | Place of Noise |
|---|---|---|---|
| [Teich, 2001] | Probabilistic Dominance | Uniform | Output |
| [Hughes, 2001] | Probabilistic Dominance | Gaussian | Input, Output |
| [Büche et al., 2002] | Lifetime-Based Archiving | Gaussian | Output |
| [Babbar et al., 2003] | Objective Space Neighborhood | Gaussian | Output |
| [Bui et al., 2005a] | Fitness Inheritance in Reevaluations | Gaussian | Output |
| [Fieldsend and Everson, 2005] | Noise Variance Estimation | Gaussian | Output |
| [Basseur and Zitzler, 2006] | Indicator Value Estimation | Uniform | Input, Output |
| [Goh and Tan, 2007] | Necessity and Necessity-Possibility Dominance | Gaussian | Output |
| [Mehnen et al., 2007] | Desirability Functions Optimization | Gaussian | Output |
| [Woźniak, 2007] | Weighted Reference Points | Gaussian | Output |
| [Boonma and Suzuki, 2009] | Confidence-Based Solution Comparison | Gaussian, Uniform, Poisson | Output |
| [Bui et al., 2009] | Search Space Neighborhood | Gaussian | Output |
| [Eskandari and Geiger, 2009] | Stochastic Dominance | Gaussian | Output |
| [Kaji et al., 2009] | Value Estimation with Locally Weighted Ridge Regression | Gaussian | Output, Constraints |
| [Soares et al., 2009] | Worst-Case Analysis | Gaussian | Output |
| [Syberfeldt et al., 2010] | Confidence-Based Reevaluations | Gaussian | Output |
| [Park and Ryu, 2011] | Reevaluation Accumulation | Gaussian | Output |

and estimate the value of objective and constraint functions by a locally weighted ridge regression of second order. The weights are defined using a Gaussian kernel. A safety margin is also introduced to the constraint functions and dynamically adjusted depending on the variance of the estimated constraint values.

Basseur and Zitzler [2006] proposed an indicator-based EMO algorithm for noisy environments, considering the presence of noise in both the objective functions and the input variables. They proposed methods to estimate the expected value of the epsilon indicator for ranking the solutions. A uniform noise is assumed for both inputs and outputs. Boonma and Suzuki [2009] considered different types of distributions like Gaussian, uniform and Poisson for the noise model. Assuming that the quality of each solution is represented with several reevaluations of its objectives, they used a support vector machine to determine the confidence level in these objective values. The coverage metric is used to determine the dominance between two solutions after their values are classified as statistically reliable. The confidence level for accepting the objective values of the solutions is dynamically adjusted during evolution according to the disorder among objective values, which is computed with an entropy-based function.

Table 11.1 summarizes the reviewed algorithms.

## 11.3 $\alpha$-Degree Pareto Dominance

Most of the methods reviewed in the previous section for handling the noise when selecting a subset of solutions, only consider singular objective values. Although these singular values maybe obtained as the result of averaging over several reevaluations, they still do not directly take into account the extent of the noise in the values. We also saw that some of the proposed methods optimize the expected values of the objectives and consider the variance of the objective values, implicitly assuming a type of confidence interval for the objective values. A closely related approach is the possibilistic dominance method [Goh and Tan, 2007], where objective values are considered to be fuzzy numbers. In this section we propose a method for directly comparing any kind of intervals, not necessarily confidence intervals, which also takes into account interval widths (representing the amount of noise). Moreover, the proposed method can deal with MOPs consisting of both singular and interval objectives.

One way to deal with the noise in an MOP when the objective values are represented with intervals is to extend the Pareto dominance definition (Definition 8.1). In the following definitions we assume that the set of objectives $\mathcal{F}$ is partitioned into two disjoint subsets of objectives with singular values $\mathcal{F}_S$, and noisy objectives with interval values $\mathcal{F}_I$, such that $\mathcal{F}_S \cup \mathcal{F}_I = \mathcal{F}$ and $\mathcal{F}_S \cap \mathcal{F}_I = \emptyset$.

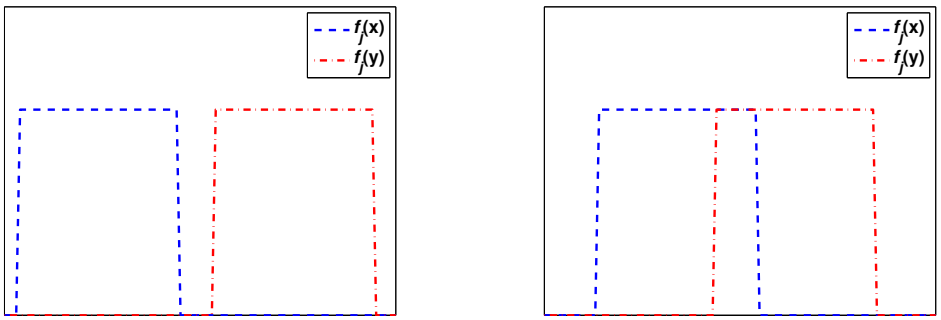
**Definition 11.2. (Extended Pareto Dominance)** *Let $\lfloor f_j(\boldsymbol{x}) \rfloor$ and $\lceil f_j(\boldsymbol{x}) \rceil$ respectively represent the lower and upper bounds of the interval value returned for solution $\boldsymbol{x} \in \mathbb{D}$, by the noisy objective function $f_j \in \mathcal{F}_I$, i.e. $U_j$ lies in the interval $\Big[\lfloor f_j(\boldsymbol{x}) \rfloor, \lceil f_j(\boldsymbol{x}) \rceil\Big]$. Then, solution $\boldsymbol{x}$ is said to strictly dominate solution $\boldsymbol{y}$, denoted as $\boldsymbol{x} \prec_e \boldsymbol{y}$, if and only if:*

1. $\forall f_j \in \mathcal{F}_S \quad f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{y})$, *and*
2. $\forall f_j \in \mathcal{F}_I \quad \lceil f_j(\boldsymbol{x}) \rceil \leq \lfloor f_j(\boldsymbol{y}) \rfloor$, *and*
3. $\Big(\exists f_k \in \mathcal{F}_S \quad f_k(\boldsymbol{x}) < f_k(\boldsymbol{y}) \quad \textit{or} \quad \exists f_k \in \mathcal{F}_I \quad \lceil f_k(\boldsymbol{x}) \rceil < \lfloor f_k(\boldsymbol{y}) \rfloor\Big)$.

This definition treats interval values similar to the way it considers singular values and only allows a solution to dominate other solutions if its corresponding interval values, returned by noisy objective functions, are *strictly* better than those of other solutions. In real-world noisy MOPs, such a requirement is hardly satisfied. A further extension is to relax this strict requirement and allow the intervals to have some degree of overlapping.

**Definition 11.3. ($\alpha$-Degree Pareto Dominance)** *Assume the same notations as those of Definition 11.2. Then, solution $\boldsymbol{x}$ is said to dominate another solution $\boldsymbol{y}$ with a degree $\alpha \in (0, 1]$, denoted as $\boldsymbol{x} \prec_\alpha \boldsymbol{y}$, if and only if:*

1. $\forall f_j \in \mathcal{F}_S \quad f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{y})$, *and*
2. $\forall f_j \in \mathcal{F}_I \quad \deg_j(\boldsymbol{x}, \boldsymbol{y}) \geq \alpha$, *and*
3. $\Big(\exists f_k \in \mathcal{F}_S \quad f_k(\boldsymbol{x}) < f_k(\boldsymbol{y}) \quad \vee \quad \exists f_k \in \mathcal{F}_I \quad \deg_k(\boldsymbol{x}, \boldsymbol{y}) > \alpha\Big)$,




(a) $\deg_j(\boldsymbol{x}, \boldsymbol{y}) = 1$ (b) $\deg_j(\boldsymbol{x}, \boldsymbol{y}) = 0.75$

(c) $\deg_j(\boldsymbol{x}, \boldsymbol{y}) = 0.5$ (d) $\deg_j(\boldsymbol{x}, \boldsymbol{y}) = 0$

Figure 11.1: Examples of interval values and the resulting degrees of dominance.

*where* $\deg_j(\cdot, \cdot)$ *gives the degree that a solution dominates another with respect to the noisy objective function* $f_j \in \mathcal{F}_I$

$$\deg_j(\boldsymbol{x}, \boldsymbol{y}) = \min\left\{1, \max\{0, \frac{\lfloor f_j(\boldsymbol{y}) \rfloor - \lfloor f_j(\boldsymbol{x}) \rfloor}{\lceil f_j(\boldsymbol{x}) \rceil - \lfloor f_j(\boldsymbol{x}) \rfloor}\}\right\}. \tag{11.2}$$

Intuitively, $\deg_j(\boldsymbol{x}, \boldsymbol{y})$ computes the percentage of the interval obtained for solution $\boldsymbol{x}$ that is not overlapped by the interval obtained for solution $\boldsymbol{y}$ in objective $f_j \in \mathcal{F}_I$. Thus, only the segment in the interval obtained for solution $\boldsymbol{x}$ that is better than the best point in the interval obtained for solution $\boldsymbol{y}$ (i.e. its lower bound, when minimizing objectives) is taken into account. If $\alpha = 1$, then Definition 11.3 is reduced to Definition 11.2. Definition 11.3 allows a solution to dominate other solutions when its corresponding interval values are partially better than those of other solutions.

### 11.3.1 Discussion

Figure 11.1 shows some examples of possible placements of two intervals and the corresponding values of the $\deg_j(\cdot, \cdot)$ function. With higher values of $\alpha$, a solution can only dominate other solutions if a major part of its corresponding interval values are better than the best points of the interval values corresponding to other solutions. Thus, higher values of $\alpha$ place a stricter condition for accepting a solution as non-dominated.

**Proposition 11.1.** *$\alpha$-degree Pareto dominance defines a partial relation, i.e. with irreflexivity, antisymmetry and transitivy properties, on the space of candidate solutions.*

*Proof.* Since no $\boldsymbol{x} \in \mathbb{D}$ exists that satisfies $\boldsymbol{x} \prec_\alpha \boldsymbol{x}$ for any $\alpha \in (0, 1]$, the relation is irreflexive. To see the antisymmetry of the relation, assume $\boldsymbol{x} \prec_\alpha \boldsymbol{y}$. If $\exists f_j \in \mathcal{F}_S$ such that $f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$ then $\boldsymbol{y} \not\prec_\alpha \boldsymbol{x}$. If $\exists f_j \in \mathcal{F}_I$ such that $\deg_j(\boldsymbol{x}, \boldsymbol{y}) > \alpha$, then by definition of $\deg_j(\cdot, \cdot)$ in Equation (11.2) if $\deg_j(\boldsymbol{x}, \boldsymbol{y}) > 0$ then $\deg_j(\boldsymbol{y}, \boldsymbol{x}) = 0$ and therefore $\boldsymbol{y} \not\prec_\alpha \boldsymbol{x}$ for any $\alpha \in (0, 1]$.

For transitivity, assume that $\boldsymbol{x} \prec_\alpha \boldsymbol{y}$ and $\boldsymbol{y} \prec_\alpha \boldsymbol{z}$. Then we have $\forall f_j \in \mathcal{F}_S$, $f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{y})$ and $f_j(\boldsymbol{y}) \leq f_j(\boldsymbol{z})$. Therefore, $f_j(\boldsymbol{x}) \leq f_j(\boldsymbol{z})$, $\forall f_j \in \mathcal{F}_S$. Similarly, we have $\deg_j(\boldsymbol{x}, \boldsymbol{z}) \geq \alpha$, $\forall f_j \in \mathcal{F}_I$, because if $\forall f_j \in \mathcal{F}_I$ such that $\lfloor f_j(\boldsymbol{x}) \rfloor < \lfloor f_j(\boldsymbol{y}) \rfloor$ and $\lfloor f_j(\boldsymbol{y}) \rfloor < \lfloor f_j(\boldsymbol{z}) \rfloor$, then we have $\lfloor f_j(\boldsymbol{x}) \rfloor < \lfloor f_j(\boldsymbol{z}) \rfloor$, $\forall f_j \in \mathcal{F}_I$. Now, given either $\exists f_j \in \mathcal{F}_S$ such that $f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$ or $\exists f_j \in \mathcal{F}_I$ such that $\deg_j(\boldsymbol{x}, \boldsymbol{y}) > \alpha$ (similarly $\exists f_j \in \mathcal{F}_S$ such that $f_j(\boldsymbol{y}) < f_j(\boldsymbol{z})$ or $\exists f_j \in \mathcal{F}_I$ such that $\deg_j(\boldsymbol{y}, \boldsymbol{z}) > \alpha$) we can respectively conclude either $\exists f_j \in \mathcal{F}_S$ such that $f_j(\boldsymbol{x}) < f_j(\boldsymbol{z})$ or $\exists f_j \in \mathcal{F}_I$ such that $\deg_j(\boldsymbol{x}, \boldsymbol{z}) > \alpha$, which completes the proof of proposition. ■

The partial relation defined by $\alpha$-degree Pareto dominance allows to readily adopt the terms of $\alpha$-degree Pareto optimal solution, $\alpha$-degree Pareto optimal set, $\alpha$-degree Pareto optimal front and $\alpha$-degree Pareto non-dominated set for noisy MOPs, in the same way they are defined using the conventional Pareto dominance relation. The $\alpha$-degree Pareto dominance relation also has interesting properties when different confidence levels are considered for the objective values. To see this, lets assume that the values returned by noisy objective functions for solution $\boldsymbol{x}$ are confidence intervals given in the form of

$$\Big(\hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big) - \varepsilon_\gamma\big(f_j(\boldsymbol{x})\big), \hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big) + \varepsilon_\gamma\big(f_j(\boldsymbol{x})\big)\Big),$$

where $\hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big)$ represents an estimation of the expected value of the $j$th objective function for solution $\boldsymbol{x}$ and

$$\varepsilon_\gamma\big(f_j(\boldsymbol{x})\big) = z_{\frac{1-\gamma}{2}} \hat{\sigma}\big(f_j(\boldsymbol{x})\big)$$

is the half-width [Eskandari and Geiger, 2009] of the confidence interval computed according to a specific confidence level $\gamma$. Here, $z_{(1-\gamma)/2}$ denotes the value for which $\Phi(Z > z_{(1-\gamma)/2}) = \frac{1-\gamma}{2}$, where $\Phi(Z)$ is the cumulative standard Gaussian (or t-student) distribution. $\hat{\sigma}\big(f_j(\boldsymbol{x})\big)$ is the estimation for the standard deviation of mean objective value. With confidence intervals, the definition of $\deg_j(\boldsymbol{x}, \boldsymbol{y})$ for a noisy objective $f_j \in \mathcal{F}_I$ can be rewritten as

$$\deg_j(\boldsymbol{x}, \boldsymbol{y}) = \min\big\{1, \max\{0, \frac{\Big(\hat{\mathbb{E}}\big(f_j(\boldsymbol{y})\big) - \hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big)\Big) - \Big(\varepsilon_\gamma\big(f_j(\boldsymbol{y})\big) - \varepsilon_\gamma\big(f_j(\boldsymbol{x})\big)\Big)}{2\varepsilon_\gamma\big(f_j(\boldsymbol{x})\big)}\}\big\}. \tag{11.3}$$

According to this definition, the degree that a solution dominates another solution with respect to a noisy objective $f_j \in \mathcal{F}_I$ is determined by the differences in both expected values and half-widths. The following propositions show how the change in the confidence level $\gamma$ or dominance degree $\alpha$ affects the $\alpha$-degree Pareto dominance relation.

**Proposition 11.2.** *Any reduction in the confidence level of the interval values given by noisy objective functions in $\mathcal{F}_I$ does not affect $\boldsymbol{x} \prec_\alpha \boldsymbol{y}$, if for every objective function $f_j \in \mathcal{F}_I$ we have $\hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big) < \hat{\mathbb{E}}\big(f_j(\boldsymbol{y})\big)$.*

*Proof.* We only consider noisy objectives in $\mathcal{F}_I$ here because the change in confidence level does not affect singular values. Let $\boldsymbol{x} \prec_\alpha^\gamma \boldsymbol{y}$ denote that solution $\boldsymbol{x}$ dominates solution $\boldsymbol{y}$ with a degree $\alpha$, when the confidence level in the interval value of all noisy objectives in $\mathcal{F}_I$ is at least $\gamma$.

Assuming $\boldsymbol{x} \prec_\alpha^{\gamma_1} \boldsymbol{y}$ implies that

$$\begin{aligned} \forall f_j \in \mathcal{F}_I, \quad & \frac{\Big(\hat{\mathbb{E}}\big(f_j(\boldsymbol{y})\big) - \hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big)\Big) - \Big(\varepsilon_{\gamma_1}\big(f_j(\boldsymbol{y})\big) - \varepsilon_{\gamma_1}\big(f_j(\boldsymbol{x})\big)\Big)}{2\varepsilon_{\gamma_1}\big(f_j(\boldsymbol{x})\big)} \geq \alpha \\ \Rightarrow \quad & \frac{\hat{\mathbb{E}}\big(f_j(\boldsymbol{y})\big) - \hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big)}{2z_{\frac{1-\gamma_1}{2}}\hat{\sigma}\big(f_j(\boldsymbol{x})\big)} - \frac{\hat{\sigma}\big(f_j(\boldsymbol{y})\big) - \hat{\sigma}\big(f_j(\boldsymbol{x})\big)}{2\hat{\sigma}\big(f_j(\boldsymbol{x})\big)} \geq \alpha. \end{aligned}$$

Changing the confidence level does not influence the second quotient above. The numerator of the first quotient is positive because of the assumption of the proposition and, since $z_{\frac{1-\gamma_1}{2}} > z_{\frac{1-\gamma_2}{2}}$, $\forall \gamma_2 < \gamma_1$, this quotient will become larger if confidence level decreases to $\gamma_2$. Therefore, for all $f_j \in \mathcal{F}_I$

$$\begin{aligned} & \frac{\Big(\hat{\mathbb{E}}\big(f_j(\boldsymbol{y})\big) - \hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big)\Big) - \Big(\varepsilon_{\gamma_2}\big(f_j(\boldsymbol{y})\big) - \varepsilon_{\gamma_2}\big(f_j(\boldsymbol{x})\big)\Big)}{2\varepsilon_{\gamma_2}\big(f_j(\boldsymbol{x})\big)} > \\ & \qquad \frac{\Big(\hat{\mathbb{E}}\big(f_j(\boldsymbol{y})\big) - \hat{\mathbb{E}}\big(f_j(\boldsymbol{x})\big)\Big) - \Big(\varepsilon_{\gamma_1}\big(f_j(\boldsymbol{y})\big) - \varepsilon_{\gamma_1}\big(f_j(\boldsymbol{x})\big)\Big)}{2\varepsilon_{\gamma_1}\big(f_j(\boldsymbol{x})\big)} \geq \alpha, \end{aligned}$$

which means $\boldsymbol{x} \prec_\alpha^{\gamma_2} \boldsymbol{y}$. ■

Now, let $\mathcal{PS}_\alpha^\gamma$ denote the $\alpha$-degree Pareto optimal set imposed by $\alpha$-degree Pareto dominance relation $\prec_\alpha^\gamma$, when the confidence level in the interval values returned by the noisy objectives is at least $\gamma$.

**Corollary 11.1.** *If $\gamma_2 < \gamma_1$, then $\mathcal{PS}_\alpha^{\gamma_2} \subset \mathcal{PS}_\alpha^{\gamma_1}$.*

*Proof.* Assume that there exists a solution $\boldsymbol{y}$ such that $\boldsymbol{y} \in \mathcal{PS}_\alpha^{\gamma_2}$ and $\boldsymbol{y} \notin \mathcal{PS}_\alpha^{\gamma_1}$. This means that there exists a solution $\boldsymbol{x} \in \mathcal{PS}_\alpha^{\gamma_1}$ such that $\boldsymbol{x} \prec_\alpha^{\gamma_1} \boldsymbol{y}$. By Proposition 11.2 we know that this implies $\boldsymbol{x} \prec_\alpha^{\gamma_2} \boldsymbol{y}$. But the last relation means that $\boldsymbol{y} \notin \mathcal{PS}_\alpha^{\gamma_2}$ which contradicts the hypothesis. ■

**Proposition 11.3.** *The $\alpha$-degree Pareto dominance relation between solutions is preserved when the dominance degree $\alpha$ is decreased.*

*Proof.* It is trivial since if $\alpha_2 < \alpha_1$ then $\deg_j(\boldsymbol{x}, \boldsymbol{y}) \geq \alpha_1$ implies that $\deg_j(\boldsymbol{x}, \boldsymbol{y}) > \alpha_2$, for any two solutions $\boldsymbol{x}$, $\boldsymbol{y}$ and every objective function $f_j \in \mathcal{F}_I$. ■

**Corollary 11.2.** *If $\alpha_2 < \alpha_1$, then $\mathcal{PS}_{\alpha_2}^\gamma \subset \mathcal{PS}_{\alpha_1}^\gamma$.*

*Proof.* The same as the proof for Corollary 11.1. ■

## 11.4 Noisy Multi-Objective Optimization with MBN-EDA

We want to study the behavior of MBN-EDA in multi-objective optimization with the presence of noise in objective values. In this section, we describe how $\alpha$-degree Pareto dominance is used for ranking the solutions, and how noisy objective values can affect joint probabilistic modeling in MBN-EDA.

### 11.4.1 $\alpha$-Degree Non-Dominated Sorting

We use $\alpha$-degree Pareto dominance relation to develop a version of the well-known non-dominated sorting algorithm [Deb et al., 2002a] that can be applied for solution ranking in noisy environments, when the objectives are given as intervals. The main steps of this ranking method are shown in Algorithm 11.1.

**Inputs:**
A set of solutions $P$
A dominance degree $\alpha$
$r \leftarrow 0$
**while** there are more solutions in $P$ **do**
  $r \leftarrow r + 1$
  $S_r \leftarrow \alpha$-degree Pareto non-dominated solutions of $P$
  $P \leftarrow P \setminus S_r$
**end while**
**for all** $i \in \{1, \ldots, r\}$ **do**
  $D_i \leftarrow$ Crowding distances of solutions in $S_i$
  $S_i \leftarrow$ Reorder the solutions in $S_i$ in decreasing value of $D_i$
**end for**
**Output:** $\{S_1, \ldots, S_r\}$

Algorithm 11.1: The $\alpha$-Degree Non-Dominated Sorting Algorithm.

The algorithm first orders the solutions into a number of $\alpha$-degree Pareto non-dominated sets, by comparing the solutions of the population with the $\alpha$-degree Pareto dominance relation. Then, within each $\alpha$-degree non-dominated set, the solutions are ordered according to their crowding distances in the objective space, which reflects how scattered is each solution in the objective space with respect to the other solutions in the same $\alpha$-degree non-dominated set. In practice, when this ranking method is used with techniques like truncation selection, the crowding distance is computed only for the solutions of the $\alpha$-degree non-dominated set which cannot be added completely to the set of selected solutions.

To compute the crowding distances, the solutions are ordered with respect to each of the objectives individually, and the crowding distance of the solutions with minimum and maximum value for each objective is set to a large number, indicating that these solutions are well scattered with respect to that objective. The crowding distance of other in-between solutions is computed by summing up the normalized distances of each solution to its preceding and succeeding solutions in each objective with respect to the ordering. To order the solutions in each objective dimension, the quick sort algorithm is adapted to work with interval values, comparing two solutions by checking whether

$\deg_j(\cdot,\cdot) > 0$. The value of $\deg_j(\cdot,\cdot)$ function is also used as a normalized estimation of the distances between interval values of the solutions in each objective.

### 11.4.2 Joint Model Learning

The presence of noise in objective values, represented with intervals, means that in the joint vector $(X_1,\ldots,X_n,Q_1,\ldots,Q_m)$, a subset of objective variables $\boldsymbol{Q}_I$, where $I$ is the set of objective function indices in $\mathcal{F}_I$, take on interval values. Thus, not all of the values in the joint variable-objective dataset provided for probabilistic modeling are scalar values.

There are some studies in the area of imprecise probabilities and credal sets for estimating a probability distribution for a vector of random variables, when some of these variables take non-scalar values (e.g. set of values or intervals). When BNs are used to encode this kind of probability distributions they are called credal networks [Corani et al., 2010]. Because of the inherent complexity of this type of models, the methods proposed for their learning and inference are usually very time consuming.

To be able to iteratively perform the joint modeling of variables and objectives in each generation of MBN-EDA, we have used a simple approach for learning a probabilistic model in the presence of noisy objectives with interval values. Before learning the joint model, all of the interval values are replaced by representative scalar values. For example, when the values returned by noisy objective functions are considered to be confidence intervals, a good representative value for each interval is its estimated expected value, $\hat{\mathbb{E}}(Q_j)$. Once the dataset is scalarized, the methods described in the previous chapters for estimating and sampling the joint model can be applied. This scalarization is also justified when taking into account the fact that in joint modeling, objective values are only used to obtain an approximation of the influence of objectives on each other and on the variables.

## 11.5 Experiments

The optimization performance of MBN-EDA in noisy multi-objective optimization is tested on a set of noisy MOPs to examine its behavior and ability in handling the noise in objective values. Five of the previously used WFG problems, namely WFG1, WFG2, WFG3, WFG7 and WFG9, with three objectives and 10 variables are selected for the experiments in this section. Noise is introduced to the output of all three objective functions in each MOP, resulting in a confidence interval for each solution in the search space. Further details of the noise model and experimental design are explained in the following sections.

### 11.5.1 Noise Model

In Section 11.2, we saw that many of the works in the literature simulate the noise in objective functions of an MOP with an additive zero-mean Gaussian distribution:

$$f_j(\boldsymbol{x}_i) + \mathcal{N}(0,\sigma_n^2), \tag{11.4}$$

where $f_j(\boldsymbol{x}_i)$ is the true value of the $j$th objective function for solution $\boldsymbol{x}_i$, and $\sigma_n$ controls the level of noise introduced to the objective functions of MOP, which often varies in the

range $[0.01, 1]$. Arnold and Beyer [2006] have also studied several other noise models and their influence on the performance of EAs.

As explained earlier, the noise model in Equation (11.4), which is used in many of the previous works, results in singular values for the objective functions. To obtain interval values for noisy objectives, as it is assumed in this paper, we can draw several random values $\xi_k$ from the noise model and use these values to compute a confidence interval, based on a specific confidence level $\gamma$, for each objective function $f_j$ and each solution $\boldsymbol{x}_i$:

$$\big(f_j(\boldsymbol{x}_i) + \bar{\xi} \ \pm \ z_{\frac{1-\gamma}{2}} \frac{s(\xi)}{\sqrt{K}}\big), \tag{11.5}$$

where $K$ is the number of random values drawn from the noise model, and $\bar{\xi}$ and $s(\xi)$ are respectively their mean and standard deviation. However, in practice, using such a method to generate interval values for the noisy objective functions imposes a significant computational overhead to the solution evaluation phase, especially for larger values of $K$. Moreover, this method can reduce the stochasticity of the interval values generated for the objective functions, as for example with a Gaussian noise model when $K$ increases, the mean and standard deviation of the sampled random values tend to the corresponding parameters of the noise model.

To have a somewhat similar randomness in the generated interval values as in the singular values obtained from Equation (11.4), we draw two random values $\xi_m$ and $\xi_s$ from a Gaussian noise model to compute a confidence interval for each objective function $f_j$ and each solution $\boldsymbol{x}_i$:

$$\big(f_j(\boldsymbol{x}_i) + \xi_m \ \pm \ z_{\frac{1-\gamma}{2}} \xi_s\big). \tag{11.6}$$

These two random values can be generated from two different Gaussian distributions. However, as it is explained in [Eskandari and Geiger, 2009], it is more reasonable that the level of noise in $\xi_m$ and $\xi_s$ increase and decrease correspondingly. In this study we use similar Gaussian distributions for sampling these two values.

### 11.5.2 Experimental Design

In the experiments performed in this section, we study and compare two solution ranking methods when the noisy objective values are given as intervals. The first method is the proposed $\alpha$-degree non-dominated sorting algorithm (Algorithm 11.1), hereafter referred to as degree ranking (DR), and tested with three different dominance degrees: $\alpha \in \{0.1, 0.5, 0.9\}$. As the second method, we adopt probabilistic ranking (PR) [Hughes, 2001], based on dominance probability (Definition 11.1), which is often used as a reference solution ranking method in the studies on multi-objective optimization in noisy environments, and ranks each solution $\boldsymbol{x}_i$ as follows:

$$\text{rank}_{PR}(\boldsymbol{x}_i) = \sum_{k=1}^{N} P(\boldsymbol{x}_k \prec \boldsymbol{x}_i) + \frac{1}{2} \sum_{k=1}^{N} P(\boldsymbol{x}_k \equiv \boldsymbol{x}_i) - \frac{1}{2}, \tag{11.7}$$

where

$$P(\boldsymbol{x} \equiv \boldsymbol{y}) = 1 - P(\boldsymbol{x} \prec \boldsymbol{y}) - P(\boldsymbol{x} \succ \boldsymbol{y})$$

represents the probability that neither $\boldsymbol{x}$ nor $\boldsymbol{y}$ dominate each other, and $N$ is the population size. The last term in Equation (11.7) is subtracted so that

$$\sum_{k=1}^{N} \text{rank}_{PR}(\boldsymbol{x}_k) = \frac{N(N-1)}{2}.$$

This ranking method defines a total ordering between the solutions of population, allowing to sort the solutions based on the ranks assigned by $\text{rank}_{PR}(\cdot)$.

To simplify dominance probability computation in multi-objective case, which involves multivariate integration, the objective functions are assumed to be statistically independent [Teich, 2001; Hughes, 2001], and therefore the dominance probability in Equation (11.1) is approximated as:

$$P(\boldsymbol{x} \prec \boldsymbol{y}) = \prod_{j=1}^{m} P(U_j < V_j), \tag{11.8}$$

where $U_j = f_j(\boldsymbol{x})$ and $V_j = f_j(\boldsymbol{y})$ are the random variables representing the $j$th objective values returned for two solutions $\boldsymbol{x}$ and $\boldsymbol{y}$. Moreover, when noise is modeled as a Gaussian distribution, Hughes [2001] proposed an approximation of the univariate integration required in the computation of $P(U_j < V_j)$ to further reduce the computational complexity of the overall solution ranking:

$$P(U_j < V_j) = \frac{1}{2} - \frac{1}{2}\text{erf}\left(\frac{U_j - V_j}{\sigma_V\sqrt{2 + 2(\frac{\sigma_U}{\sigma_V})^2}}\right) \approx \frac{1}{2} - \frac{1}{2}\tanh\left(\frac{U_j - V_j}{0.8\sigma_V\sqrt{2 + 2(\frac{\sigma_U}{\sigma_V})^2}}\right), \tag{11.9}$$

where $\sigma_U$ and $\sigma_V$ are the standard deviations of the random variables $U_j = f_j(\boldsymbol{x})$ and $V_j = f_j(\boldsymbol{y})$, respectively. When objective values are given as confidence intervals, the estimated expected values and half-widths are used as approximations for $U_j$ and $V_j$ and their standard deviations in the computation of dominance probability with Equation (11.9).

These ranking methods are used in MBN-EDA to rank and select a subset of solutions for offspring generation with probabilistic modeling. A deterministic binary tournament selection with replacement strategy is employed to select 50% of the population solutions. For better comparison, we have also included a standard EMO algorithm based on NSGA-II [Deb et al., 2002a] in the experimentation. We refer to this algorithm simply as EA in contrast to MBN-EDA which will be indicated as EDA in the experimental results. Both of the algorithms use an elitist replacement strategy, a population of 50 solutions and stop after 300 generations. The initial population is generated randomly in both algorithms by uniform sampling from the variables domain.

For each of the five MOPs tested in our experiments, we have studied three different levels of noise and confidence: $\sigma_n \in \{0.01, 0.1, 1\}$ and $\gamma \in \{0.90, 0.95, 0.99\}$. Therefore, all together, there are $5 \times 3 \times 3 \times 2 \times 4 = 360$ different possible combinations for the experiments. For each combination, 10 independent runs are performed. To evaluate the results of experiments, we have used the hypervolume [Zitzler and Thiele, 1999] and inverted generational distance (IGD) [Coello Coello and Cortés, 2005] quality indicators. For hypervolume indicator, a value of 100 is used for all objectives in the nadir point. For IGD computation, a sampling of $10,000$ points is used for representing the Pareto

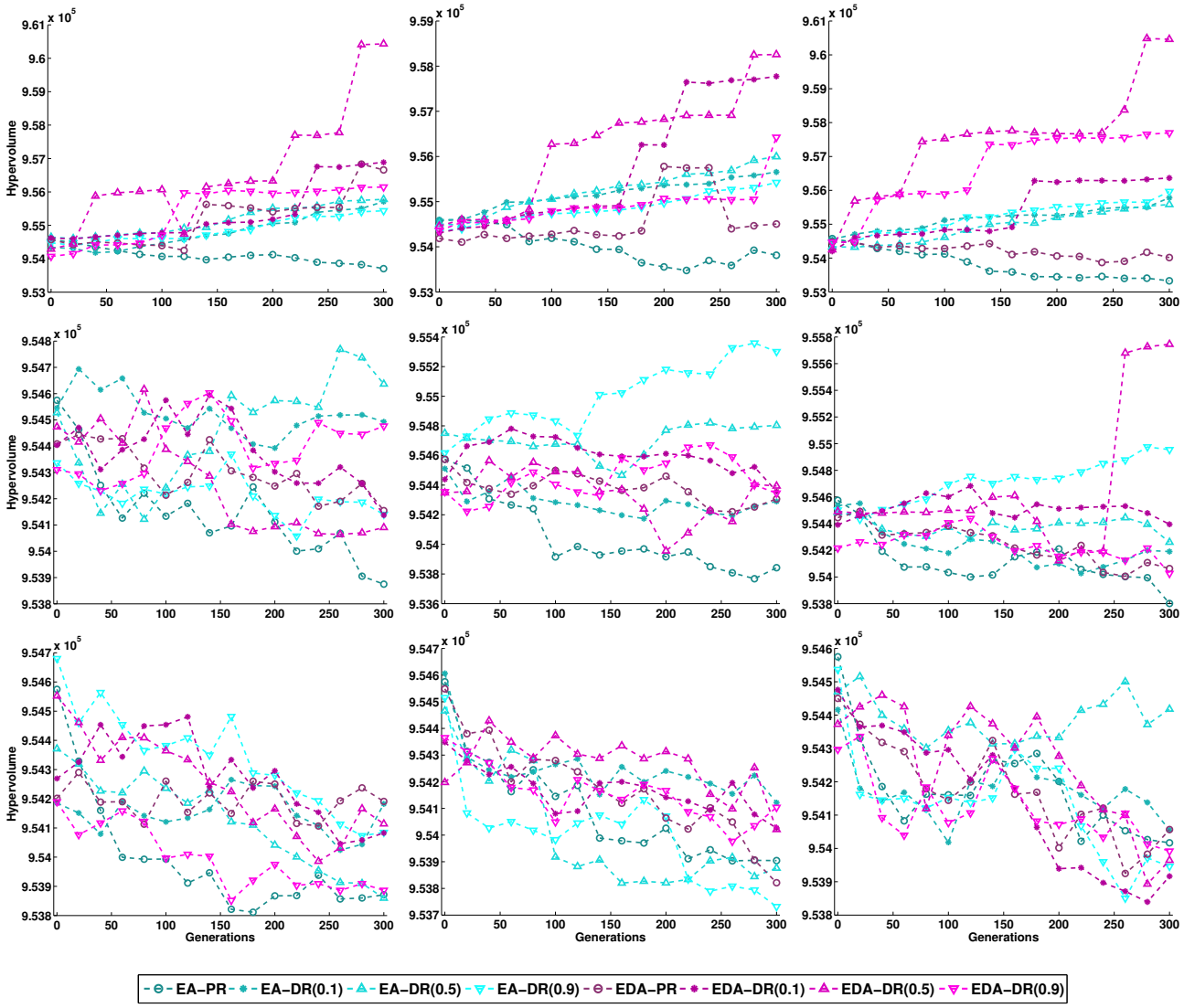



Figure 11.2: Average Hypervolume of the Pareto fronts obtained for WFG1 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be maximized.

optimal front of the MOPs. In the evaluation process, the Pareto set approximated by each algorithm is used to obtain a noiseless Pareto front which is then used to compute these indicators. Therefore the noisy Pareto fronts approximated by the algorithms might be (and usually are) better than the noiseless Pareto fronts, especially for higher levels of noise.

### 11.5.3 Results

Figures 11.2–11.11 show the average hypervolume and IGD indicator values for the Pareto solutions obtained along the evolution path of different combinations of the two algorithmic frameworks and the studied ranking methods, for the tested MOPs. The figures are organized so that rows from top to bottom and columns from left to right respec-

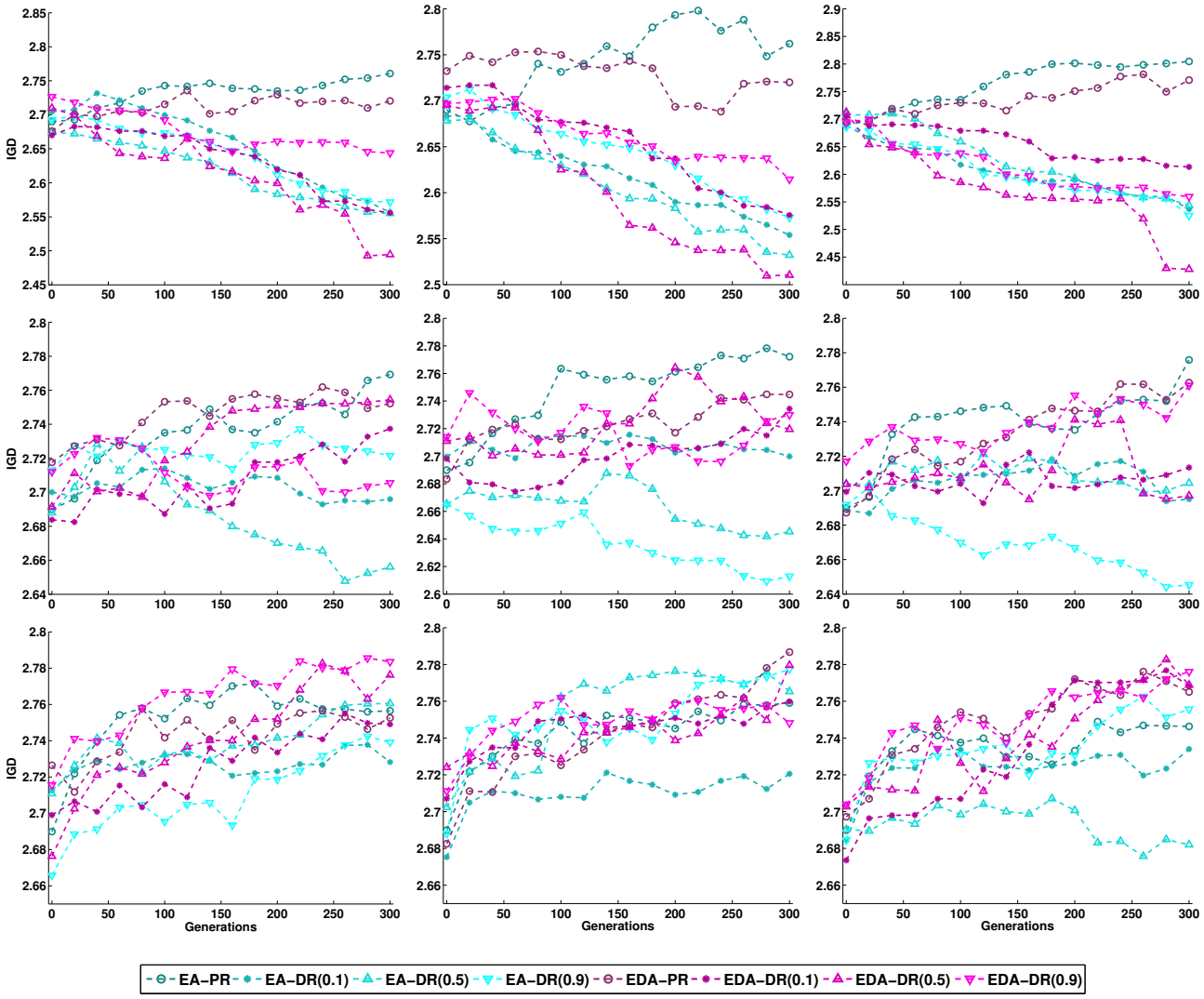



Figure 11.3: Average IGD of the Pareto fronts obtained for WFG1 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be minimized.

tively show increasing levels of noise and confidence in the interval values of the objective functions.

With the increase in the noise level, the change in the position of intervals with respect to the original noiseless values of the objective function will be higher, and also the length of the intervals increases. Besides, increasing the confidence level results in wider intervals. Therefore, in each of the figures as we move from top to bottom and from left to right, the value of objective functions are more distorted. Because of this, we focus the analysis of the results on the common patterns for the average behavior of the algorithms on different instances (with different levels of noise and confidence) of the tested problems.

In the objective functions of WFG1 problem, the input variables are greatly biased, making an even exploration of the search space very difficult, especially in the presence of noise. According to the employed quality indicators, the solution ranking provided by

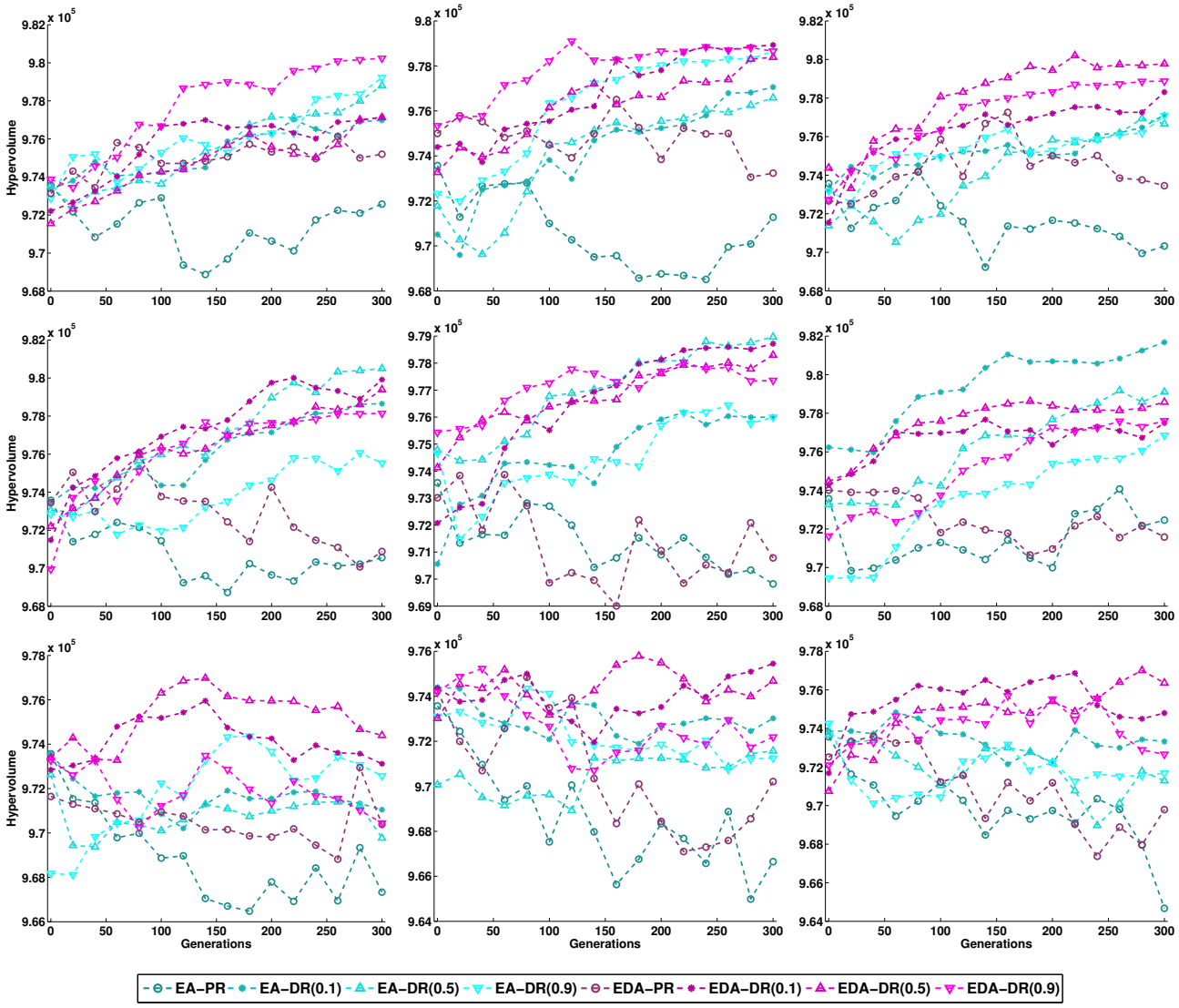



Figure 11.4: Average Hypervolume of the Pareto fronts obtained for WFG2 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be maximized.

DR method (with different degrees of dominance) allows to obtain better approximations of Pareto set on most of the problem instances. As the noise level increases, the quality of the Pareto sets obtained using each of the two ranking methods become closer. The increase in the noise level also blurs the differences in the performance of the algorithms using different degrees of dominance for DR method. Moreover, the increase in the confidence level has less influence on DR method when using a high dominance degree like 0.9.

A closer inspection of the Pareto sets obtained for WFG1 problem shows that the algorithms are not able to obtain a well covering approximation of the Pareto optimal front. Actually, our analysis of the Pareto fronts approximated by EA shows (the results are not included here for brevity) that their spread constantly shrinks along the evolution path for smaller noise levels, as a result of the high bias in the objectives. However,

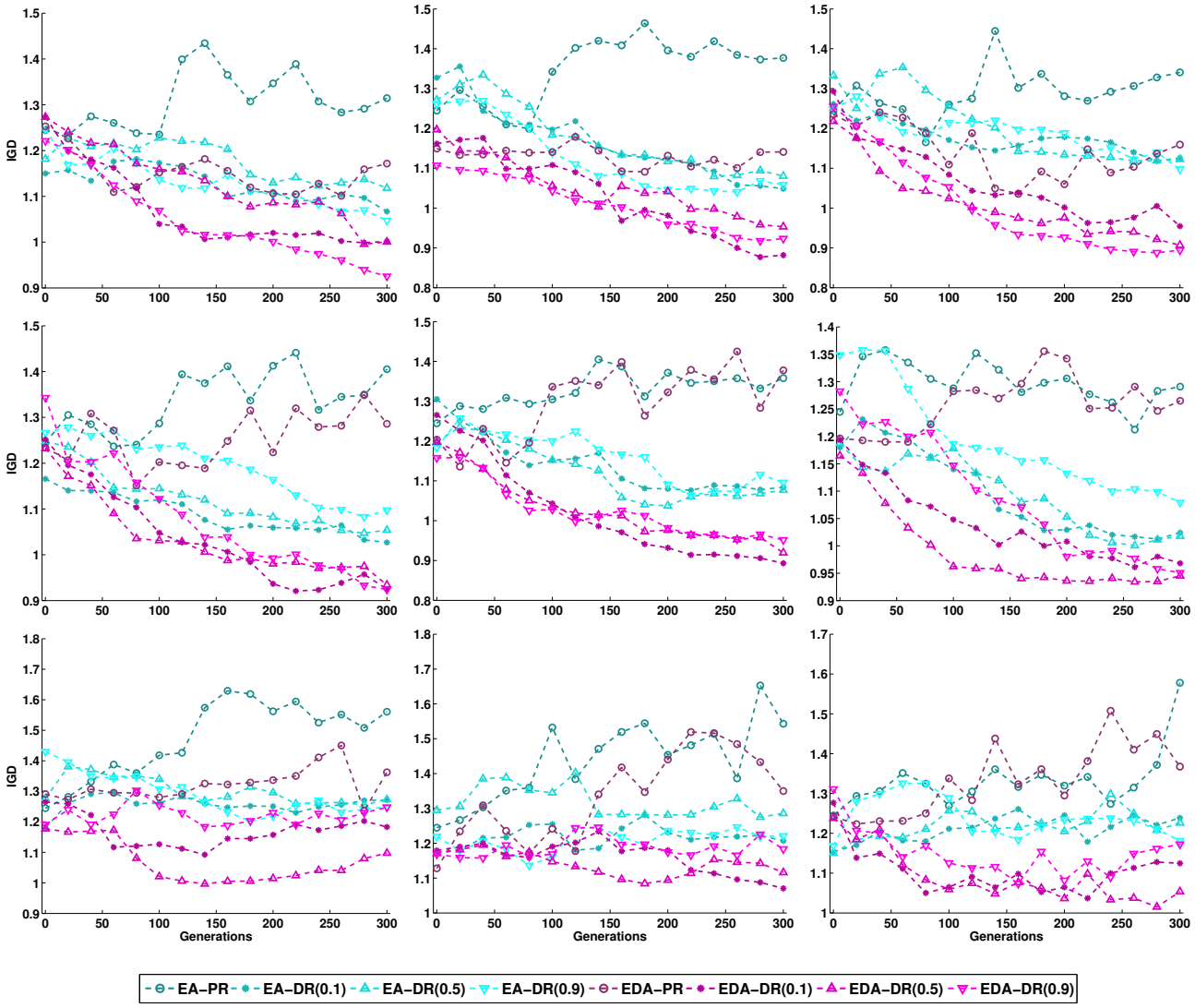



Figure 11.5: Average IGD of the Pareto fronts obtained for WFG2 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be minimized.

the joint modeling adopted in EDA allows a better exploration of this problem's search space, at least for lower noise levels, which is also reflected by the values obtained for hypervolume and IGD indicators.

The results obtained for WFG2 problem (Figures 11.4 and 11.5) also show that the Pareto sets approximated with DR method are superior to those obtained using probabilistic ranking method. When a higher degree of dominance is used for the DR method, the performance of the algorithms with respect to the different quality indicators are more sensitive to the increase in the length of the intervals, i.e. when increasing the level of noise or confidence. In the last objective of this problem, the variables are considered to be correlated and therefore, employing model learning to find out this relationships helps EDA to obtain better Pareto set approximations under different levels of noise and confidence.

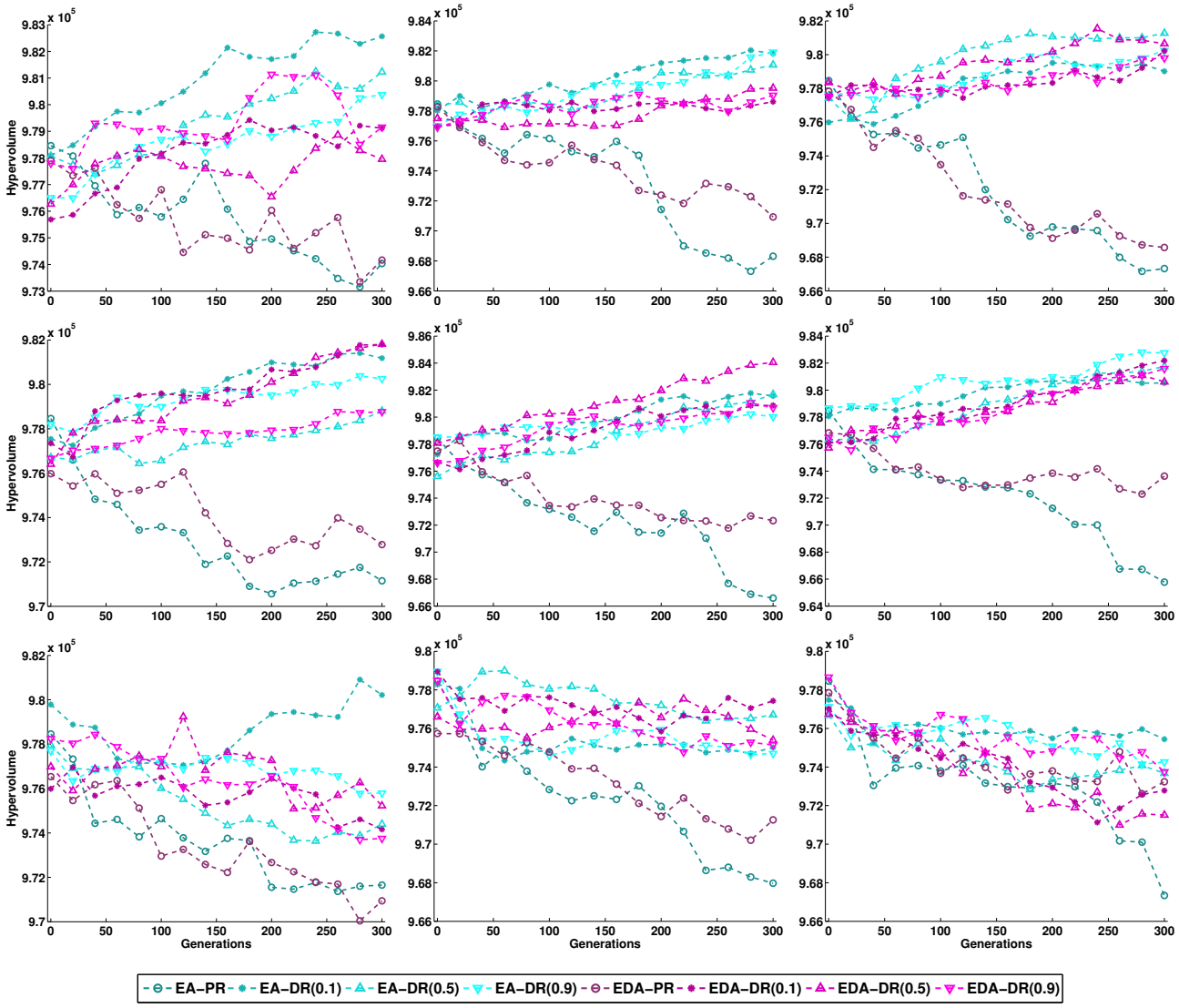



Figure 11.6: Average Hypervolume of the Pareto fronts obtained for WFG3 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be maximized.

Very similar results to those of the previous problem are obtained for WFG3 problem. The difference in the average performance of the algorithms using PR and DR methods are clearer on this problem, and it can be seen that the solution ranking provided by PR method completely misguides the algorithms during evolution for approaching the Pareto optimal set according to the quality indicators, even with small levels of noise and confidence. The DR method using smaller dominance degrees results in relatively better algorithm performance on different levels of noise and confidence with respect to different quality indicators.

The values of different indicators obtained for the Pareto sets approximated by EA and EDA are not consistent for this problem. According to hypervolume indicator, EA obtains relatively better results on WFG3 problem, whereas with respect to IGD indicator the results obtained by EDA outperform those of EA especially with larger noise

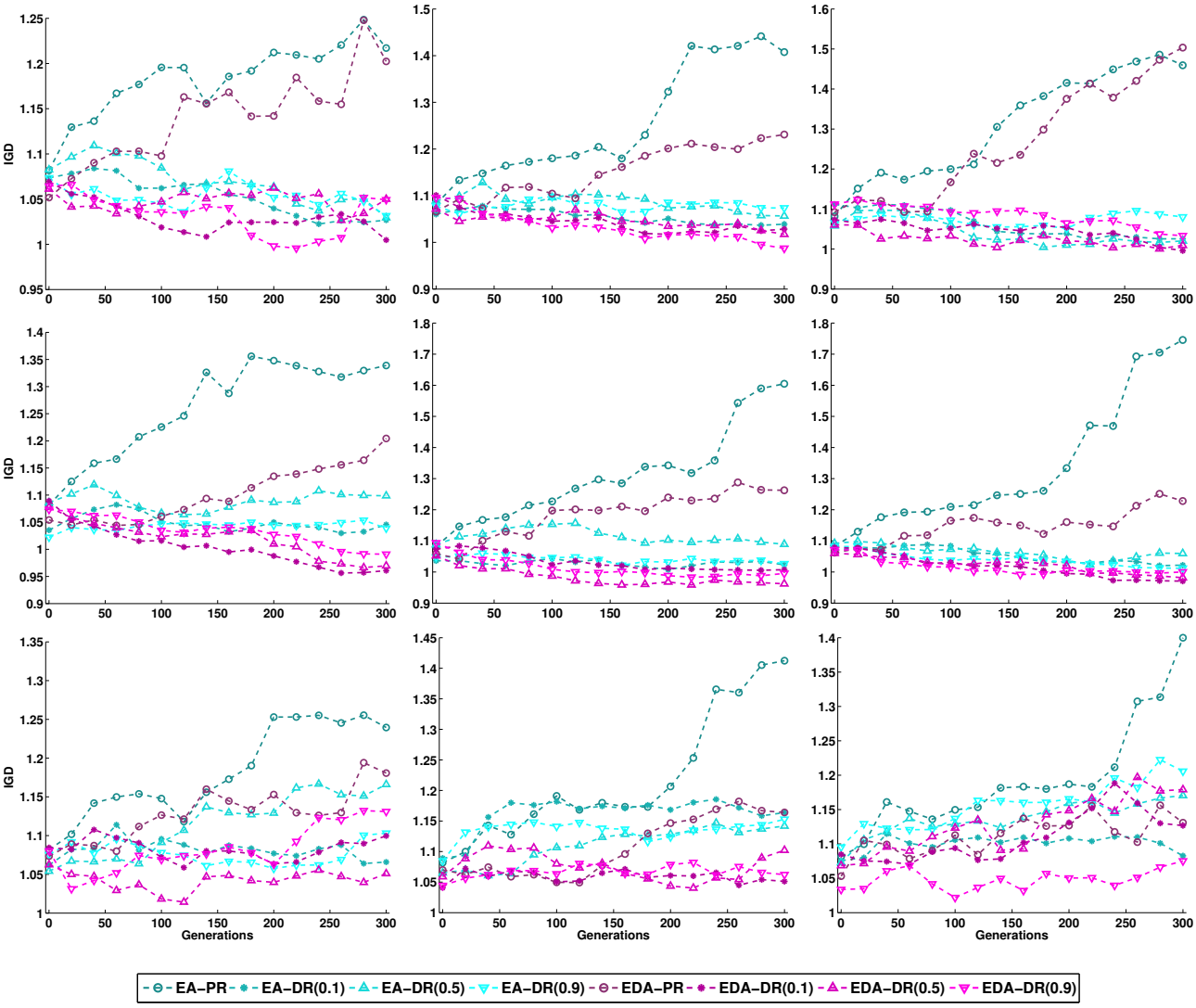



Figure 11.7: Average IGD of the Pareto fronts obtained for WFG3 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be minimized.

levels. WFG3 problem has a degenerated Pareto optimal front, and therefore a correct optimization strategy would focus the search more around the lower-dimensional optimal front during evolution. However, the hypervolume value of such a Pareto front will be lower than a worse approximation which is more spread in the objective space. The IGD indicator on the other hand, computes the distance between points on the Pareto optimal front and the approximated front, thus penalizing the more spread front in this case.

The objective functions of WFG7 problem are separable and unimodal, with bias in the optimum values for some of the variables. The results obtained with DR method on this problem outperform those of PR method with a relatively great margin, although the increase in the noise level reduces the effectiveness of this ranking method, especially when using it with EDA. Again, the Pareto fronts approximated with lower dominance degrees for DR method are better. However, it can be seen that when the confidence

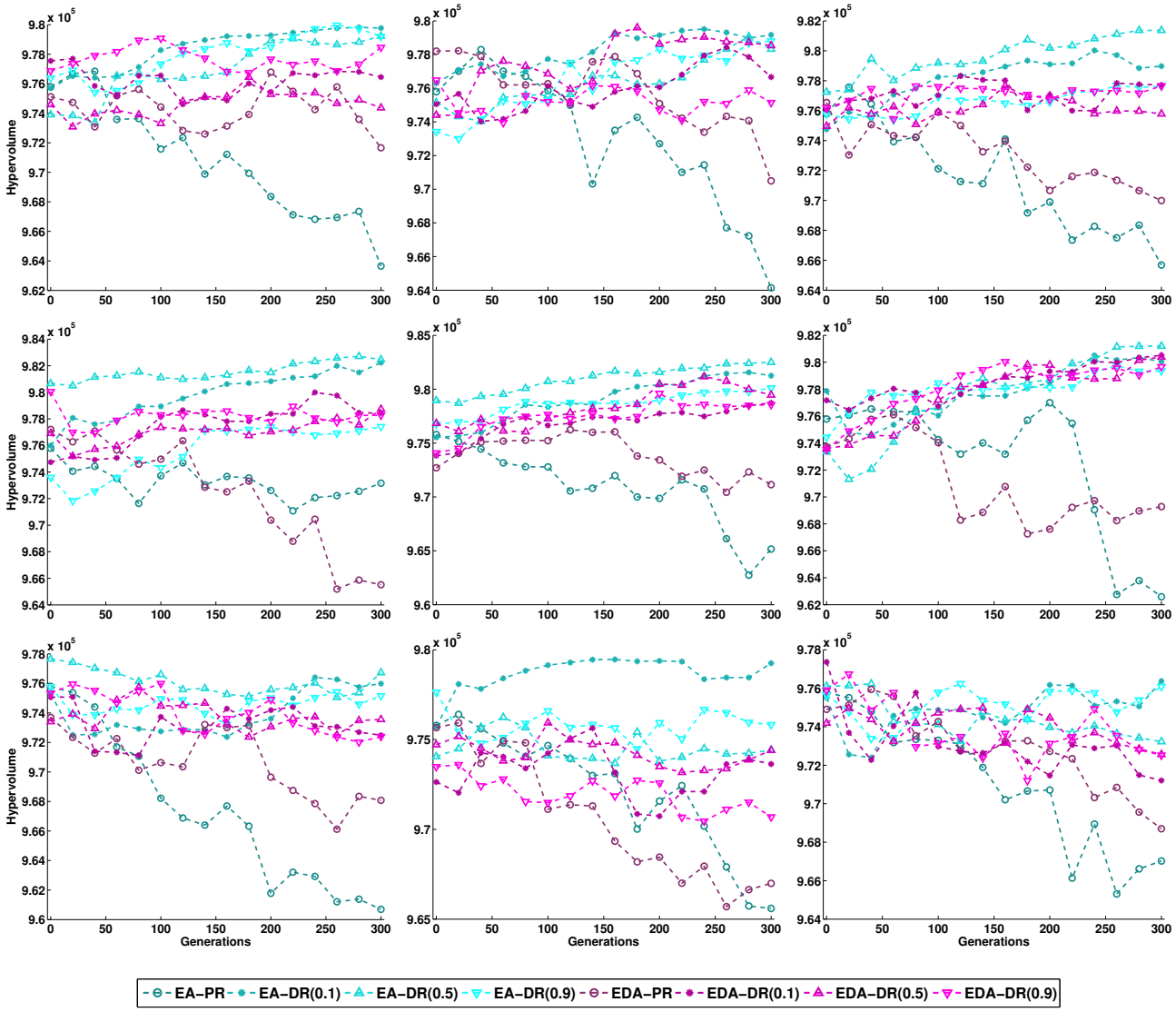



Figure 11.8: Average Hypervolume of the Pareto fronts obtained for WFG7 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be maximized.

level increases the performance of the algorithms using a high level of dominance degree is less affected. In general, the search with EA results in relatively better Pareto front approximations comparing with the fronts obtained by EDA-based search according to the quality indicators values. Moreover, with the increase in the level of noise or confidence, which makes the problem harder, the results obtained by the two algorithmic frameworks become comparable.

The objective functions of WFG9 problem in contrast to those of the previous problem are non-separable and multi-modal, with multi-modality being deceptive towards local optima for some of the objectives. As a result introducing noise to the objectives of this problem will make it very difficult to solve. The indicator values computed for the approximated Pareto fronts show that the proposed DR method outperforms solution

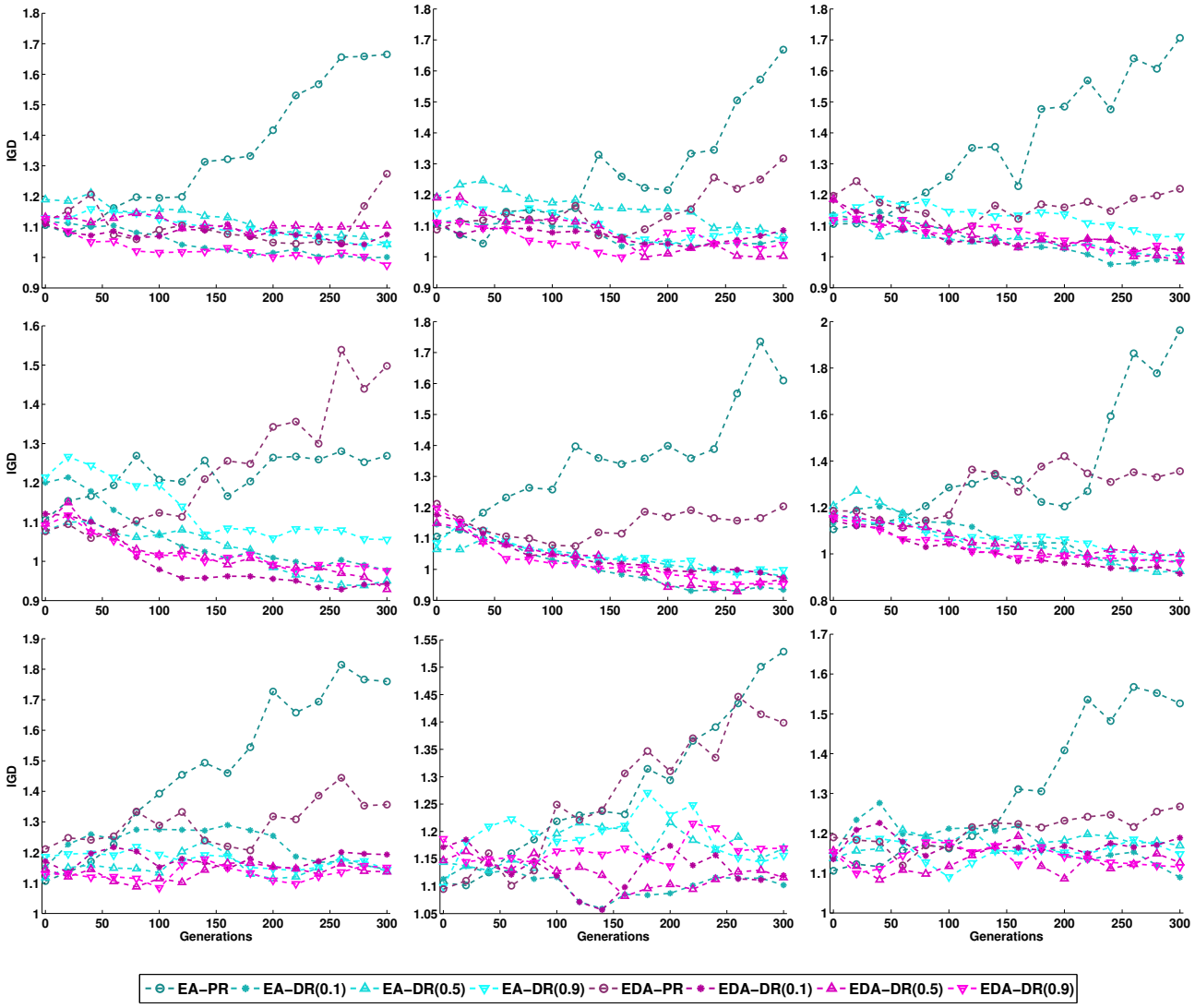



Figure 11.9: Average IGD of the Pareto fronts obtained for WFG7 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be minimized.

ranking based on dominance probability, though with the increase in the noise level the difference in the results obtained by the two methods gradually diminishes.

According to the results (Figures 11.10 and 11.11), lower dominance degrees like $\alpha = 0.1$ and $\alpha = 0.5$ allow a better ranking of solutions in DR method. Moreover, on the contrary to the previous problem, when the confidence level is increased DR method with higher dominance degree ($\alpha = 0.9$) is also greatly affected. Comparing the EDA and EA based search, it is observed that the Pareto fronts approximated by EDA are comparable or better to those obtained by EA, depending on the quality indicator. This better performance can be explained by the ability of the proposed EDA to capture the relationships between variables of the problem, which is necessary for finding the solutions.

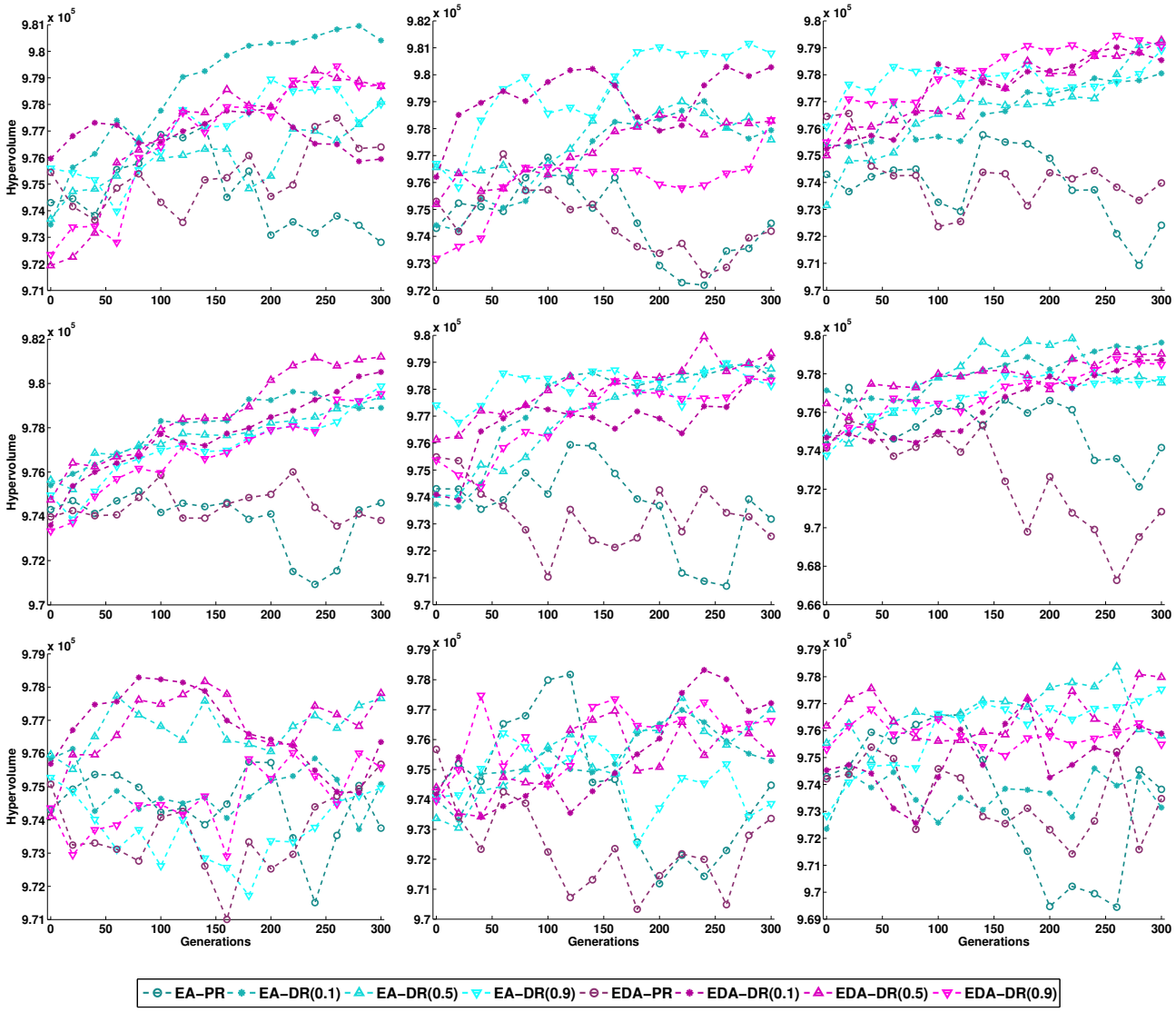



Figure 11.10: Average Hypervolume of the Pareto fronts obtained for WFG9 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be maximized.

### 11.5.4 Discussion

The values of the quality indicators for the Pareto sets obtained along the evolution path, show how the population of the different algorithm versions evolve. In general, based on the change in the values of these indicators during evolution, we can see that the solution ranking provided by DR method guides the optimization in the correct direction through the search space of the tested MOPs, with some exceptions like WFG1 and WFG3 with high level of noise. On the contrary, when PR method is used for ranking the solutions, the algorithm is misguided in the search space of many of the tested problems (e.g. see Figures 11.8 and 11.9).

When using DR method, the Pareto fronts approximated with smaller to medium values of dominance degree $\alpha$ are better on most of the tested problems with different

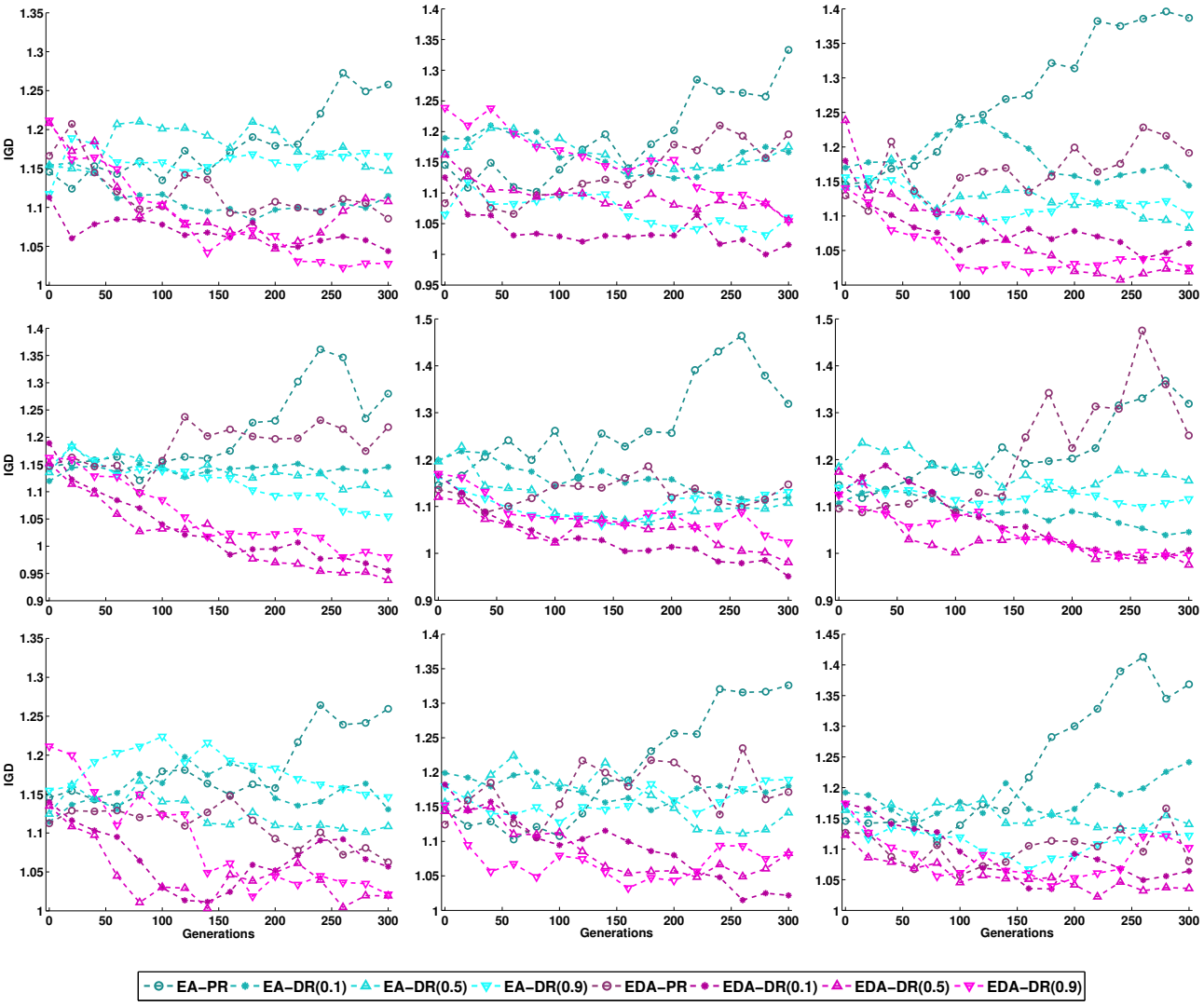



Figure 11.11: Average IGD of the Pareto fronts obtained for WFG9 problem with increasing noise (rows from top to bottom) and confidence levels (columns from left to right). This indicator should be minimized.

levels of noise and confidence. On the one hand, smaller values of $\alpha$ allow higher degrees of overlapping between intervals, meaning that the solutions can dominate each other easier. With small values of $\alpha$, according to Corollary 11.2 the ratio of dominated solutions increases and therefore more solutions become comparable, resulting in a finer ordering of the solutions by the employed $\alpha$-degree non-dominated sorting algorithm. This increase in the number of comparable solutions seems to compensate for the lack of strict decisions on the quality of the solutions. On the other hand, strict decisions about solution dominance allow only very good solutions to enter the $\alpha$-degree non-dominated set, while causing many solutions to become incomparable. Based on the experimental results, with a combination of the previous two effects which is achieved when using medium values of dominance degree (e.g. $\alpha = 0.5$), better Pareto set can be approximated on many of the tested MOPs and for different levels of noise and confidence.

Similarly, as the confidence level of the intervals (i.e. the values of the objective functions) decreases, more and more solutions become dominated (Corollary 11.1), and thus the solutions will be ordered in a larger number of $\alpha$-degree non-dominated sets during the first step of Algorithm 11.1, essentially leading to better solution ranking. Therefore, when DR method is used in the optimization the Pareto sets approximated for lower confidence levels are usually better, according to the quality indicator values. Taking more strict decisions by using high dominance degrees (e.g. $\alpha = 0.9$) when the confidence level increases, reduces the number of $\alpha$-degree non-dominated sets found in the first step of Algorithm 11.1, and thus the ordering of the solutions will depend more on their crowding distance computed in the second step of $\alpha$-degree non-dominated sorting algorithm. For some of the tested problems (e.g. WFG1 and WFG7), such an ordering with high $\alpha$ values seems to be less affected as the confidence level increases.

## 11.6 Conclusions

In real-world multi-objective optimization, the value of objective functions may involve noise and thus do not correctly represent the quality of solutions. Population-based EAs, which are one of the most successful methods in solving MOPs, have inherent abilities to deal with small levels of noise in objective values. However, with larger noise levels, especially when there are several conflicting objectives to be optimized, specific considerations are needed to perform successful optimization.

We considered noisy objective values given as intervals and proposed $\alpha$-degree Pareto dominance to deal with this type of values for the objective functions. This relation allowed to determine the dominance between solutions even when there is a specific degree of overlapping between some of the interval values, which is controlled by parameter $\alpha$. The similarity between this dominance relation and the conventional Pareto dominance allowed related terminology, like Pareto optimal set, to be easily defined. Assuming confidence intervals with a specific confidence level as the values of the noisy objective functions, we studied some of the properties of $\alpha$-degree Pareto dominance. It was shown that the $\alpha$-degree dominance relations defined between solutions are unaffected when the confidence level $\gamma$ or dominance degree $\alpha$ are reduced, and that such a reduction decreases the size of Pareto optimal set.

Based on this relation, $\alpha$-degree Pareto non-dominated sorting algorithm, an adaptation of the well-known non-dominated sorting algorithm in noisy domains with interval values, was proposed and integrated into MBN-EDA to find the solutions of noisy MOPs. The algorithm was tested on a set of MOPs where noisy objective values are given as intervals, and its solution ranking and search space exploration were respectively compared with a reference ranking method based on dominance probability and standard EA operators for continuous optimization.

The analysis of the approximated Pareto sets with two different quality indicators based on the corresponding noiseless Pareto fronts showed that the proposed solution ranking method based on $\alpha$-degree Pareto dominance relation allows the algorithms to achieve considerably better results comparing with the well-known probabilistic ranking method on the tested problems and for increasing levels of noise and confidence. We discussed how the change in the dominance degree and confidence level changes compa-

rability of the solutions using $\alpha$-degree Pareto dominance relation. Moreover, depending on the specific properties of the MOPs, the quality indicator values showed that the joint probabilistic modeling of variables and objectives allowed MBN-EDA to find Pareto set approximations for some of the tested noisy problems that are superior to those approximated by the standard multi-objective EA in different levels of noise.

# Chapter 12

# An Interval-Based Multi-Objective Approach to Feature Subset Selection Using Joint Probabilistic Modeling

## 12.1 Introduction

In its simplest form, a (supervised) classification task in data mining is to use a set of labeled data points to induce a classifier model, which can then be used to predict the label of new data points. The input data points are characterized by a number of feature values and a label, which identifies the class-value of each point. By features, we refer to the attributes or columns of a dataset and we use class to refer to the column containing the label or class-value of the data points. The classifier induced from the input data is used to find the class-value of an unlabeled data point, given its feature values.

A well-known problem related to classification is to find the subset of features that should be used for determining the class-values [Liu and Motoda, 1998], often referred to as feature subset selection (FSS). Selecting an appropriate subset of features can reduce the overall computational complexity and improve the classification accuracy. Therefore, usually an additional step is carried out before/whilst learning the classifier model from a training dataset to search for an appropriate subset of features, especially in high-dimensional problems with a large number of features. Moreover, the space of all possible feature subsets is huge, making it impossible to use exhaustive search methods to find the optimal feature subset(s) according to some optimization criteria. Therefore, a very good candidate for searching this space is to use stochastic heuristics. Especially, EAs have been used for FSS, as we saw in Section 3.6 for the case of Bayesian classifiers. One of the approaches to FSS which has gained a lot of attention in the past few years is multi-objective optimization. An important motivation for this approach is the intrinsic conflict between problem goals (e.g. maximize accuracy and minimize model complexity) which cannot be easily aggregated to a single objective.

Usually, the objectives considered for FSS cannot be computed directly from each subset of features. Instead, the objectives are *estimated* using a set of data points with/without a simulation process. Therefore, there is an inherent uncertainty in the

objective values obtained for feature subsets, which varies depending on the method and data points used for estimation. The optimization algorithm employed for FSS should take into account the noise in objective values during search. The noise handling techniques explained in Chapter 11 are especially useful for this purpose.

In this chapter, we adapt MBN-EDA for solving the FSS problem formulated as an MOP with six objective functions. These functions, whose values describe the performance of a classifier, are the area under the ROC curve, sensitivity, specificity, precision, F1 measure and Brier score. Following the discussion in the previous chapter, we consider interval values for each of these objectives to deal with the noise in their values. These intervals are obtained from estimating the objectives in different conditions (e.g. with different sets of data points). We propose a solution ranking method based on $\alpha$-degree Pareto dominance relation to select a subset of solutions in each generation.

Since the feature subsets are encoded as binary strings, the joint probabilistic model in MBN-EDA should be estimated in a mixed discrete-continuous domain. We propose a two-step approach for joint modeling of variables and objectives in MBN-EDA. In the first step, $\ell_1$-regularization is used to identify the set of more relevant variables to each objective. In the second step, an MBN is estimated as the joint model of variables and objectives, starting from the initial structure approximated in the first step. This method simplifies the estimation of joint model by removing irrelevant variables. Moreover, thanks to the interactions encoded in the joint model, we can study the relationships between the objectives considered for this problem, something which is less studied so far in FSS.

## 12.2 EMO Algorithms in FSS

FSS can be formally expressed as selecting the best subset of features for a learner model, given the set of all candidate features [Inza et al., 2000]. Therefore, the objective of FSS is to reduce the number of features used to characterize the dataset while improving the performance of the learner model on that dataset. According to Blum and Langley [1997], an FSS method should address the following issues:

1. Initial point: the starting point(s) of the search process. It can be an empty subset (i.e. no features selected), a full subset or a randomly generated subset.

2. Search strategy: the algorithm used to explore the space of possible feature subsets. This space is exponential in the number of features ($2^n$–where $n$ is the number of features), and thus FSS is considered to be a difficult combinatorial problem with an intractable computational complexity [Davies and Russell, 1994; Liu and Yu, 2005].

3. Feature subset evaluation: measuring the quality of different feature subsets, so that high quality subsets can be preferred over others. Generally, two major approaches to feature subset evaluation exist. The *wrapper* approach uses the performance of a learner model, trained with the features in a subset, as the evaluation criteria of that subset. In the *filter* approach, instead of the learner model performance, data-driven measures (e.g. correlation, mutual information, etc.) are used to evaluate the subset of features.

4. Search stopping criteria: determine how the search method will be terminated. For example insignificant change in the quality of feature subsets, or reaching a maximum number of feature subset evaluations.

Apart from the above mentioned filter and wrapper approaches, there is another less frequently approach called embedded FSS, where feature selection takes place during the training of the learner model. $\ell_1$-regularized model learning is an example of these methods.

A search algorithm for FSS deals with three different spaces. First, the space of all possible feature subsets which defines the search space. This is the space that the search algorithm explores. Second, the samples in the dataset represent points in the data space. For each solution in the search space, a different projection of the data space, obtained according to the features included in that solution, is used to evaluate the solution (e.g. by training and testing a learner model). Third, the result of evaluating each solution is a point in the objective space. Viewed in this way, feature subset evaluation is a function that maps each solution in the search space through data space to a point in the objective space.

Both wrapper and filter approaches have been used for evaluating feature subsets when using EMO algorithms for FSS. To evaluate a solution (i.e. feature subset) within a wrapper approach, first a learner model is trained using a training dataset and only taking the features included in that solution. Then this model is tested on a separate validation or test dataset to assess its performance. To increase the accuracy of the assessment, usually this process is repeated several times. Techniques like bootstrapping, $k$-fold cross-validation or leave-one-out (a special version of the latter) are used for partitioning the dataset and repeated evaluation of a learner model.

This way of evaluating solutions has a high computational complexity. However, from the perspective of learner model performance, the solutions found with this approach are often superior to those found with filter-based methods. Therefore, most of the EMO algorithms for FSS are based on a wrapper approach, which uses the $k$-fold cross-validation of a usually simple learner model with short training time. Some of the methods have also used distributed evaluation to speed up solution evaluation [Oliveira et al., 2006], or approximation techniques to prevent retraining the classifier for each single solution [Emmanouilidis et al., 2001; Oliveira et al., 2002].

Techniques like bootstrapping and $k$-fold cross-validation can obtain a good estimation of the quality of each solution. However, more accurate estimation of quality can be obtained by testing the final solutions found by the search algorithm on an independent dataset which is not used during the search algorithm. Moreover, to have a statistical estimation of a search algorithm performance in FSS, that algorithm should be run several times. Thus, to combine these two requirements for quality and performance assessment, several bi-partitions of the given dataset are considered. For each bi-partition, two individual runs of the search algorithm are performed. In the first run, one of the partitions is used to evaluate the solutions during the search process (e.g. by a $k$-fold cross-validation method) and after the search, the other partition is used to test the final solutions. In the second run the role of the partitions is exchanged.

In the following two sections we briefly review some of the works in the literature that use EMO algorithms for FSS. These works are summarized in Table 12.1. The intuitive method of representing a feature subset, adopted by all of the algorithms reviewed here,

is to use a binary encoding, i.e. a bit string of length equal to the number of features, where a zero value means the exclusion of the corresponding feature from the subset and a value of one means its inclusion. Thus, there is a one to one correspondence between the features of the dataset and the variables in the solutions to the FSS problem.

### 12.2.1 FSS in Classification

Emmanouilidis et al. [2000] proposed a commonality-based crossover in the context of niched Pareto GA (NPGA) [Horn et al., 1994], where common alleles of the parent solutions are directly copied to the offspring solutions, and the other genes are inherited based on a probability computed from the number of common genes. They used the classification accuracy of artificial neural networks (ANN) together with the size of feature subset as the objectives of optimization. They also applied their method to the rotating machinery fault diagnosis problem with respectively two (approximated root mean squared error of ANN and feature subset size) [Emmanouilidis et al., 2001] and three objectives (sensitivity and specificity of a nearest neighbor (NN) classifier and the size of feature subset) [Emmanouilidis, 2001].

Oliveira et al. [2002] used the first version of non-dominated sorting GA (NSGA) [Srinivas and Deb, 1994], which is based on fitness sharing, to select a good subset of features for handwritten digit recognition problem. The classification accuracy of an ANN classifier and the size of feature subsets were used as optimization objectives. They also used EMO algorithms to search for the best ensemble of the classifiers found in the Pareto set after FSS search [Oliveira et al., 2006].

Some other works have used NSGA-II to search for good feature subsets in FSS problems. Shi et al. [2004] tried to find the best feature subsets for an ensemble of support vector machines (SVMs) in the protein fold recognition problem. Three objectives were used for optimization: 1) cross-validation classification accuracy, 2) test classification accuracy, and 3) feature subset size. Hamdani et al. [2007] studied the performance of NSGA-II for FSS on several datasets of varying sizes. They used the classification accuracy of an NN classifier and feature subset size as optimization objectives. Ekbal et al. [2010] searched for relevant features of the named entity recognition problem in the field of natural language processing with a wrapper method which uses maximum entropy-based classifiers. Recall and precision of the classifiers were used as objectives during the search process, and F-measure was used to select one of the feature subsets from the final Pareto set. Instead of considering all feature subset sizes together, Huang et al. [2010] performed separate optimizations for each of the feature subset sizes in the problem of predicting customer churn in telecommunications. Classification accuracy, true positive rate and true negative rate of a decision tree (DT) classifier were used as objectives in each of the optimization runs.

In a different context, Rodríguez and Lozano [2008] used NSGA-II to perform a multi-objective search for the best structure of an MBN, which involves selecting the subset of features relevant to each class variable. They used the classification accuracy of each of the classes as the optimization objectives. Radtke et al. [2009] proposed a three phase multi-objective optimization scheme for: 1) feature extraction, 2) single classifier or ensemble components selection, and 3) FSS to improve the performance of the selected classifier or ensemble. They compared the performance of NSGA-II and a multi-objective memetic

Table 12.1: Summary of the methods for FSS using EMO algorithms.

| | Approach | Optimizer | Learner Model | # Objectives |
|---|---|---|---|---|
| Classification | | | | |
| [Emmanouilidis et al., 2000] | Wrapper | NPGA | ANN | 2 |
| [Emmanouilidis et al., 2001] | Wrapper | NPGA | ANN | 2 |
| [Emmanouilidis, 2001] | Wrapper | NPGA | 1-NN | 3 |
| [Oliveira et al., 2002] | Wrapper | NSGA | ANN | 2 |
| [Oliveira et al., 2006] | Wrapper | NSGA | ANN | 2 |
| [Shi et al., 2004] | Wrapper | NSGA-II | Ensemble of SVMs | 3 |
| [Hamdani et al., 2007] | Wrapper | NSGA-II | 1-NN | 2 |
| [Rodríguez and Lozano, 2008] | Wrapper | NSGA-II | MBN | 2 |
| [Ekbal et al., 2010] | Wrapper | NSGA-II | Max entropy-based classifier | 2 |
| [Huang et al., 2010] | Wrapper | NSGA-II | DT | 3 |
| [Radtke et al., 2009] | Wrapper | NSGA-II, Memetic EMO | ANN, Projection distance-based classifier | 2 |
| [Vatolkin et al., 2012] | Wrapper | Indicator-based EMO | DT, Random forest, NB, SVM | 2 |
| [Zhu et al., 2009] | Hybrid | Memetic EMO | DT | 3, 4 |
| [Spolaôr et al., 2011] | Filter | NSGA-II | — | 2 |
| Clustering | | | | |
| [Kim et al., 2002] | Wrapper | ELSA | K-means, EM | 3, 4 |
| [Morita et al., 2003] | Wrapper | NSGA | K-means | 2 |
| [Zhang et al., 2006] | Wrapper | Immunology-based EMO | Fuzzy c-means | 3 |
| [Handl and Knowles, 2006] | Wrapper, Filter | PESA-II | K-means | 2 |
| [Zaharie et al., 2007] | Filter | NSGA-II | — | 4 |

algorithm on the handwritten digit recognition problem, with respect to classification accuracy and feature subset size as objectives.

Zhu et al. [2009] used a hybrid wrapper-filter approach by combining wrapper-based NSGA-II and filter-based local search in a memetic algorithm. The accuracies of each of the class-values in a one-versus-all classification scheme, obtained from DT classifiers, were used as optimization objectives in NSGA-II, whereas the criterion for local search was based on the feature-class relevance. Spolaôr et al. [2011] proposed several filter-based bi-objective optimizations using NSGA-II for FSS, each time pairing interclass distance measure with one of the following criteria: ratio of inconsistent pairs of samples in the dataset, feature-class correlation, Laplacian score of the samples in the dataset, and features entropy.

Very recently, Vatolkin et al. [2012] employed a hypervolume indicator-based EMO algorithm to search for good feature subsets in the high-dimensional problem of musical instruments recognition in a polyphonic audio mixture. They used the relative feature subset size and mean squared error of classification as optimization objectives. DTs, random forests, NBs and SVMs were used as alternative classifiers to compute the second objective.

### 12.2.2 FSS in Clustering

In the context of unsupervised learning or clustering, the solution encoding may be extended to also include the number of clusters. Thus, the algorithm will be simultaneously searching the space of possible feature subsets and the space of possible numbers of cluster. The learner model in this context is a clustering algorithm which assigns the samples in the dataset to a number of clusters. The objectives here are usually based on increasing the closeness of the samples in the same cluster (cluster compactness), while increasing the separation between different clusters.

Kim et al. [2002] proposed an evolutionary local search algorithm (ELSA) to select the proper subset of features for clustering. They proposed four objectives when using the K-means algorithm and three objectives when using the EM algorithm for clustering. Morita et al. [2003] used NSGA with two objectives for FSS when clustering handwritten month names with the K-means algorithm. Handl and Knowles [2006] proposed a general framework for bi-objective FSS in clustering problems which encompasses both filter and wrapper approaches. Their framework is based on the second version of Pareto envelope-based selection algorithm (PESA-II) [Corne et al., 2001] and uses the K-means algorithm for clustering when the objectives are evaluated with the wrapper approach.

Instead of representing the feature subsets with bit strings, Zhang et al. [2006] encode feature saliencies in real-valued strings and select only those features with a saliency value above a given threshold. They employed an immunology-based EA to solve a three-objective optimization problem by using a fuzzy c-means algorithm for clustering. Zaharie et al. [2007] used NSGA-II, with four different objectives in a filter approach, to find the best ranking of the features, a problem closely related to FSS.

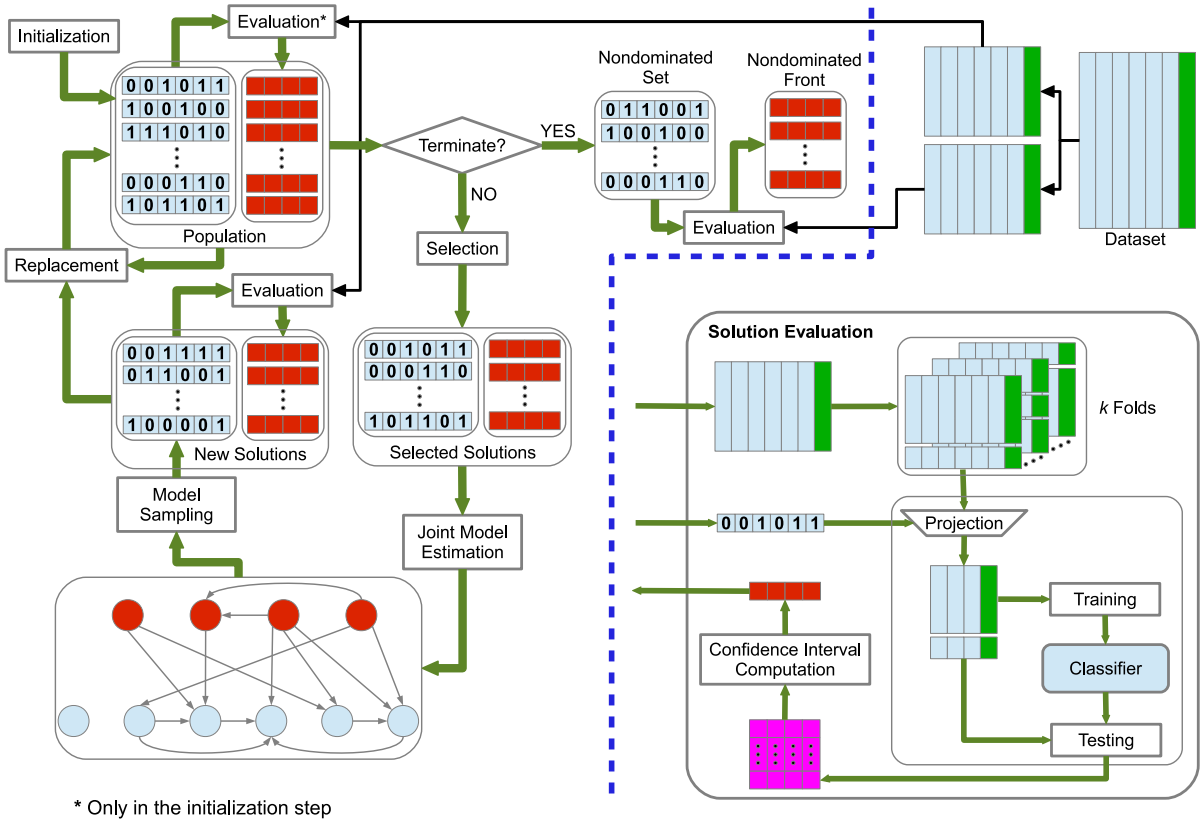



Figure 12.1: The overview of the proposed EDA for FSS in classification.

## 12.3 Multi-Objective FSS with Joint Modeling of Variables and Objectives

In the following sections we explain the essential steps of the algorithm proposed for FSS. The algorithm, which is an adaptation of MBN-EDA for FSS, employs a solution ranking method based on $\alpha$-degree Pareto dominance for dealing with the noise in the objective values when selecting a subset of solutions. A special method is proposed for learning a joint probabilistic model of variables and objectives for this problem. Figure 12.1 shows the overall outline of the proposed algorithm.

### 12.3.1 Solution Ranking

In the last chapter, we saw that one of the ways to increase the confidence in the objective values returned for a noisy MOP is the reevaluation of objective functions on each solution. However, in some of the MOPs, solution evaluation inherently involves multiple reevaluations. In the case of FSS problem, when the feature subsets are evaluated using $k$-fold cross-validation or bootstrapping, a set of values are obtained for each of the objectives. These values are then averaged to obtain a more accurate estimation of the quality of each feature subset.

Instead of estimating singular values as the value of each objective using these reevaluations, an alternative approach, which has been less studied so far, is to use all of the

reevaluation values, representing the quality of each solution under different conditions, and assume that each objective function returns interval values. We saw that in this way the algorithm can take into account the noise in objective values when selecting a subset of solutions.

Having interval values for the objectives, the PR method based on probabilistic dominance can be used for ordering the solutions, as was explained in the last chapter. However, the computational complexity of calculating the probabilities when using this ranking method is one of its major drawbacks. Here, we propose a method for ordering the solutions based on $\alpha$-degree Pareto dominance, called degree ranking (DR). In this method, first the solutions in the population are sorted into a number of $\alpha$-degree Pareto sets, and then the rank of each solution $\boldsymbol{x}_i$ within the population is computed as

$$\text{rank}_{DR}(\boldsymbol{x}_i) = \theta_{ND} \cdot r_i + \frac{1}{N} \sum_{k=1}^{N} \sum_{j=1}^{m} \deg_j(\boldsymbol{x}_k, \boldsymbol{x}_i) + \theta_B \cdot \sum_{j=1}^{m} I\big(\lfloor f_j(\boldsymbol{x}_i) \rfloor, f_j^*\big), \qquad (12.1)$$

where $r_i$ is the rank of the $\alpha$-degree Pareto set containing solution $\boldsymbol{x}_i$ (starting from 1 for the best Pareto set). $I(a, b)$ is an indicator function returning one if $a$ is equal to $b$ and zero otherwise, and $f_j^*$ is the best value reached so far in the $j$th objective. Here, we are assuming a minimization problem and therefore $\lfloor f_j(\boldsymbol{x}_i) \rfloor$ shows the best point of the interval value obtained for $f_j(\boldsymbol{x}_i)$. $\theta_{ND}$ and $\theta_B$ are the scaling coefficients determining the importance of respectively $\alpha$-degree Pareto ranks and best-found boundary values in ranking the solutions.

The ranking method in Equation (12.1) combines the measures of solution convergence and dispersion with $\alpha$-degree Pareto ranks. This measures are very similar to the gain-based and distance to best-based ranking methods [Garza-Fabre et al., 2009, 2010a] that where individually used for multi-objective optimization in Chapter 10. The second and third terms of Equation (12.1) can be computed while sorting the solutions into $\alpha$-degree Pareto non-dominated sets (which has a complexity of order $O(N^2 m)$). Thus, their computation does not impose additional computational overhead like when computing crowding distances in the adapted non-dominated sorting algorithm used in the previous chapter. Since solution ranking is one of the most time consuming steps in EMO algorithms, this reduction in computational time is highly favorable.

### 12.3.2 Joint Model Learning

As it was explained in Section 12.2, FSS is a combinatorial problem with a discrete search domain where candidate solutions are represented with bit strings. However, the objectives considered for this problem are usually continuous-valued functions or, as assumed in Section 12.3.1, functions with interval values. Although, as it was reviewed in Chapter 1, there are some approaches for learning probabilistic models in domains with mixed continuous-discrete variables, but their application to the joint modeling of variables and objectives in our algorithm will be computationally very demanding, and thus can not scale well due to its complexity. The method we propose here for learning a joint model of variables and objectives consists of two major steps. Figure 12.1 shows the outline of this approach.

**Inputs:**
Selected solutions $\mathcal{D}_{\boldsymbol{X}}$
Their objective values $\mathcal{D}_{\mathcal{F}}$

// First part
$\mathcal{D}_{\mathcal{F}}^{\hat{\mathbb{E}}} \leftarrow$ Estimate expected values of objectives from $\mathcal{D}_{\mathcal{F}}$
**for all** $f_j \in \mathcal{F}$ **do**
  $\mathcal{M}_R[\boldsymbol{X}, f_j] \leftarrow$ Estimate an RRM from $\left(\mathcal{D}_{\boldsymbol{X}}, \mathcal{D}_{f_j}^{\hat{\mathbb{E}}}\right)$
**end for**
$\boldsymbol{X}_S \leftarrow$ Combine variables selected in $\mathcal{M}_R[\boldsymbol{X}, f_j], \forall f_j \in \mathcal{F}$
$\bar{\boldsymbol{X}}_S \leftarrow \boldsymbol{X} \setminus \boldsymbol{X}_S$
$\mathcal{S}_I[\boldsymbol{X}_S, \mathcal{F}] \leftarrow$ Initialize to empty structure
**for all** $f_j \in \mathcal{F}$ **do**
  $\mathcal{M}_R[\boldsymbol{X}_S, f_j] \leftarrow$ Remove variables in $\bar{\boldsymbol{X}}_S$ from $\mathcal{M}_R[\boldsymbol{X}, f_j]$
  $\mathcal{S}_I[\boldsymbol{X}_S, \mathcal{F}] \leftarrow \mathcal{S}_I[\boldsymbol{X}_S, \mathcal{F}]+$ Structure of $\mathcal{M}_R[\boldsymbol{X}_S, f_j]$
**end for**

// Second part
$\mathcal{D}_{\mathcal{F}}^{D} \leftarrow$ Discretize objective values in $\mathcal{D}_{\mathcal{F}}^{\hat{\mathbb{E}}}$
$\mathcal{M}_1[\boldsymbol{X}_S, \mathcal{F}] \leftarrow$ Estimate an MBN from $\left(\mathcal{D}_{\boldsymbol{X}_S}, \mathcal{D}_{\mathcal{F}}^{D}\right)$ starting from the structure $\mathcal{S}_I[\boldsymbol{X}_S, \mathcal{F}]$
**for all** $X_i \in \bar{\boldsymbol{X}}_S$ **do**
  $\mathcal{M}_2[X_i] \leftarrow$ Estimate univariate probability distribution from $\mathcal{D}_{X_i}$
**end for**

**Output:** $\left(\mathcal{M}_1[\boldsymbol{X}_S, \mathcal{F}], \{\mathcal{M}_2[X_i] \mid X_i \in \bar{\boldsymbol{X}}_S\}\right)$

Algorithm 12.1: Outline of the joint model estimation method.

### Variable Selection with Regularized Regression

In the first part of the joint modeling algorithm, we adopt a method similar to the first approach of regularized model learning in Section 5.2, based on regularized regression, for finding the most related subset of variables to each objective. Formally, let the set $\left\{\left(\boldsymbol{x}_1, \boldsymbol{f}(\boldsymbol{x}_1)\right), \ldots, \left(\boldsymbol{x}_N, \boldsymbol{f}(\boldsymbol{x}_N)\right)\right\}$ denote a joint dataset of variable-objective values, with objective values given as confidence intervals. Then, we learn an $\ell_1$ regularized regression model (RRM) (Section 4.2) for each objective $f_j$ given the variables:

$$\arg\min_{\boldsymbol{\beta}_j} \left( \sum_{i=1}^{N} \left( \hat{\mathbb{E}}\big(f_j(\boldsymbol{x}_i)\big) - \big(\boldsymbol{\beta}_{j<0>} + \sum_{k=1}^{n} \boldsymbol{\beta}_{j<k>} \boldsymbol{x}_{i<k>}\big) \right)^2 + \lambda \sum_{k=1}^{n} |\boldsymbol{\beta}_{j<k>}| \right). \tag{12.2}$$

In this equation, the estimated expected values of each objective are used to estimate the RRM parameters, which as explained in Section 11.4.2 can be a good representative of the corresponding interval values. The $n+1$-dimensional vector $\boldsymbol{\beta}_j$ is the parameter estimated for the RRM of the $j$th objective. AIC [Akaike, 1974] is used as the scoring metric to select between different parameter vectors for each RRM. Because of the variable selection property of $\ell_1$ regularization, each of the estimated RRMs shows the subset of variables which is more relevant to the corresponding objective. Bearing in mind that in joint modeling of variables and objective with MBN, the bridge subgraph also encodes

a kind of variable selection for each objective, the estimated RRMs can be used as an approximation of the initial structure of this subgraph.

At the end of this part of joint modeling, the subset of variables selected for each of the objectives in the RRMs are combined to obtain a common subset of most relevant features to all objectives. In Section 4.3 we saw that strategies like the union or the intersection of the variable subsets can be used for combining these subsets [Meinshausen and Bühlmann, 2006]. Here, we adopt an intermediate approach by selecting those variables that have appeared in at least half of the RRMs. As a result, the set of variables are divided into two groups: those selected in the combined model, $\boldsymbol{X}_S$, and the rest, $\bar{\boldsymbol{X}}_S = \boldsymbol{X} \setminus \boldsymbol{X}_S$.

### MBN Estimation

In the second part, an MBN is estimated, modeling the objectives and variables in $\boldsymbol{X}_S$. For this purpose, using an equal frequency discretization method, the objective values are discretized into three nominal values: good, average and bad. Before starting the greedy local search for estimating MBN, its structure, or more specifically its bridge subgraph, is initialized with the structure of the combined model obtained in the first part. Our experiments have shown that this initialization can highly reduce the search effort needed to find a good MBN structure with greedy local search, as they are usually good approximations of the interactions encoded in the bridge subgraph. Since the variables and objectives are discrete, the parameters encoded in all MBN nodes are conditional probability tables which are used to evaluate different MBN structures with the BIC scoring metric, as formulated in Equation (1.8).

The variables in $\bar{\boldsymbol{X}}_S$, which are considered less important to the objectives, can be ignored in the modeling process, only copying their values when generating new solutions. However, to allow the possibility of future participation of these variables in joint modeling (in the next generations), we estimate a univariate marginal probability distribution for each variable in $\bar{\boldsymbol{X}}_S$ (similar to UMDA [Mühlenbein and Paaß, 1996]) to explore the subspace of these variables with a low-complexity modeling.

At the end, the adaptation of joint probability distribution of variables and objectives, presented in Equation (10.2), which is now encoded in both MBN and individual univariate probability distributions, is given by

$$P\big(x_1,\ldots,x_n,q_1,\ldots,q_m\big) = \prod_{X_i \in \boldsymbol{X}_S} P(x_i|\boldsymbol{pa}_i) \cdot \prod_{X_k \in \bar{\boldsymbol{X}}_S} P(x_k) \cdot \prod_{j=1}^{m} P(q_j|\boldsymbol{pa}'_j). \qquad (12.3)$$

All of the notations in this equation have the same meaning as in Equation (10.2). For example, $\boldsymbol{q} = (q_1,\ldots,q_m)$ denotes a possible discrete value-setting for the objective variables $\boldsymbol{Q} = (Q_1,\ldots,Q_m)$.

The probability distribution of Equation (12.3) is used in the sampling step of MBN-EDA to generate new candidate solutions. For the variables included in the MBN ($\boldsymbol{X}_S$), the second sampling approach described in Section 9.3.2 is used to generate new values according to the objective values encoded in the joint model. For the rest of the variables ($\bar{\boldsymbol{X}}_S$), new values are sampled from their estimated univariate marginal probability distributions.

## 12.4 Problem Formulation

This section describes how FSS is formulated as an MOP to be solved by the proposed adaptation of MBN-EDA. We use a wrapper approach to evaluate feature subsets with two different Bayesian classifiers, namely the NB and TAN classifiers. It should be noted that these two classifiers define two different optimization problems on each dataset since the best feature subset obtained for one type of classifier might not necessarily be the best feature subset for another type of classifier.

The NB classifier training consists only of finding the prior probability of class-values and the conditional probabilities of each feature values. Despite its simple structure, NB classifier is shown to have very good classification performance in many real-world problems. The TAN classifier training algorithm involves finding the maximum weighted spanning tree over the features, based on their conditional mutual information given the class. It is proven that this algorithm learns the maximum log-likelihood TAN classifier for a given dataset [Friedman et al., 1997].

These two classifiers choose the class-value with the highest posterior probability as the predicted label of a given value-setting $\boldsymbol{x}$ of features. In a binary classification problem (i.e. when there are only two different class-values) the predicted class-values for a set of data points can be compared against their true class-values to assess the accuracy of the classification. Assuming that the class-values assigned to the data points are either *positive* or *negative*, the number of correct (true) and wrong (false) classifications are usually organized as follows in a data structure called confusion matrix:

| | | True Label: Positive | True Label: Negative |
|---|---|---|---|
| Predicted Label | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

In this matrix, TP and TN show the number of correct classifications in each of the classes, whereas FP and FN show the number of data points which where wrongly classified.

We use six different performance measures, based on the classification accuracy given in the confusion matrix and the encoded class-value probabilities, to evaluate the classifiers on a test set. These measures are: sensitivity, specificity, precision, area under receiver operating characteristics curve (AUC), F1 and Brier score, and are computed as

follows:

$$
\begin{aligned}
f_{AUC} &= \frac{G_1 + 1}{2}, \\
f_{sens} &= \frac{TP}{TP + FN}, \\
f_{spec} &= \frac{TN}{TN + FP}, \\
f_{prec} &= \frac{TP}{TP + FP}, \\
f_{F1} &= \frac{2TP}{2TP + FN + FP}, \\
f_{Brier} &= \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{C} \big(P(c_k \mid \boldsymbol{x}_i) - \delta(c_k, \boldsymbol{x}_i)\big)^2,
\end{aligned} \tag{12.4}
$$

where $G_1$ is the Gini coefficient estimated by approximating the area under the Lorenz curve [Gastwirth, 1972]. $N$ is the total number of samples, i.e. $N = TP+TN+FP+FN$. $C$ is the number of possible class-values: $\{c_1, \ldots, c_C\}$, and function $\delta(c_k, \boldsymbol{x}_i)$ returns one if the true class-value of instance $\boldsymbol{x}_i$ is $c_k$ and zero otherwise. These measures define the objective functions of optimization. The first five objectives should be maximized and have a range of values in $[0, 1]$, whereas the last objective should be minimized and represents the calibration error in classification.

Following the common practice in FSS literature, we use binary encoding to represent feature subsets with bit strings of length equal to the number all features. Each feature subset is evaluated using $k$-fold cross-validation of a classifier on a given dataset, projected over the features in the subset. Thus, we will obtain $k$ values for each of the classifier performance measures (objectives). These values are then used to compute a confidence interval with a confidence level $\gamma \in [0, 1]$ for each of the performance measures.

## 12.5 Experiments

We have used three datasets with an increasing number of features to study the performance of our proposal for FSS. These datasets, which are all retrieved from UCI online machine learning repository[1], are Wisconsin diagnostic breast cancer (WDBC), ozone level detection (Ozone) and Hill-Valley, with details presented in Table 12.2. To handle the missing values in the Ozone dataset, we discard samples with missing class-value, or if half of the features are missing. Otherwise, the missing values are replaced with the mean value of that feature.

The Pareto optimal set of a FSS problem is not known beforehand. Therefore, the three quality indicators of hypervolume, maximum spread [Zitzler et al., 2000] and Schott's spacing [Schott, 1995] are used to respectively examine the convergence, diversity and distribution of the final Pareto fronts approximated by the algorithm. These indicators are computed using the expected values of the objectives. As it was defined

[1] `http://archive.ics.uci.edu/ml/datasets.html`

Table 12.2: Datasets considered for the experiments

| Name | # samples | # features | # class-values | Missing values |
|---|---|---|---|---|
| WDBC | 569 | 30 | 2 | No |
| Ozone | 2536 | 72 | 2 | Yes |
| Hill-Valley | 2424 | 100 | 2 | No |

in Equation (10.11), hypervolume indicator computes the total hyper-volume of the objective space that is dominated by the solutions in the approximated Pareto front, and larger values of this indicator show better approximations.

Maximum spread indicator gives an estimation of the Pareto front diversity by taking into account the minimum and maximum values achieved for each objective. Given a Pareto set approximation $A$, this indicator is computed as

$$MS(A) = \sqrt{\sum_{j=1}^{m} \max_{\boldsymbol{x},\boldsymbol{y}\in A} \mathrm{d}\big(f_j(\boldsymbol{x}), f_j(\boldsymbol{y})\big)}. \tag{12.5}$$

Larger values of this indicator show a more diverse front and are desired. Schott's spacing indicator is a measure of how evenly the approximated Pareto front is distributed. It is defined as the standard deviation of distances between each solution and its nearest neighbor in the objective space. Specifically, given a Pareto set approximation $A$, the distance contribution of a solution $\boldsymbol{y} \in A$ to the value of this indicator is computed as

$$\min_{\boldsymbol{x}\in A, \boldsymbol{x}\neq\boldsymbol{y}} \mathrm{d}\big(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y})\big),$$

where $\mathrm{d}(\cdot,\cdot)$ can be any distance function like city-block or Euclidean distance. Lower values of Schott's spacing indicator are favored.

### 12.5.1 Experimental Design

To evaluate the performance of the proposed algorithm, 5 random bi-partitions of each dataset are generated (elsewhere this method is called $5 \times 2$ cross-validation), resulting in 10 independent runs. For each run, the number of cross-validation folds is set to $k = 5$ and a confidence level of $\gamma = 0.95$ is used to compute the intervals for the performance measures (objectives). The evaluation of the final Pareto set of each run on the independent test set is used to compute the final Pareto front approximated in that run.

The initial population of MBN-EDA is generated randomly with a uniform distribution. The full set of features and the empty subset are also added to the initial population to provide both of the possible values for all of the variables in the initial population. The population size is set to $N = 2,000$ to provide adequate statistics for the estimations of MBN parameters. Such a population size is also suggested in other works using BN-based EDAs for FSS [Inza et al., 2000]. When estimating the MBN structure, the maximum number of parents for each node is set to $\max\{m, \lceil \log_3(\tau N) \rceil\}$, allowing the possibility for the variable nodes to have all objectives as their parents, while preventing very complex structures. In our experiments, the MBN learning algorithm virtually never

needed to surpass this maximum (in less than 0.15% of model estimations an operation was canceled because of reaching this maximum), which shows its negligible influence on MBN estimation. To compensate for the large population size requirement of MBN-EDA, we set the maximum number of generations to 50.

### 12.5.2 MBN-EDA with Different Solution Ranking Methods

In the first set of experiments, the feature subsets found by MBN-EDA when using PR and DR methods for solution ranking are compared and analyzed. PR is computed based on the simplifying assumption of Equation (11.8) and the approximation of Equation (11.9). Similar to the previous chapter, three different degrees of dominance are studied for the $\alpha$-degree Pareto dominance relation in DR method, i.e. $\alpha \in \{0.1, 0.5, 0.9\}$. When choosing the values of the scaling coefficients in Equation (12.1), two points should be taken into consideration: 1) solutions in lower-ranked Pareto sets should generally receive better ranks than the solutions in higher-ranked Pareto sets, and 2) solutions with objective values close to the best-value found for any of the objectives should be preferred to advocate Pareto fronts with larger diversity. After testing different combinations of values for these coefficients, they are set to $\theta_{ND} = 2$ and $\theta_B = -2$ in the following experiments.

Figures 12.2–12.4 show the values of the quality indicators for the final Pareto sets obtained with these ranking methods, on the three datasets considered in this study. For the WDBC dataset (Figure 12.2), the PR method has a slightly better average performance with respect to all three indicators and for both classifiers. Comparing the results obtained with different $\alpha$ values in the DR method we see that, according to the hypervolume indicator, with higher $\alpha$ values better Pareto sets are approximated. As it was explained in the previous section, higher $\alpha$ values place stricter requirements for a solution to dominate the others, resulting in fewer non-dominated solutions but with a higher degree of reliability. Thus, in the presence of noise in the objective values, higher $\alpha$ values allow better convergence of the approximated Pareto fronts.

For Ozone and Hill-Valley datasets which have large search spaces and thus the level of noise in the objective values can increase, the Pareto fronts approximated with the DR method are better spread than the ones obtained with the PR method, resulting in higher hypervolume values especially for the NB classifier. It can be seen in Figure 12.4 that lower diversity of the fronts obtained for Hill-Valley dataset by the PR method causes small spacing between the solutions in these fronts. This means that with the PR method the search is focused on a smaller region of the space. In general, smaller spacing is favorable in the comparison of two non-dominated fronts if they have similar diversity.

Figure 12.5 shows the time required by each of the ranking methods to order the solutions for the replacement step of MBN-EDA, on a machine with 2.66 GHz Intel Core-i5 processor and 6 GB of memory. The times are averaged over the generations of each run and over the three datasets. It can be seen that DR method based on $\alpha$-degree Pareto dominance requires significantly less time (less than half) than the PR method based on probabilistic dominance, even when using the approximation in Equation (11.9) for its computation. This time is not directly dependent on the specific dataset or classifier used for evaluation. Rather, the choice of dataset and classifier influences the objective values of the solutions, creating different instances of the population to be ranked.

Although NB and TAN classifiers define two different optimization problems for each

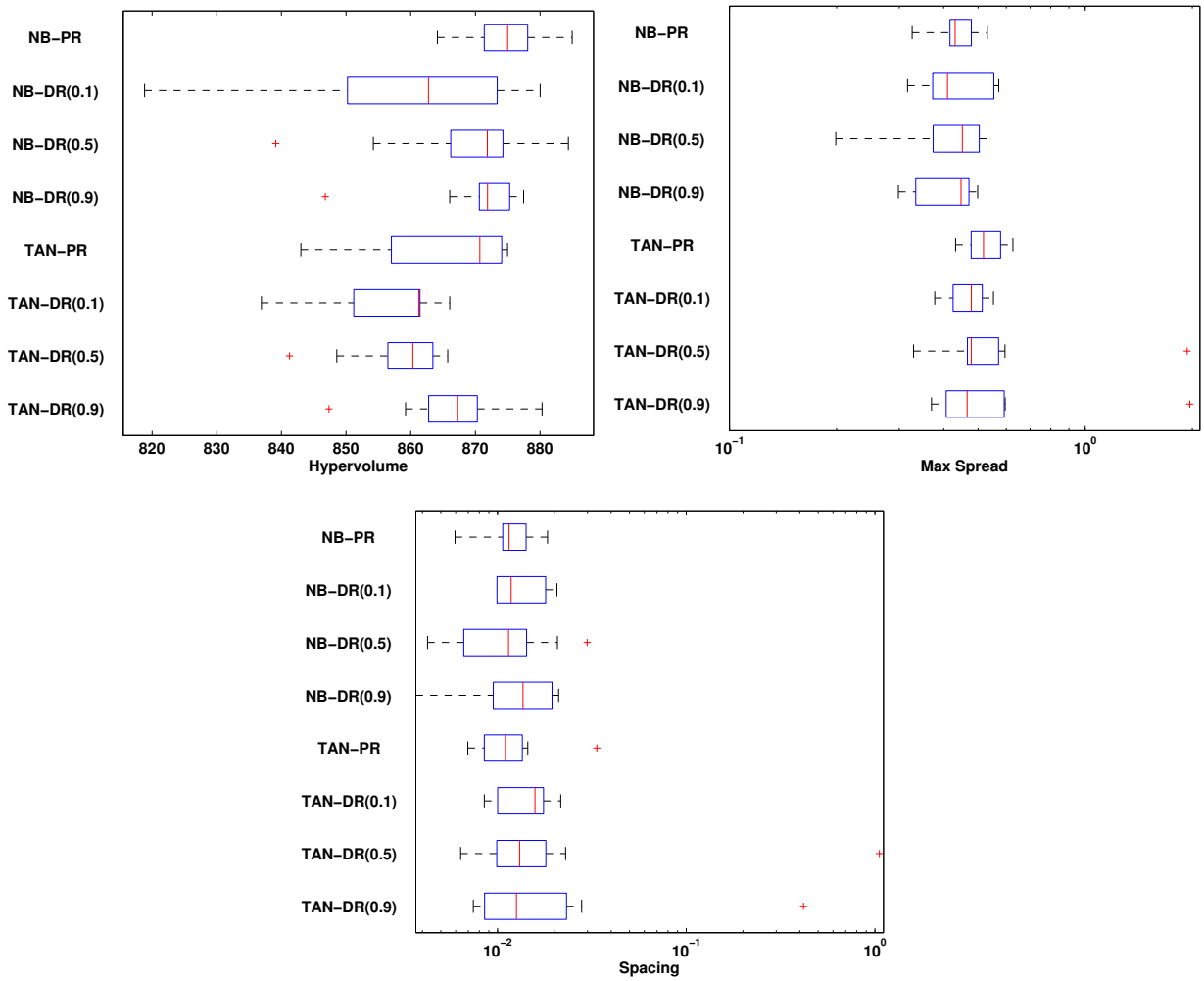



Figure 12.2: Hypervolume, maximum spread and spacing of the final Pareto sets obtained for NB and TAN classifiers on WDBC dataset using PR and DR ranking methods in MBN-EDA.

dataset, the quality indicators show that the feature subsets found for NB classifier give a better overall classification performance. It seems that with TAN classifier the level of noise in objective values is higher. A closer look at the feature subsets in the final Pareto sets also show that fewer features are selected for the NB classifier (Figure 12.6). Moreover, for this classifier usually a similar number of features are selected in the final Pareto sets found by MBN-EDA with both PR and DR methods (the latter with different $\alpha$ values). For the TAN classifier, on the other hand, the feature subsets in the final Pareto set are larger and the results with different ranking methods, especially for the Hill-Valley dataset, are not similar.

The range of objective values in the *aggregation* of the final Pareto fronts found in all of the runs, when using different ranking methods are given in Table 12.3. These aggregated Pareto fronts are obtained by taking the non-dominated solutions of the union of the Pareto sets found in 10 different runs. For the WDBC dataset, the range of objective values are small and very close to their optimal values (considering each objective individually). The sensitivity and F1 measure of the solutions in the final Pareto sets seem to be slightly

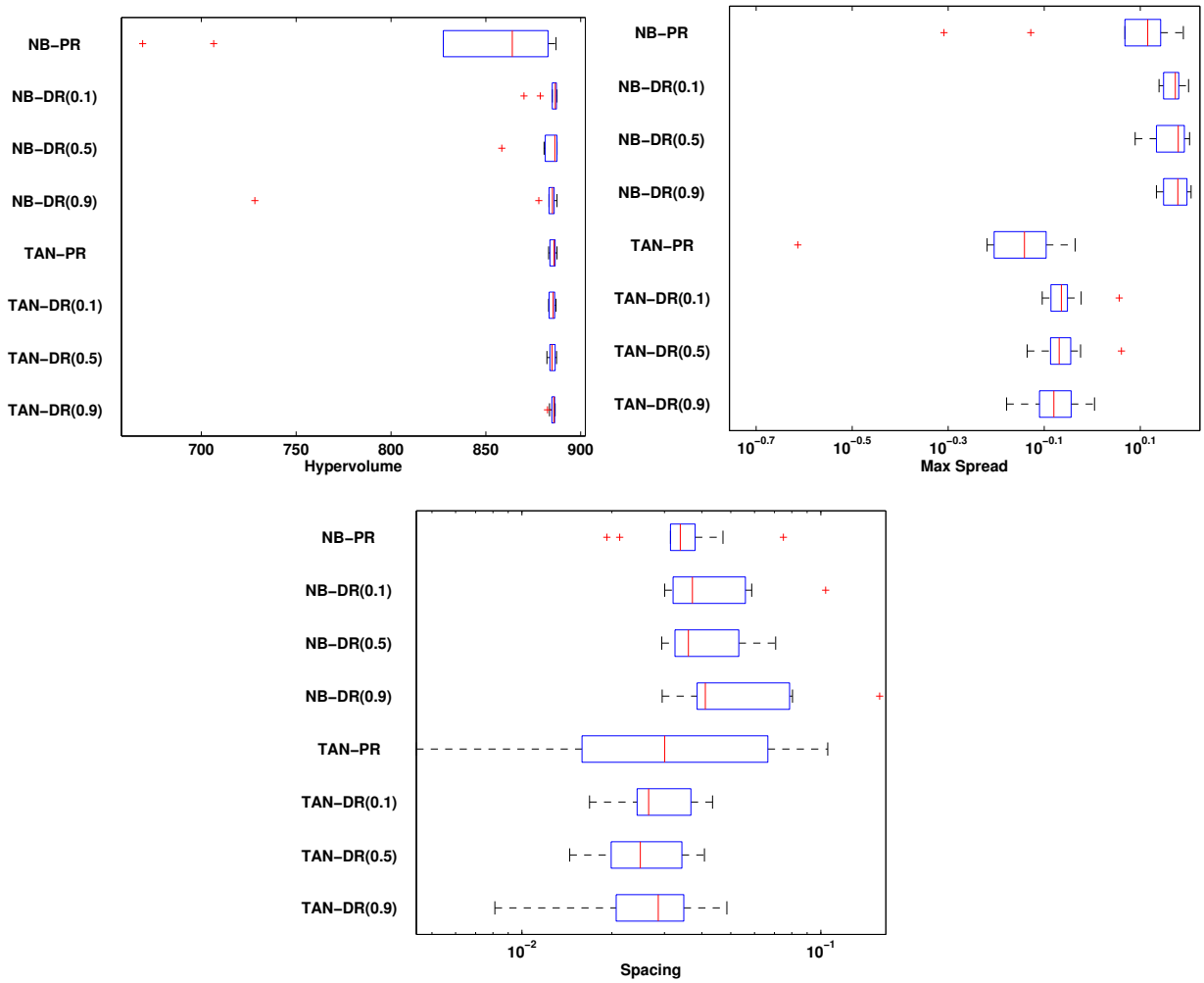



Figure 12.3: Hypervolume, maximum spread and spacing of the final Pareto sets obtained for NB and TAN classifiers on Ozone dataset using PR and DR ranking methods in MBN-EDA.

worse than other performance measures.

As the noise in objective values increases for the Ozone and Hill-Valley datasets, making these problems more difficult, the range of objective values of the Pareto fronts also increases and gets farther from the optimal values. First, for the Ozone dataset, the highly unbalanced number of samples (less than 3% of the samples are positive) affect the sensitivity, precision and F1 measure functions which are based on the number of TP classifications. This influence can be specially observed when using TAN classifier to evaluate solutions, where despite very small sensitivity, they have small Brier scores, explaining the good results of hypervolume indicator for this combination of dataset and classifier.

Second, the sensitivity and specificity of the solutions found for the Hill-Valley dataset, cover a large range of possible values of these objectives, whereas their AUC and Brier score indicate poor performances. This shows that for larger search spaces, sensitivity and specificity functions take priority over the other objectives in the optimization process. Thus, considering several performance measures for evaluating the feature subsets allows

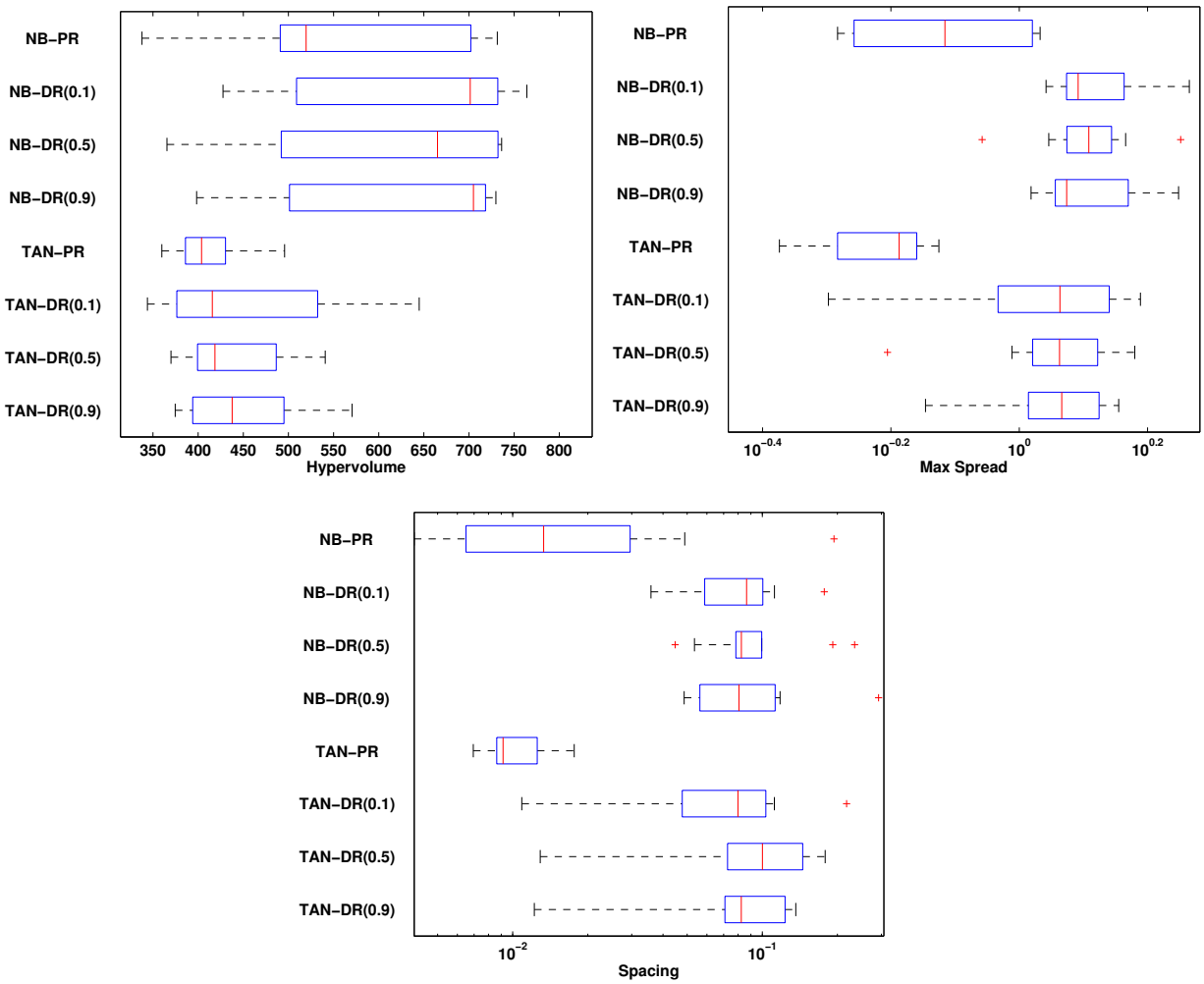



Figure 12.4: Hypervolume, maximum spread and spacing of the final Pareto sets obtained for NB and TAN classifiers on Hill-Valley dataset using PR and DR ranking methods in MBN-EDA.

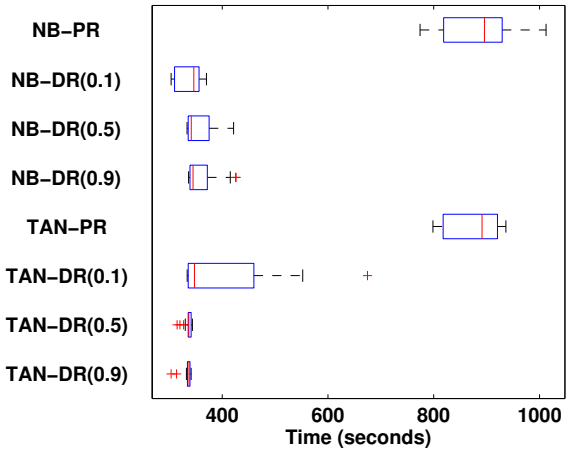


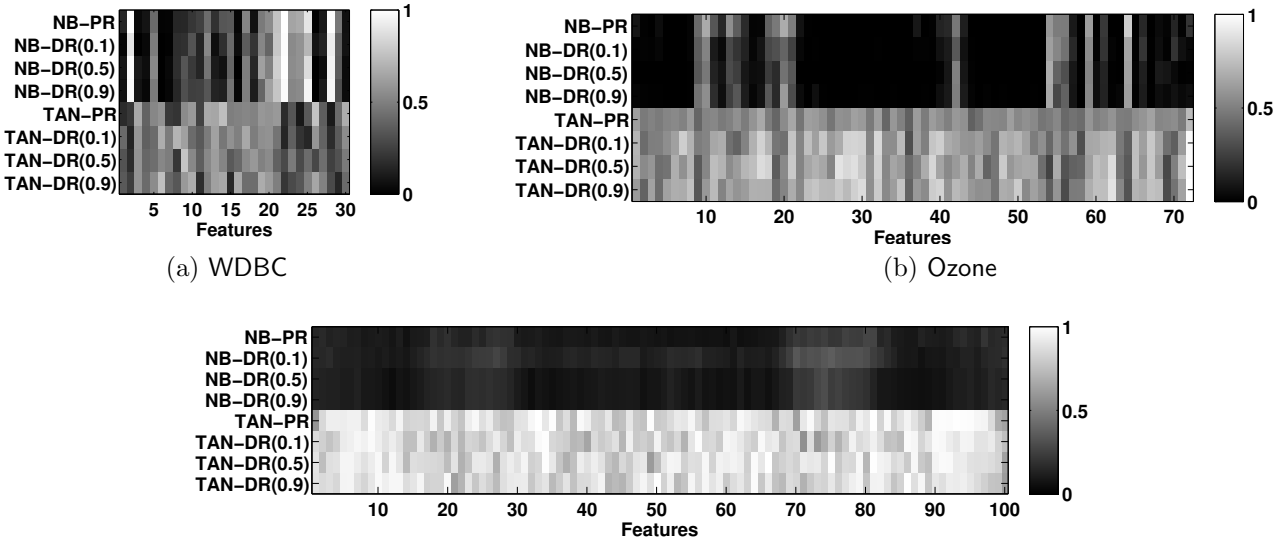

Figure 12.5: Average solution ranking time in each generation of MBN-EDA with PR and DR ranking methods.

Table 12.3: The range of objective values of the solutions in the aggregated Pareto set, obtained from the union of the final Pareto sets of 10 independent MBN-EDA runs, for the three considered datasets.

| WDBC | | $f_{AUC}$ | $f_{sens}$ | $f_{spec}$ | $f_{prec}$ | $f_{F1}$ | $f_{Brier}$ |
|---|---|---|---|---|---|---|---|
| NB-PR | Min. | 0.974 | 0.897 | 0.958 | 0.917 | 0.912 | 0.038 |
| | Max. | 0.998 | 0.970 | 0.994 | 0.990 | 0.961 | 0.111 |
| NB-DR(0.1) | Min. | 0.978 | 0.864 | 0.948 | 0.920 | 0.903 | 0.068 |
| | Max. | 0.991 | 0.937 | 0.989 | 0.980 | 0.944 | 0.097 |
| NB-DR(0.5) | Min. | 0.980 | 0.911 | 0.968 | 0.936 | 0.923 | 0.054 |
| | Max. | 0.987 | 0.955 | 0.984 | 0.967 | 0.954 | 0.084 |
| NB-DR(0.9) | Min. | 0.979 | 0.920 | 0.957 | 0.924 | 0.925 | 0.059 |
| | Max. | 0.991 | 0.941 | 0.989 | 0.982 | 0.954 | 0.090 |
| TAN-PR | Min. | 0.958 | 0.838 | 0.912 | 0.851 | 0.856 | 0.077 |
| | Max. | 0.994 | 0.958 | 0.978 | 0.957 | 0.940 | 0.166 |
| TAN-DR(0.1) | Min. | 0.966 | 0.875 | 0.918 | 0.856 | 0.863 | 0.085 |
| | Max. | 0.988 | 0.944 | 0.961 | 0.935 | 0.936 | 0.154 |
| TAN-DR(0.5) | Min. | 0.969 | 0.874 | 0.903 | 0.848 | 0.861 | 0.081 |
| | Max. | 0.988 | 0.937 | 0.972 | 0.952 | 0.940 | 0.179 |
| TAN-DR(0.9) | Min. | 0.976 | 0.880 | 0.924 | 0.891 | 0.896 | 0.030 |
| | Max. | 0.996 | 0.979 | 0.990 | 0.981 | 0.979 | 0.132 |

| Ozone | | $f_{AUC}$ | $f_{sens}$ | $f_{spec}$ | $f_{prec}$ | $f_{F1}$ | $f_{Brier}$ |
|---|---|---|---|---|---|---|---|
| NB-PR | Min. | 0.763 | 0.000 | 0.807 | 0.000 | 0.000 | 0.054 |
| | Max. | 0.902 | 0.696 | 1.000 | 0.700 | 0.350 | 0.341 |
| NB-DR(0.1) | Min. | 0.720 | 0.000 | 0.818 | 0.000 | 0.000 | 0.048 |
| | Max. | 0.919 | 0.783 | 1.000 | 0.433 | 0.360 | 0.309 |
| NB-DR(0.5) | Min. | 0.793 | 0.000 | 0.798 | 0.000 | 0.000 | 0.052 |
| | Max. | 0.916 | 0.674 | 1.000 | 0.527 | 0.314 | 0.364 |
| NB-DR(0.9) | Min. | 0.777 | 0.000 | 0.824 | 0.000 | 0.000 | 0.054 |
| | Max. | 0.922 | 0.748 | 1.000 | 0.662 | 0.423 | 0.309 |
| TAN-PR | Min. | 0.686 | 0.000 | 0.992 | 0.000 | 0.000 | 0.051 |
| | Max. | 0.861 | 0.092 | 1.000 | 0.467 | 0.124 | 0.073 |
| TAN-DR(0.1) | Min. | 0.721 | 0.000 | 0.990 | 0.000 | 0.000 | 0.052 |
| | Max. | 0.845 | 0.164 | 1.000 | 0.467 | 0.208 | 0.076 |
| TAN-DR(0.5) | Min. | 0.757 | 0.000 | 0.989 | 0.000 | 0.000 | 0.054 |
| | Max. | 0.842 | 0.130 | 1.000 | 0.600 | 0.157 | 0.070 |
| TAN-DR(0.9) | Min. | 0.690 | 0.000 | 0.991 | 0.000 | 0.000 | 0.052 |
| | Max. | 0.874 | 0.096 | 1.000 | 0.467 | 0.140 | 0.077 |

| Hill-Valley | | $f_{AUC}$ | $f_{sens}$ | $f_{spec}$ | $f_{prec}$ | $f_{F1}$ | $f_{Brier}$ |
|---|---|---|---|---|---|---|---|
| NB-PR | Min. | 0.465 | 0.073 | 0.228 | 0.317 | 0.118 | 0.526 |
| | Max. | 0.546 | 0.741 | 0.903 | 0.591 | 0.548 | 0.904 |
| NB-DR(0.1) | Min. | 0.461 | 0.000 | 0.057 | 0.000 | 0.000 | 0.500 |
| | Max. | 0.543 | 0.924 | 1.000 | 0.594 | 0.641 | 0.933 |
| NB-DR(0.5) | Min. | 0.449 | 0.000 | 0.082 | 0.000 | 0.000 | 0.500 |
| | Max. | 0.541 | 0.899 | 1.000 | 0.585 | 0.621 | 0.932 |
| NB-DR(0.9) | Min. | 0.438 | 0.000 | 0.000 | 0.000 | 0.000 | 0.500 |
| | Max. | 0.546 | 1.000 | 1.000 | 0.578 | 0.675 | 0.945 |
| TAN-PR | Min. | 0.488 | 0.343 | 0.346 | 0.457 | 0.387 | 0.513 |
| | Max. | 0.591 | 0.723 | 0.665 | 0.598 | 0.617 | 0.605 |
| TAN-DR(0.1) | Min. | 0.444 | 0.000 | 0.000 | 0.000 | 0.000 | 0.501 |
| | Max. | 0.581 | 1.000 | 1.000 | 0.573 | 0.690 | 0.682 |
| TAN-DR(0.5) | Min. | 0.480 | 0.000 | 0.000 | 0.000 | 0.000 | 0.500 |
| | Max. | 0.582 | 1.000 | 1.000 | 0.579 | 0.685 | 0.690 |
| TAN-DR(0.9) | Min. | 0.462 | 0.000 | 0.000 | 0.000 | 0.000 | 0.499 |
| | Max. | 0.575 | 1.000 | 1.000 | 0.576 | 0.685 | 0.697 |




(a) WDBC (b) Ozone

(c) Hill-Valley

Figure 12.6: The rate of selecting each feature in the solutions of the final Pareto sets approximated for NB and TAN classifiers on the three considered datasets using different ranking methods in MBN-EDA. The rates are averaged over 10 independent runs.

us to inspect the final solutions from different points of view, which would not be possible when using only one or two objectives.

### 12.5.3 Comparison with GA

To evaluate the proposed joint model estimation in MBN-EDA, we compare it with a standard multi-objective GA (MGA), similar to the EMO algorithms reviewed in Section 12.2, and study their optimization performance for FSS. Since the recombination operators used in MGA do not require a large population as in the probabilistic modeling of MBN-EDA, we have set the population size of MGA to $N = 300$ and allowed the algorithm to evolve for more generations by setting the maximum number of generations to 350. Thus, while MGA and MBN-EDA are using two different strategies for evolution, both of them have access to similar resources when considering the maximum number of function evaluations. MGA considered in the experiments employs a two-point crossover and bit-flip mutation with probabilities $P_{cross} = 0.8$ and $P_{mut} = 1/n$, respectively. The rest of parameters like the selection ratio are set similar to MBN-EDA as described in the previous section.

Figures 12.7–12.9 compare the final Pareto sets approximated by MGA and MBN-EDA on each of the three datasets when using the NB classifier, with respect to different quality indicators. Very similar results are also obtained for the TAN classifier. The figures show that the Pareto sets found by MBN-EDA are better than or comparable to those obtained by MGA on all datasets according to all quality indicators. Especially, for the WDBC and Ozone datasets, the hypervolume of the Pareto fronts approximated by MBN-EDA is considerably better. This indicates that, although MBN-EDA evolves in fewer generations, it is able to perform a more effective search using its joint probabilistic

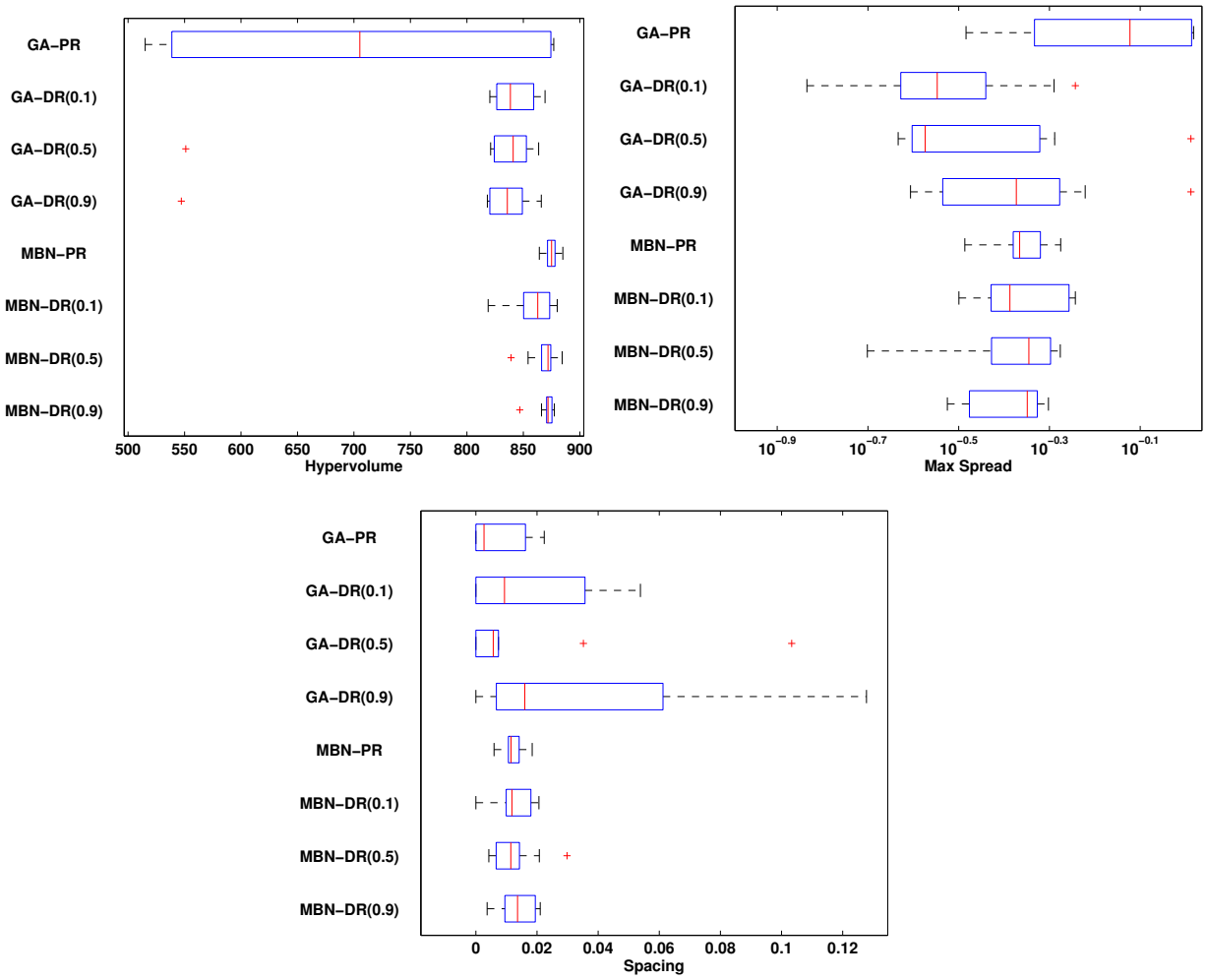



Figure 12.7: Hypervolume, maximum spread and spacing of the final Pareto sets approximated for WDBC dataset with NB classifier, using MGA and MBN-EDA (denoted as MBN).

modeling.

Furthermore, these figures show that the Pareto sets found by MGA using the DR method have better hypervolume values than those found using the PR method, when considering the overall behavior on all datasets. This suggests that, in spite of the method used to explore the search space, the solution ranking provided by DR method can often help to converge to better Pareto fronts in the noisy FSS problem.

### 12.5.4 Analysis of Joint Probabilistic Modeling

In this section, the two constituent parts of the joint probabilistic modeling in MBN-EDA are studied during evolution. We are especially interested in the information encoded in the model about the intrinsic properties of each of the FSS problems. For this purpose, we consider the models estimated using the DR method with a dominance degree of $\alpha = 0.9$. First, the set of variables selected in the first part of joint model estimation using RRMs are examined. Figure 12.10 shows the selected variables during different

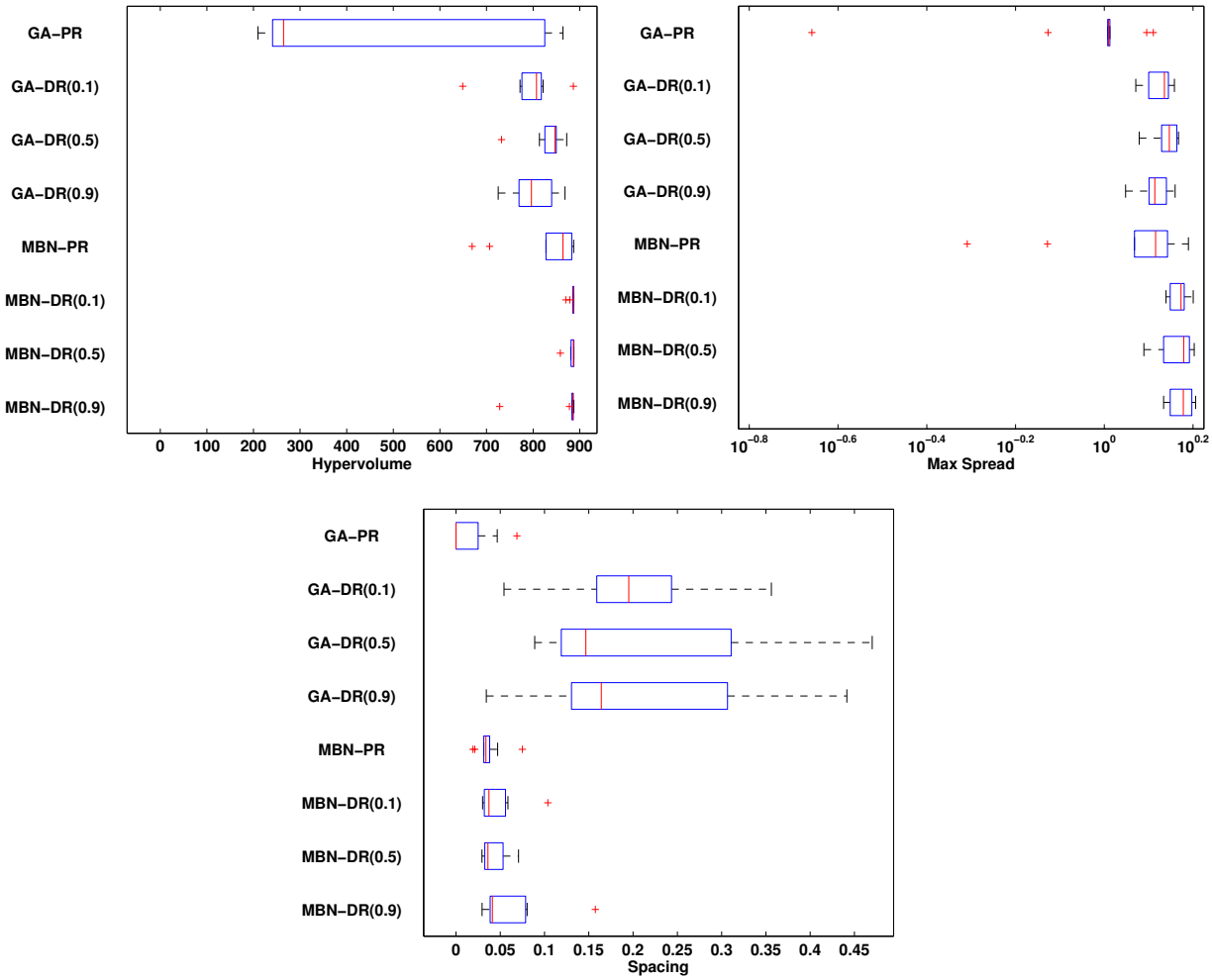



Figure 12.8: Hypervolume, maximum spread and spacing of the final Pareto sets approximated for Ozone dataset with NB classifier, using MGA and MBN-EDA (denoted as MBN).

generations of evolution, for the tested datasets and classifiers. It can be seen that for NB classifier, initially most of the variables are selected in the individuals and during evolution, gradually the set of the chosen variables becomes smaller (except for Hill-Valley dataset which does not exactly follow this pattern). On the contrary, for TAN classifier the variable selection frequency usually increases over time.

Here, the selection frequency of variables determines their relevance to the computation of objective values. Unlike the feature selection frequency obtained from the solutions, this relevance is not affected only by a certain value (e.g. a zero value) of the variables. In other words, both inclusion and exclusion of a specific feature influence the approximation of its relevance to the objectives.

Another part of the study considers the MBN structures estimated in the second part of joint modeling algorithm. As it is expected and explained in Section 10.6, considerably more arcs are added to the class and bridge subgraphs of MBN (i.e. between objectives and between variables and objectives) than to the feature subgraph, indicating the importance of these relations. Here, we depict only the class subgraphs of the estimated

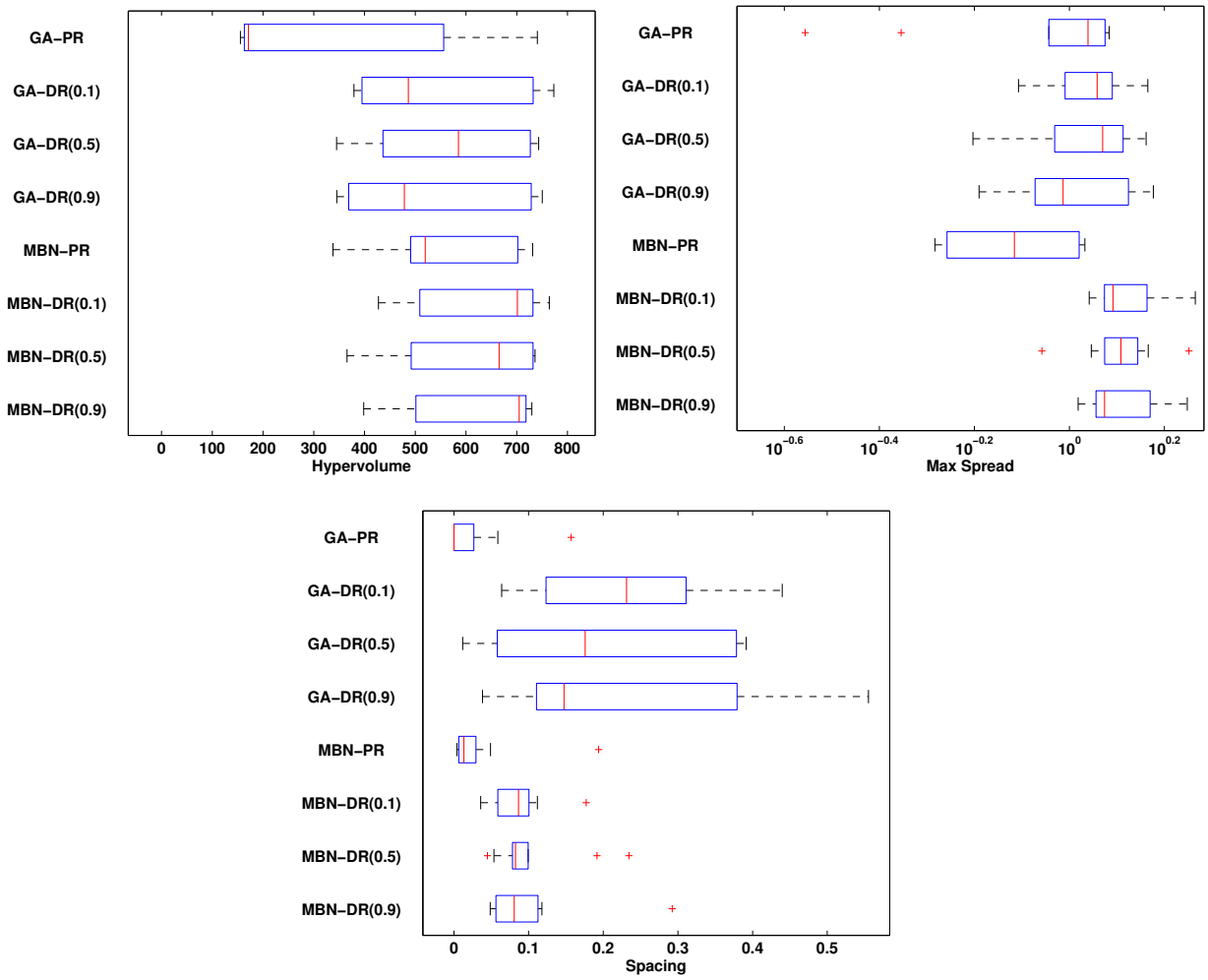
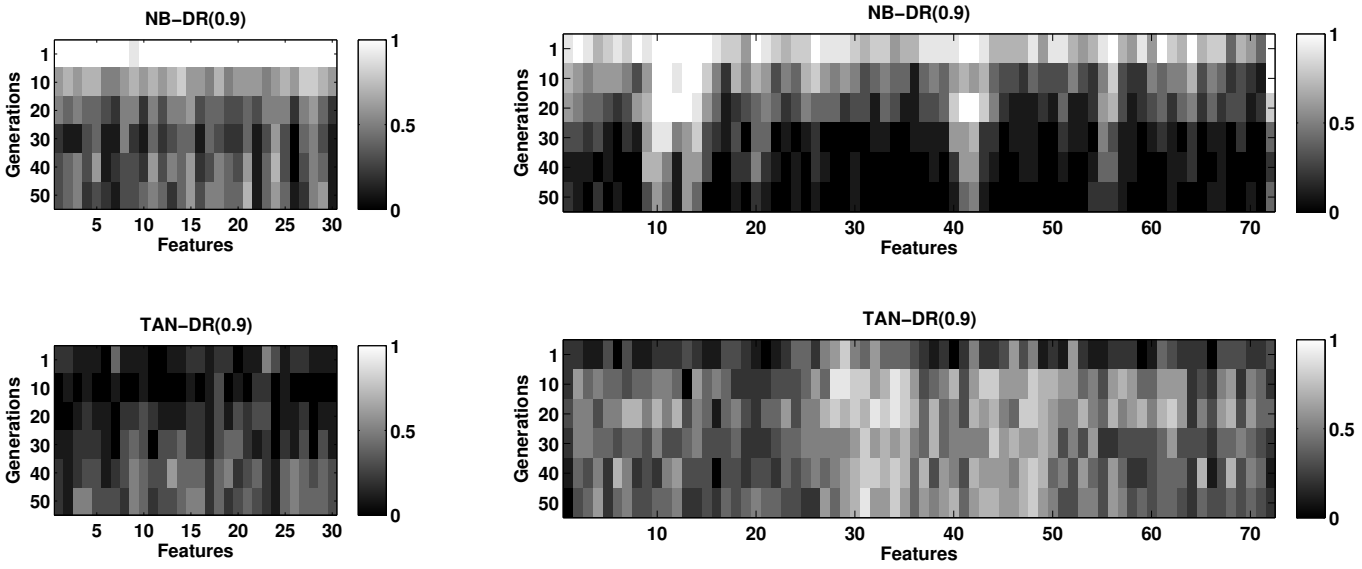



Figure 12.9: Hypervolume, maximum spread and spacing of the final Pareto sets approximated for Hill-Valley dataset with NB classifier, using MGA and MBN-EDA (denoted as MBN).

MBNs to preserve the space. Figure 12.11 shows the overall frequency of arcs in the class subgraph of the estimated MBN structures in all generations of MBN-EDA and in 10 independent runs, for the tested datasets and classifiers.

It can be seen that certain patterns of interaction between objectives have a higher frequency of occurrence in FSS on the tested datasets. These include the dependencies between sensitivity, specificity and precision, between AUC and Brier score, and between sensitivity and F1 function. Some of these relationships can be easily approved by looking at the definitions of objective functions in Equation (12.4). For example, F1 measure is the harmonic mean of sensitivity and precision, and thus any information on the value of one of these objectives can be used to approximate the value of the other.

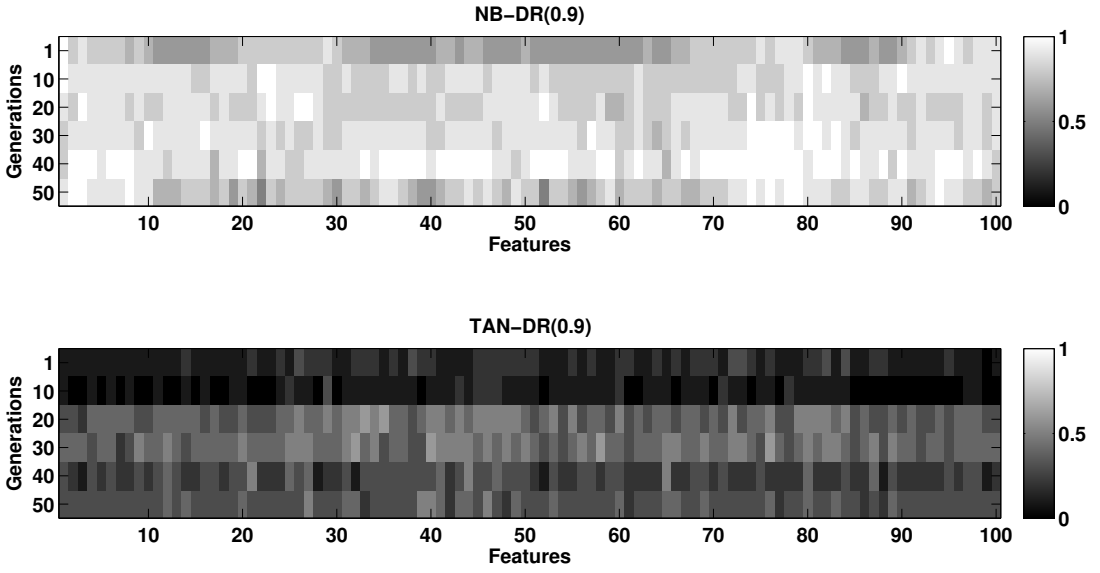
An interesting observation is the role of classifiers in detecting the interactions between objectives. It seems that the proposed joint model estimation is able to identify these kind of relationships better when the TAN classifier is used for solution evaluation. One explanation for this behavior is that for the TAN classifier the sets of variables selected in the first part of model learning using RRMs are smaller, especially at the early generations




(a) WDBC




(b) Ozone

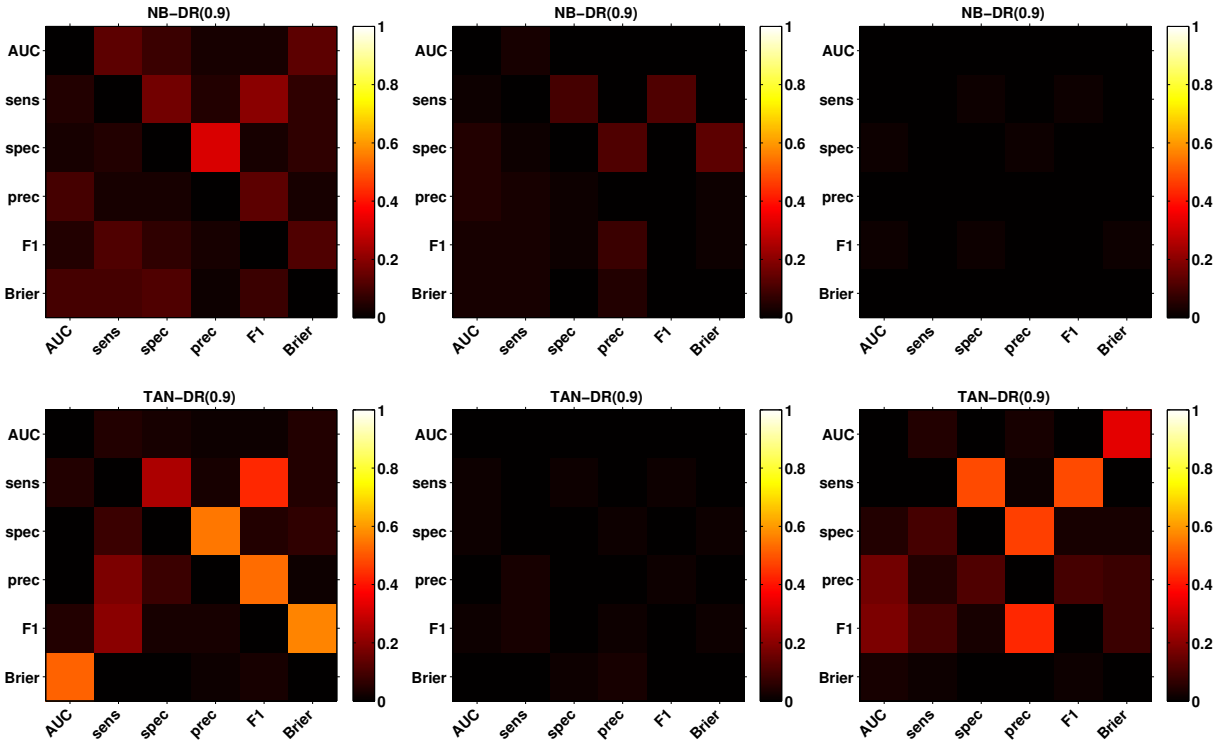



(c) Hill-Valley

Figure 12.10: The average frequency of selecting variables using RRMs in the first part of joint model estimation, along different generations of MBN-EDA and in 10 independent runs.

of the evolution, where the algorithm is detecting the direction of movement in the search space. This allows to filter out variables that would introduce noise to the MBN induction process, which in turn helps to detect the relationships between objectives.

For some of the problem instances under study, the probability of recovering the relationships between objectives in the joint probabilistic model is very low. This situation can be observed for Ozone dataset with TAN classifier (bottom-middle of Figure 12.11) and Hill-Valley dataset with NB classifier (top-right of Figure 12.11). These cases show two possible situations where objective relationships are not considered to be important




(a) WDBC (b) Ozone (c) Hill-Valley

Figure 12.11: The average frequency of arcs in the class subgraph of the MBNs estimated in all generations of MBN-EDA and in 10 independent runs.

for optimization by MBN-EDA. In the latter case, most of the variables are selected for inclusion in MBN estimation by the first part of joint modeling algorithm, in all generations (see Figure 12.10). As it was already explained, this directly affects the detection of relationships in the class subgraph. In the former case, although a relatively smaller number of variables are selected for inclusion in MBN, the high level inconsistency of the objective values due to noise (especially with the unbalanced dataset) makes it hard for the MBN induction algorithm to detect any interactions.

## 12.6 Conclusions

An adaptation of MBN-EDA for FSS problem based on joint modeling of variables and objectives, when the noise in objective values is represented as intervals was proposed in this chapter. The algorithm was applied for FSS in classification, although the method can be easily adapted for FSS in clustering. To deal with the inherent noise in the estimation of objective values, the proposed algorithm employs a ranking method that is able to order the solutions in the population when objective values are given as intervals. Based on this ordering, a subset of promising solutions were selected for model estimation.

A two-step learning method was proposed for estimating a joint probabilistic model of variables and objectives from the selected solutions. In the first step, the variable selection property of $\ell_1$-regularization technique was employed to select a subset of variables with higher relevance to the objectives. This helped to simplify MBN estimation by reducing

the space of possible structures and provided an initial approximation of the bridge subgraph structure of MBN. In the second step, an MBN was estimated encoding the objectives and the set of selected variables.

We defined a six objective optimization problem and used the proposed algorithm to select feature subsets for two Bayesian classifiers, NB and TAN, in three different datasets having an increasing number of features. The experimental results showed that, although requiring considerably less time, the Pareto sets approximated with the proposed DR method were comparable or better than the Pareto sets found using the well-known PR method. Moreover, the comparison of results with those of a standard multi-objective GA showed that the proposed algorithm is able to obtain better Pareto sets in terms of both convergence and diversity.

The estimated joint probabilistic models were also used to analyze the interactions between objectives and variables. We saw different variable selection behaviors with each of the classifiers which resulted in detecting different patterns of objective interactions. It was observed that, though the level of noise in the objective values is higher when using TAN classifier for solution evaluation, the variable selection method deployed in the first part of the model estimation can identify these relationships.

# Part IV

# Conclusions

# Chapter 13

# Conclusions and Future Lines of Research

In this section, we first review the work presented throughout this thesis, and then list the main results of our study. At the end, we also give some lines of future research regarding the presented work. The main contributions of the thesis are summarized in Table 13.1.

## 13.1 Thesis Overview

The inherent properties of EDAs and the way they solve optimization problems make them very interesting, and though they are a relatively new class of EAs, many works have studied these algorithms from both theoretical and application-based points of view, and they are still topic of active research. Probabilistic modeling, which is the essential part of EDAs, is one of the topics that requires further and a more thorough investigation. In this thesis, we proposed and studied the use of regularization techniques in the model estimation of EDAs as a promising method for both continuous and multi-objective optimization.

Two main approaches for regularized model learning in EDAs was presented in Chapter 5. In the first approach, an estimation of the model structure is obtained by explicitly using regularization to capture the dependencies between variables, and then the model parameters are estimated separately. In the second approach, regularization techniques are applied to obtain an estimation of the probabilistic model, which is an MGD. The proposed methods were analyzed considering different aspects when used for estimating models from the populations generated by sampling a number of Gaussian models with an increasing level of interactions between the variables. The resulting RegEDA was then applied to continuous function optimization with a high-dimensional setting for its population size in Chapter 6, and the achieved results were analyzed by comparing them with those of several other well-known Gaussian-based EDAs.

We then used $\ell_1$-regularized modeling for obtaining a decomposition of problem variables in Chapter 7, and to learn a marginally factorized probability distribution encoded in a GMRF model. The proposed algorithm first identifies the (in)dependencies between variables and then uses their interaction strengths to divide the variables into a number of disjoint groups with the AP clustering algorithm which can automatically identify the

Table 13.1: A summary of the contributions and algorithms proposed in the thesis and their domains of application.

| | Single-Objective | | Multi-Objective | | | |
|---|---|---|---|---|---|---|
| | | | Noiseless | | Noisy | |
| | Continuous | | | | | Discrete |
| Algorithm | RegEDA | GMRF-EDA | JGBN-EDA | | MBN-EDA | |
| Probabilistic Model | MGD/GMRF | Marginally Factorized GMRF | BN | | MBN | |
| Proposal | Regularized Model Learning | Regularized Decomposition Learning | Joint Variable-Objective Modeling | | | |
| | | | | | $\alpha$-Degree Pareto Dominance Based Ranking | |
| Regularized Structure Learning | • | • | | | | • |
| Regularized Parameter Learning | • | | • | • | • | |
| Chapter | 5, 6 | 7 | 9 | 10 | 11 | 12 |
| Publication | [Karshenas et al., 2013b] | [Karshenas et al., 2012a] | [Karshenas et al., 2011b] | [Karshenas et al., 2012b] | [Karshenas et al., 2013a] | |

number of clusters. The behavior of GMRF-EDA was then tested on a continuous additively decomposable function and two instances of the off-lattice protein folding model.

In Chapter 9, joint variable-objective probabilistic modeling was proposed as a novel approach for multi-objective optimization with EDAs. We presented methods for learning and sampling the joint probabilistic model, which was encoded in a BN. Specifically, in continuous domains we used a regularized estimation of GBN to address the increased dimensionality due to the inclusion of objectives in the joint probabilistic model. The proposed JGBN-EDA was then compared against a similar GBN-based EDA, which does not consider objectives in its probabilistic model, on a set of benchmark MOPs to study the effect of incorporating the objective values in probabilistic modeling of multi-objective EDAs.

This study was further extended in Chapter 10 by using MBNs for encoding three types of interactions in the joint probabilistic models and specific solution ranking methods proposed in the literature for many-objective optimization. The performance of the resulting algorithm, which is called MBN-EDA, was tested in an extensive experimental study on two sets of benchmark MOPs with different number of objectives, and compared with several other state-of-the-art algorithms. Moreover, we took a closer look at the joint probabilistic models estimated in MBN-EDA and analyzed the ability of this algorithm to approximate the MOP structure.

In Chapter 11, the $\alpha$-degree Pareto dominance relation was defined to deal with the noise in the objective values of MOPs, given as intervals. We studied some of the properties of this dominance relation when noisy objective values are given as confidence intervals, and used it to develop a modified version of the non-dominated sorting algorithm for ranking the solutions in the presence of noise in the objective values. This ranking method was compared to probabilistic ranking, a widely used noise handling method

in EMO algorithm which can also be applied to objective values given as intervals, to optimize a set of noisy MOPs with three different levels of Gaussian noise using MBN-EDA.

Finally in Chapter 12, we proposed a version of MBN-EDA for multi-objective FSS in classification, when several measures of classification performance are considered as objectives. The different quality values obtained in the cross-validation evaluation of the feature subsets were used to compute confidence intervals as the objective values, and a degree ranking method based on the $\alpha$-degree Pareto dominance was proposed to order the solutions. To learn a joint variable-objective probabilistic model, a two-step estimation approach was proposed. In the first step, $\ell_1$-regularization is used to identify the variables related to each objective, and a subset of variables which have more interactions with the objective are selected. In the second step, a joint model of objectives and selected variables is estimated in an MBN, and marginal univariate probability distributions are estimated for the rest of variables. The resulting algorithm was then tested on a number of datasets with increasing number of features, when using two different Bayesian classifier, and its constituent parts for solution ranking and reproduction were compared with other commonly used methods. The estimated model structures during evolution on each tested dataset-classifier combination were also analyzed.

The discussions in the thesis covered a broad range of topics in a variety of domains and applications. Therefore, to introduce the context of study, the main concepts and the proposed methods and approaches were reviewed in each case. This reviews include: two main types of PGMs, probabilistic modeling in EAs, BN learning and inference related optimization problems solved with EAs, regularization techniques for estimating different types of models, multi-objective EDAs and their ranking methods and probabilistic models, methods for noisy optimization with EMO algorithms, and the methods used for multi-objective FSS in classification and clustering with EMO algorithms.

## 13.2 Main Results

Although we have given some conclusions at the end of each chapter, summarizing the contributions and achievements, there are a number of conclusions that can be pointed out as the main results of the research conducted in this thesis. Following the organization of the thesis, these findings can be divided into two parts. Regarding continuous function optimization with EDAs, the main results are:

- The analysis of the proposed methods for regularized model estimation with high-dimensional settings for the population showed the usefulness of each technique for continuous optimization of a different type of problem with EDAs. Some of the methods are able to detect many interactions in a short time, whereas others require more time to obtain sparser model estimations. It was shown that the level of sparsity of the interactions between problem variables can decrease the quality of the models obtained with regularized estimation.

- We tested RegEDA with different methods for regularized model estimation on continuous problems with an increasing number of variables in a high-dimensional setting, simulated with a population size that is logarithmic in the number of vari-

ables. The optimization results showed that because of the robust model estimation based on regularization, RegEDA is less affected by the curse of dimensionality phenomenon in comparison to several other Gaussian-based EDAs, and is able to achieve significantly better results on larger dimensions of many of the tested continuous functions.

- The variable selection property of $\ell_1$-regularization can be a promising approach for linkage learning in EDAs. The method we proposed based on this type of regularization for estimating a marginally factorized probability distribution, often known as MPM in EDA literature, was shown to be more effective than estimating non-marginal factorized models or models with complete variable independence assumption, in the optimization of continuous additively decomposable problems.

The above results also apply to EDAs used for multi-objective optimization. Yet, there are a number of interesting findings in our study of multi-objective optimization with EDAs:

- Incorporating information about objectives into model estimation and sampling of EDAs proved to be useful for a more effective multi-objective optimization, and allowed achieving significantly better results on several MOPs in comparison to other EMO algorithms that ignore this kind of information when generating new solutions. The proposed joint variable-objective probabilistic modeling approach provides a systematic way for modeling the relationships between objectives which is often needed for many-objective optimization. We showed that with this type of model estimation, the resulting EDA is also able to implicitly obtain a good approximation of the MOP structure in a probabilistic framework, which can be an interesting alternative to a set of solutions for DMs.

- The study of the estimated probabilistic models showed that the two new types of interactions captured as a result of joint variable-objective modeling, which are the objective-variable and objective-objective relationships, are considerably more important for multi-objective optimization than the variable-variable relationships usually modeled in other EDAs.

- It was shown that $\alpha$-degree Pareto dominance proposed for solution ordering in noisy multi-objective optimization, when objective values are given as intervals, defines a partial relation between the solutions in the search space. Moreover, when noisy objective values are represented as confidence intervals, it was demonstrated that any reduction in the minimum acceptable dominance degree or confidence level does not affect the $\alpha$-degree Pareto dominance relation between the solutions. We showed that the Pareto set approximations obtained using two solution ranking methods based on this dominance relation are better or comparable to those obtained by a standard solution ranking method while having a considerably less computational complexity.

- Relatively good feature subsets can be found in a small number of generations when using multi-objective EDAs for FSS in classification problems. This approach to FSS is more effective for datasets with small-medium dimensionality, due to

the large populations required for datasets with a large number of features. The regularization-based joint probabilistic modeling in this problem allows to obtain an estimation of how different measures of classification performance are related.

## 13.3 Future Lines of Research

There are a number of possibilities for continuing the research carried out in this thesis. In this section we give some hints for further extension of our study of EDAs in continuous and multi-objective optimization.

Continuous optimization has specific properties that makes it very different from discrete optimization of combinatorial problems. For example, a single variable may have a global optimal value with a small basin of attraction and many deceptive local optima. One of the approaches used to improve continuous optimization is the incorporation of gradient information in the search. This allows the optimization algorithm to adapt to the landscape of the problem at hand and increase or decrease its pace. CMA-ES is one of the successful continuous EDAs that has employed this approach. However, as it was shown in Chapter 6, this adaption considerably affects the performance of this algorithm on problems with large dimensionality. On the other hand, regularized model estimation offers a method for learning models with better generalization ability. An interesting line of future research is to combine these two approaches of model estimation for continuous optimization with EDAs. A starting point for this study is to decompose the problem and use an adapted model learning in the subspaces while controlling the overall fitting of the models with regularized estimation. The variable selection property of $\ell_1$-regularization can be especially useful for this purpose. Such an approach is specifically applicable to high-dimensional problems where decomposition becomes necessary for the feasibility and efficiency of the optimization with EDAs.

Another future work of regularized model estimation in EDAs is to explicitly consider nonlinear interactions between variables when learning their relationships. A preliminary study of this type of regularized model estimation for learning discrete MNs was presented in [Santana et al., 2011]. The emergence of new techniques in the statistics community for efficient regularized learning of GGMs considering higher order relationships between variables can be useful in this respect. A closely related topic for further research is to use non-Gaussian assumptions for the variables when obtaining regularized estimations of their relationships.

With the proposed approach for multi-objective optimization based on joint variable-objective modeling, the estimated models encode probabilistic estimations for the values of both variables and objectives. We used these probabilistic estimations to generate new values for the variables. A future line of research is to analyze the optimization performance when the encoded probabilistic estimations are also used to approximate the objective values of the newly generated solutions. Such an approach can be taken, for example, when optimizing MOPs with expensive (e.g. computationally or economically) objective functions [Zhang et al., 2010]. A related future study is to exploit the relationships encoded as a result of objective modeling to reduce the cost of solution evaluation in multi-objective optimization. For example in multi-objective FSS, a combination of filter-based and wrapper-based objectives can be considered. Then, depending

on the objective relationships encoded in the joint probabilistic model, the value of the wrapper-based objectives for the new solutions can be approximated using the estimated model and only the considerably cheaper filter-based objectives need to be evaluated in each generation.

The application of the methods proposed in the thesis to real-world problems is also a potential future work. There are several problems like gene expression analysis in genomics [Bielza et al., 2008] and optimal ordering of tables in text organization [Bengoetxea et al., 2011] which are characterized by high-dimensionality, involving optimization over a large number of variables. Regularization-based EDAs are promising methods for this type of problems. In multi-objective optimization, an interesting research is to evaluate the MOP structures estimated with joint probabilistic modeling for real-world problems against DM's knowledge of the problem. Moreover, preference information of DMs can be incorporated into joint probabilistic modeling to improve optimization performance of the algorithm [di Pierro et al., 2007; Hirsch et al., 2011]. It can be used as conditional independence relations between objectives and variables, or inserted as preferable values for the objectives.

# Appendix A

# List of Abbreviations

| | |
|---|---|
| **AIC** | Akaike's Information Criterion |
| **ANN** | Artificial Neural Network |
| **ANOVA** | Analysis Of Variance |
| **AP** | Affinity Propagation |
| **AUC** | Area Under ROC Curve |
| **BAV** | Best Achieved Value |
| **BDe** | Bayesian Dirichlet equivalence |
| **BIC** | Bayesian Information Criterion |
| **BN** | Bayesian Network |
| **CGN** | Conditional Gaussian Bayesian Network |
| **DAG** | Directed Acyclic Graph |
| **DAGS** | DAG Search |
| **DBN** | Dynamic Bayesian Network |
| **DM** | Decision Maker |
| **DR** | Degree Ranking |
| **DT** | Decision Tree |
| **EA** | Evolutionary Algorithm |
| **EDA** | Estimation of Distribution Algorithm |
| **EM** | Expectation Maximization |
| **EMO** | Evolutionary Multi-objective Optimization |
| **EP** | Evolutionary Programming |
| **ES** | Evolutionary Strategy |
| **FSS** | Feature Subset Selection |
| **GA** | Genetic Algorithm |
| **GBN** | Gaussian Bayesian Network |
| **GGM** | Gaussian Graphical Model |
| **GMRF** | Gaussian Markov Random Field |
| **GP** | Genetic Programming |
| **IGD** | Inverted Generational Distance |
| **kDB** | k-Dependence Bayesian Classifier |
| **MAP** | Maximum A Posteriori |
| **MCMC** | Markov Chain Monte Carlo |

| | |
|---|---|
| **MB** | Markov Blanket |
| **MBN** | Multi-dimensional Bayesian Network |
| **MDL** | Minimum Description Length |
| **MGD** | Multivariate Gaussian Distribution |
| **ML** | Maximum Likelihood |
| **MN** | Markov Network |
| **MOP** | Multi-objective Optimization Problem |
| **MPE** | Most Probable Explanation |
| **MPM** | Marginal Product Model |
| **MTE** | Mixture of Truncated Exponentials |
| **NB** | Naïve Bayes Classifier |
| **NLL** | Negative Log-Likelihood |
| **NN** | Nearest Neighbor |
| **PDAG** | Partial Directed Acyclic Graph |
| **PGM** | Probabilistic Graphical Model |
| **PLS** | Probabilistic Logic Sampling |
| **PR** | Probabilistic Ranking |
| **PSO** | Particle Swarm Optimization |
| **RRM** | Regularized Regression Model |
| **SNB** | Semi-Naïve Bayes Classifier |
| **SVM** | Support Vector Machine |
| **TAN** | Tree Augmented Naïve Bayes Classifier |
| **TSP** | Traveling Salesman Problem |
| **WFG** | Walking Fish Group |

# Bibliography


A. Abraham, L. Jain, and R. Goldberg, editors. *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, Springer, Berlin, 2005.

P. Agarwal and C. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

H. Aguirre and K. Tanaka. A study on the effects of rankings sensitive to density on many-objective MNK landscapes. In *IEEE Congress on Evolutionary Computation (CEC'10), IEEE World Congress on Computational Intelligence (WCCI 2010)*, pages 1089–1096, 2010.

C. Ahn, R. Ramakrishna, and D. Goldberg. Real-coded Bayesian optimization algorithm: Bringing the strength of BOA into the continuous world. In *Sixth Annual Conference on Genetic and Evolutionary Computation (GECCO '04)*, volume 3102 of *Lecture Notes in Computer Science*, pages 840–851. Springer, 2004.

C. W. Ahn and R. S. Ramakrishna. Multiobjective real-coded Bayesian optimization algorithm revisited: Diversity preservation. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 593–600. ACM, 2007.

C. W. Ahn, J. An, and J.-C. Yoo. Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs. *Information Sciences*, 192:109–119, 2012.

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

J. T. Alander. An indexed bibliography of genetic algorithms in medicine. Technical Report 94-1-MEDICINE, University of Vaasa, Finland, 2012.

M. E. Alden. *MARLEDA: Effective Distribution Estimation through Markov Random Fields*. PhD thesis, The University of Texas at Austin, 2007.

G. Andrew and J. Gao. Scalable training of $l_1$-regularized log-linear models. In *24th International Conference on Machine Learning (ICML '07)*, pages 33–40. ACM, 2007.

R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J. Flores, J. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, and P. Larrañaga. A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6):online, DOI: 10.1186/1756–0381–1–6, 2008.

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

D. Arnold and H.-G. Beyer. A general noise model and its effects on evolution strategy performance. *IEEE Transactions on Evolutionary Computation*, 10(4):380–391, 2006.

M. Babbar, A. Lakshmikantha, and D. E. Goldberg. A modified NSGA-II to solve noisy multiobjective problems. In *Genetic and Evolutionary Computation Conference (GECCO'03)*, pages 21–27, 2003.

J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

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

S. Baluja and S. Davies. Using optimal dependency-trees for combinational optimization. In *14th International Conference on Machine Learning*, pages 30–38. Morgan Kaufmann Publishers Inc., 1997.

S. Bandaru and K. Deb. Automated innovization for simultaneous discovery of multiple rules in bi-objective problems. In *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2011.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

O. Barrière, E. Lutton, and P.-H. Wuillemin. Bayesian network structure learning using cooperative coevolution. In *11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pages 755–762. ACM, 2009.

M. Basseur and E. Zitzler. Handling uncertainty in indicator-based multiobjective optimization. *International Journal of Computational Intelligence Research*, 2(3):255–272, 2006.

A. Beck and Y. Eldar. Regularization in regression with bounded noise: A Chebyshev center approach. *SIAM Journal on Matrix Analysis and Applications*, 29(2):606–625, 2008.

E. Bengoetxea and P. Larrañaga. EDA-PSO: A hybrid paradigm combining estimation of distribution algorithms and particle swarm optimization. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 416–423. Springer, 2010.

E. Bengoetxea, T. Miquélez, P. Larrañaga, and J. Lozano. Experimental results in function optimization with EDAs in continuous domain. In Larrañaga and Lozano [2001], chapter 8, pages 181–194.

E. Bengoetxea, P. Larrañaga, C. Bielza, and J. Fernández del Pozo. Optimal row and column ordering to improve table interpretation using estimation of distribution algorithms. *Journal of Heuristics*, 17(5):567–588, 2011.

H.-G. Beyer. *The Theory of Evolution Strategies*, Springer, 2001.

H.-G. Beyer and H.-P. Schwefel. Evolution strategies – comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

P. Bickel and E. Levina. Regularized estimation of large covariance matrices. *Annals of Statistics*, 36(1):199–227, 2008.

C. Bielza, V. Robles, and P. Larrañaga. Estimation of distribution algorithms as logistic regression regularizers of microarray classifiers. *Methods of Information in Medicine*, 16:345–366, 2008.

C. Bielza, M. Gómez, and P. P. Shenoy. A review of representation issues and modeling challenges with influence diagrams. *Omega*, 39(3):227–241, 2011a.

C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011b.

J. Bien and R. J. Tibshirani. Sparse estimation of a covariance matrix. *Biometrika*, 98 (4):807–820, 2011.

M. Bilodeau and D. Brenner. *Theory of Multivariate Statistics*, Springer-Verlag, 1999.

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

R. Blanco, P. Larrañaga, I. Inza, and B. Sierra. Gene selection for cancer classification using wrapper approaches. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(8):1373–1390, 2004.

A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1–2):245–271, 1997.

P. Boonma and J. Suzuki. A confidence-based dominance operator in evolutionary algorithms for noisy multiobjective optimization problems. In *21st International Conference on Tools with Artificial Intelligence (ICTAI'09)*, pages 387–394, 2009.

P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.

P. A. Bosman and D. Thierens. Numerical optimization with real-valued estimation-of-distribution algorithms. In Pelikan et al. [2006a], chapter 5, pages 91–120.

P. A. N. Bosman and E. de Jong. Adaptation of a success story in GAs: Estimation-of-distribution algorithms for tree-based optimization problems. In *Success in Evolutionary Computation*, volume 92 of *Studies in Computational Intelligence*, pages 3–18. Springer, 2008.

P. A. N. Bosman and J. Grahl. Matching inductive search bias and problem structure in continuous estimation of distribution algorithms. *European Journal of Operational Research*, 185(3):1246–1264, 2008.

P. A. N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In *Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *Lecture Notes in Computer Science*, pages 767–776. Springer-Verlag, 2000a.

P. A. N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In *Genetic and Evolutionary Computation Conference (GECCO '00) Workshop*, pages 197–200, 2000b.

P. A. N. Bosman and D. Thierens. Advancing continuous IDEAs with mixture distributions and factorization selection metrics. In *Optimization by Building and Using Probabilistic Models (OBUPM) Workshop at the Genetic and Evolutionary Computation Conference (GECCO '01)*, pages 208–212. ACM, 2001.

P. A. N. Bosman and D. Thierens. Adaptive variance scaling in continuous multi-objective estimation of distribution algorithms. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 500–507. ACM, 2007.

P. A. N. Bosman, J. Grahl, and D. Thierens. Enhancing the performance of maximum–likelihood Gaussian EDAs using anticipated mean shift. In *10th International Conference on Parallel Problem Solving from Nature (PPSN X)*, pages 133–143. Springer, 2008.

R. R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, Universiteit Utrecht, Faculteit Wiskunde en Informatica, 1995.

D. Brockhoff and E. Zitzler. Objective reduction in evolutionary multiobjective optimization: Theory and applications. *Evolutionary Computation*, 17(2):135–166, 2009.

A. Brownlee, J. McCall, Q. Zhang, and D. Brown. Approaches to selection and their effect on fitness modelling in an estimation of distribution algorithm. In *IEEE Congress on Evolutionary Computation (CEC 2008) – IEEE World Congress on Computational Intelligence*, pages 2621–2628. IEEE Computer Society, 2008.

A. E. I. Brownlee. *Multivariate Markov Networks for Fitness Modelling in an Estimation of Distribution Algorithm*. PhD thesis, The Robert Gordon University. School of Computing, 2009.

D. Büche, P. Stoll, R. Dornberger, and P. Koumoutsakos. Multiobjective evolutionary algorithm for the optimization of noisy combustion processes. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 32(4):460–473, 2002.

P. Bühlmann, M. Kalisch, and M. H. Maathuis. Variable selection in high-dimensional linear models: Partially faithful distributions and the PC-simple algorithm. *Biometrika*, 97(2):261–278, 2010.

L. T. Bui, H. A. Abbass, and D. Essam. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Conference on Genetic and Evolutionary Computation (GECCO'05)*, pages 779–785. ACM, 2005a.

L. T. Bui, D. Essam, H. A. Abbass, and D. Green. Performance analysis of evolutionary multi-objective optimization methods in noisy environments. *Complexity International*, 11:29–39, 2005b.

L. T. Bui, H. A. Abbass, and D. Essam. Localization for solving noisy multi-objective optimization problems. *Evolutionary Computation*, 17(3):379–409, 2009.

W. Buntine. Theory refinement on Bayesian networks. In *7th Annual Conference on Uncertainty in Artificial Intelligence (UAI '91)*, pages 52–60. Morgan Kaufmann, 1991.

W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

A. Cano, S. Moral, and A. Salmerón. Novel strategies to approximate probability trees in penniless propagation. *International Journal of Intelligent Systems*, 18(2):193–203, 2003.

G. Casella and E. I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

R. Castelo and T. Kočka. On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574, 2003.

E. Castillo, J. M. Gutiérrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*, Springer, 1997.

F. Chan and S. Chung. Multi-criteria genetic optimization for distribution network problems. *The International Journal of Advanced Manufacturing Technology*, 24:517–532, 2004.

S. Chaudhuri, M. Drton, and T. S. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94(1):199–216, 2007.

B. Chen and J. Hu. An adaptive niching EDA based on clustering analysis. In *IEEE Congress on Evolutionary Computation (CEC '10)*, pages 1–7, 2010.

S.-H. Chen and M.-C. Chen. Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems. *International Journal of Production Economics*, 141(1):24–33, 2013.

Y. Chen, A. Wiesel, and A. Hero. Robust shrinkage estimation of high-dimensional covariance matrices. In *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM '10)*, pages 189–192, 2010.

J. Cheng, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.

S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*, volume 112 of *Lecture Notes in Statistics*, pages 121–130. Springer, 1996.

D. M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, USA, 1994.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

D.-Y. Cho and B.-T. Zhang. Evolutionary optimization by distribution estimation with mixtures of factor analyzers. In *IEEE Congress on Evolutionary Computation (CEC '02)*, volume 2, pages 1396–1401. IEEE Computer Society, 2002.

D.-Y. Cho and B.-T. Zhang. Evolutionary continuous optimization by distribution estimation with variational Bayesian independent component analyzers mixture model. In *Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 212–221. Springer, 2004.

C. A. Coello Coello and N. C. Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6: 163–190, 2005.

C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer, second edition, 2007.

G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

G. Corani, A. Antonucci, and M. Zaffalon. Bayesian networks with imprecise probabilities: Theory and application to classification. Technical Report IDSIA-02-10, Dalle Molle Institute for Artificial Intelligence, Manno, Switzerland, 2010.

D. W. Corne and J. D. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*, pages 773–780. ACM, 2007.

D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Genetic and Evolutionary Computation Conference (GECCO 01)*, pages 283–290. Morgan Kaufmann Publishers, 2001.

M. Costa and E. Minisci. MOPED: A multi-objective Parzen-based estimation of distribution algorithm for continuous problems. In *Second International Conference on Evolutionary Multi-Criterion Optimization (EMO'03)*, volume 2632 of *Lecture Notes in Computer Science*, pages 282–294. Springer, 2003.

C. Cotta and J. Muruzábal. Towards a more efficient evolutionary induction of Bayesian networks. In *7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, pages 730–739. Springer-Verlag, 2002.

C. Cotta and J. Muruzábal. On the learning of Bayesian network graph structures via evolutionary programming. In *Second European Workshop on Probabilistic Graphical Models*, pages 65–72, 2004.

D. Craft and T. Bortfeld. How many plans are needed in an IMRT multi-objective plan database? *Physics in Medicine and Biology*, 53(11):2785–2796, 2008.

N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In *First International Conference on Genetic Algorithms*, pages 183–187. L. Erlbaum Associates Inc., 1985.

A. Cuesta-Infante, R. Santana, J. I. Hidalgo, C. Bielza, and P. Larrañaga. Bivariate empirical and n-variate Archimedean copulas in estimation of distribution algorithms. In *IEEE Congress on Evolutionary Computation (CEC '10)*, pages C–7693, 2010.

R. Daly, Q. Shen, and J. S. Aitken. Learning Bayesian networks: Approaches and issues. *Knowledge Engineering Review*, 26(2):99–157, 2011.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*, Cambridge University Press, 2009.

S. Davies and S. Russell. NP-completeness of searches for smallest possible feature sets. In *AAAI Symposium on Intelligent Relevance*, pages 37–39. AAAI Press, 1994.

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

J. De Bonet, C. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, volume 9, pages 424–430, 1997.

L. M. de Campos, J. A. Gámez, and S. Moral. Partial abductive inference in Bayesian belief networks using a genetic algorithm. *Pattern Recognition Letters*, 20:1211–1217, 1999.

L. M. de Campos, J. A. Gámez, P. Larrañaga, S. Moral, and T. Romero. Partial abductive inference in Bayesian networks: An empirical comparison between GAs and EDAs. In Larrañaga and Lozano [2001], chapter 16, pages 323–341.

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

L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, 2002.

P. A. D. de Castro and F. J. V. Zuben. BAIS: A Bayesian artificial immune system for the effective handling of building blocks. *Information Sciences*, 179(10):1426–1440, 2009.

P. de Waal and L. van der Gaag. Inference and learning in multi-dimensional Bayesian network classifiers. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 501–511, 2007.

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Inc., New York, NY, USA, 2001.

K. Deb and B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

K. Deb and D. Saxena. On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. Technical Report 2005011, Kanpur Genetic Algorithms Laboratory (KanGAL), 2005.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002a.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *IEEE Congress on Evolutionary Computation (CEC '02)*, volume 1, pages 825–830, 2002b.

K. Deb, S. Bandaru, and C. Celal Tutum. Temporal evolution of design principles in engineering systems: Analogies with human evolution. In *Parallel Problem Solving from Nature (PPSN XII)*, volume 7492 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2012.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

F. di Pierro, S.-T. Khu, and D. Savic. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45, 2007.

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

N. Ding, S. Zhou, and Z. Sun. Histogram-based estimation of distribution algorithm: A competent method for continuous optimization. *Journal of Computer Science and Technology*, 23(1):35–43, 2008.

L. Dong, H. Zhang, X. Ren, and Y. li Li. Classifier learning algorithm based on genetic algorithms. *International Journal of Innovative Computing, Information and Control*, 6(4):1973–1981, 2010a.

W. Dong and X. Yao. Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms. *Information Sciences*, 178(15):3000–3023, 2008.

W. Dong, T. Chen, P. Tino, and X. Yao. Scaling up estimation of distribution algorithms for continuous optimization. CoRR, 2011.

X. Dong, D. Ouyang, Y. Ye, S. Feng, and H. Yu. A stable stochastic optimization algorithm for triangulation of Bayesian networks. In *Third International Conference on Knowledge Discovery and Data Mining (WKDD '10)*, pages 466–469, 2010b.

M. Dudík, S. J. Phillips, and R. E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, 8:1217–1260, 2007.

C. Echegoyen, A. Mendiburu, R. Santana, and J. Lozano. Analyzing the probability of the optimum in EDAs based on Bayesian networks. In *IEEE Congress on Evolutionary Computation (CEC '09)*, pages 1652–1659, 2009.

B. Efron. Maximum likelihood and decision theory. *Annals of Statistics*, 10(2):340–356, 1982.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004.

A. Ekbal, S. Saha, and C. S. Garbe. Feature selection using multiobjective optimization for named entity recognition. In *20th International Conference on Pattern Recognition (ICPR '10)*, pages 1937–1940. IEEE Computer Society, 2010.

C. Emmanouilidis. Evolutionary multi-objective feature selection and ROC analysis with application to industrial machinery fault diagnosis. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, EUROGEN 2001,

pages 319–324. International Center for Numerical Methods in Engineering (CIMNE), 2001.

C. Emmanouilidis, A. Hunter, and J. MacIntyre. A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *IEEE Congress on Evolutionary Computation (CEC'00)*, volume 1, pages 309–316, 2000.

C. Emmanouilidis, A. Hunter, J. MacIntyre, and C. Cox. A multi-objective genetic algorithm approach to feature selection in neural and fuzzy modeling. *Journal of Evolutionary Optimization*, 3(1):1–26, 2001.

H. Eskandari and C. Geiger. Evolutionary multiobjective optimization in noisy problem environments. *Journal of Heuristics*, 15(6):559–595, 2009.

R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In *Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 332–339, 1999.

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

J. Fieldsend and R. Everson. Multi-objective optimisation in the presence of uncertainty. In *IEEE Congress on Evolutionary Computation (CEC'05)*, volume 1, pages 243–250, 2005.

J. L. Flores, I. Inza, and P. Larrañaga. Wrapper discretization by means of estimation of distribution algorithms. *Intelligent Data Analysis*, 11:525–545, 2007.

L. J. Fogel. *Artificial Intelligence Through Simulated Evolution*, Wiley, New York, 1966.

L. Fossati, P. Lanzi, K. Sastry, D. Goldberg, and O. Gomez. A simple real-coded extended compact genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC'07)*, pages 342–348, 2007.

C. Fraley and A. E. Raftery. Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of Classification*, 24:155–181, 2007.

B. J. Frey and D. Dueck. Mixture modeling by affinity propagation. In *Advances in neural information processing systems*, volume 18, pages 379–386. MIT Press, 2006.

B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical LASSO. *Biostatistics*, 9(3):432–441, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010a.

J. Friedman, T. Hastie, and R. Tibshirani. Applications of the lasso and grouped lasso to the estimation of sparse graphical models. Technical report, Stanford University, 2010b.

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

M. Frydenberg. The chain graph Markov property. *Scandinavian Journal of Statistics*, 17(4):333–353, 1990.

M. Gallagher, M. R. Frean, and T. Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 840–846. Morgan Kaufmann, 1999.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. EDNA: Estimation of dependency networks algorithm. In *Bio-Inspired Modeling of Cognitive Tasks*, volume 4527 of *Lecture Notes in Computer Science*, pages 427–436. Springer, 2007.

Y. Gao, X. Hu, H. Liu, and Y. Feng. Multiobjective estimation of distribution algorithm combined with PSO for RFID network optimization. In *International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, volume 2, pages 736–739. IEEE Computer Society, 2010.

M. Garza-Fabre, G. Toscano Pulido, and C. Coello Coello. Ranking methods for many-objective optimization. In *MICAI 2009: Advances in Artificial Intelligence*, volume 5845 of *Lecture Notes in Computer Science*, pages 633–645. Springer, 2009.

M. Garza-Fabre, G. Toscano Pulido, and C. Coello Coello. Alternative fitness assignment methods for many-objective optimization problems. In *Artifical Evolution*, volume 5975 of *Lecture Notes in Computer Science*, pages 146–157. Springer, 2010a.

M. Garza-Fabre, G. Toscano Pulido, and C. A. Coello Coello. Two novel approaches for many-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC'10) – IEEE World Congress on Computational Intelligence (WCCI 2010)*, pages 4480–4487, 2010b.

J. L. Gastwirth. The estimation of the Lorenz curve and Gini index. *The Review of Economics and Statistics*, 54(3):306–316, 1972.

D. Geiger and D. Heckerman. Learning Gaussian networks. In *10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 235–243. Morgan Kaufmann, 1994.

E. S. Gelsema. Abductive reasoning in Bayesian belief networks using a genetic algorithm. *Pattern Recognition Letters*, 16:865–871, 1995.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

M. Gen and R. Cheng. *Genetic Algorithms and Engineering Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 2000.

T. Goel, R. Vaidyanathan, R. T. Haftka, W. Shyy, N. V. Queipo, and K. Tucker. Response surface approximation of Pareto optimal front in multi-objective optimization. *Computer Methods in Applied Mechanics and Engineering*, 196(4–6):879–893, 2007.

C. Goh and K. Tan. An investigation on noisy environments in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 11(3):354–381, 2007.

C. Goh, K. Tan, C. Cheong, and Y. Ong. An investigation on noise-induced features in robust evolutionary multi-objective optimization. *Expert Systems with Applications*, 37(8):5960–5980, 2010.

C.-K. Goh and K. C. Tan. *Evolutionary Multi-Objective Optimization in Uncertain Environments: Issues and Algorithms*, Springer, 2009.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 1st edition, 1989.

D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

C. González, J. Lozano, and P. Larrañaga. Mathematical modelling of UMDAc algorithm with tournament selection: Behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning*, 31(3):313–340, 2002.

J. Grahl, S. Minner, and F. Rothlauf. Behaviour of UMDA$_c$ with truncation selection on monotonous functions. In *IEEE Congress on Evolutionary Computation (CEC '05)*, volume 3, pages 2553–2559. IEEE, 2005.

J. Grahl, P. A. N. Bosman, and F. Rothlauf. The correlation-triggered adaptive variance scaling IDEA. In *8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06)*, pages 397–404. ACM, 2006.

D. Greiner, B. Galván, J. Aznárez, O. Maeso, and G. Winter. Robust design of noise attenuation barriers with evolutionary multiobjective algorithms and the boundary element method. In *Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 261–274. Springer, 2009.

L. Grosset, R. LeRiche, and R. Haftka. A double-distribution statistical algorithm for composite laminate optimization. *Structural and Multidisciplinary Optimization*, 31: 49–59, 2006.

P. Grünwald. *The Minimum Description Length Principle and Reasoning Under Uncertainty*. PhD thesis, University of Amsterdam, 1998.

G. Guillén-Gosálbez. A novel MILP-based objective reduction method for multi-objective optimization: Application to environmental problems. *Computers & Chemical Engineering*, 35(8):1469–1477, 2011.

P. Guo, Y. Jia, and M. R. Lyu. A study of regularized Gaussian classifier in high-dimension small sample set case based on MDL principle with application to spectrum recognition. *Pattern Recognition*, 41(9):2842–2854, 2008.

J. Habrant. Structure learning of Bayesian networks from databases by genetic algorithms: application to time series prediction in finance. In *First International Conference on Enterprise Information Systems (ICEIS)*, volume 1, pages 225–231, 1999.

T. Hamdani, J.-M. Won, A. Alimi, and F. Karray. Multi-objective feature selection with NSGA II. In *8th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA 2007)*, volume 4431 of *Lecture Notes in Computer Science*, pages 240–247. Springer, 2007.

J. Hammersley and P. Clifford. Markov fields of finite graphs and lattice. Technical report, University of California-Berkeley, 1971.

J. Handl and J. Knowles. Feature subset selection in unsupervised learning via multiobjective optimization. *International Journal of Computational Intelligence Research*, 2 (3):217–238, 2006.

N. Hansen. The CMA evolution strategy: A comparing review. In Lozano et al. [2006], pages 75–102.

D. Hanzelka. The use of hybrid genetic algorithms in Bayesian network structure learning from data. *Journal of Applied Mathematics*, 1(2):387–396, 2008.

G. Harik, E. Cantú-Paz, D. Goldberg, and B. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999a.

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

G. R. Harik, F. G. Lobo, and K. Sastry. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In Pelikan et al. [2006a], chapter 3, pages 39–61.

W. Hart, N. Krasnogor, and J. Smith. Memetic evolutionary algorithms. *Recent Advances in Memetic Algorithms*, 166:3–27, 2005.

J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

Y. Hasegawa and H. Iba. A Bayesian network approach to program generation. *IEEE Transactions on Evolutionary Computation*, 12(6):750–764, 2008.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, 2nd edition, 2009.

M. Hauschild and M. Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

M. Hauschild, M. Pelikan, C. F. Lima, and K. Sastry. Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 523–530. ACM, 2007.

D. Heckerman. A tutorial on learning with Bayesian networks. In *NATO Advanced Study Institute on Learning in Graphical Models*, pages 301–354. Kluwer Academic Publishers, 1998.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2001.

S. Helbig and D. Pateva. On several concepts for $\epsilon$-efficiency. *OR Spectrum*, 16:179–186, 1994.

M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Second Annual Conference on Uncertainty in Artificial Intelligence (UAI '86)*, volume 2, pages 149–163. Elsevier, 1986.

T. Hesterberg, N. Choi, L. Meier, and C. Fraley. Least angle and $\ell_1$ penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008.

C. Hirsch, P. Shukla, and H. Schmeck. Variable preference modeling using multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2011.

A. E. Hoerl and R. W. Kennard. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

J. Holland. *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.

Y. Hong, G. Zhu, S. Kwong, and Q. Ren. Estimation of distribution algorithms making use of both high quality and low quality individuals. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '09)*, pages 1806–1813. IEEE Computer Society, 2009.

R. Höns. *Estimation of Distribution Algorithms and Minimum Relative Entropy*. PhD thesis, University of Bonn, Bonn, Germany, 2005.

J. Horn, N. Nafpliotis, and D. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *First IEEE Conference on Evolutionary Computation (ICEC '94), IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, 1994.

H.-P. Hsu, V. Mehra, and P. Grassberger. Structure optimization in an off-lattice protein model. *Physical Review E*, 68(3):037703, 2003.

W. H. Hsu, H. Guo, B. B. Perry, and J. A. Stilson. A permutation genetic algorithm for variable ordering in learning Bayesian networks from data. In *Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 383–390. Morgan Kaufmann Publishers Inc., 2002.

B. Huang, B. Buckley, and T.-M. Kechadi. Multi-objective feature selection by using NSGA-II for customer churn prediction in telecommunications. *Expert Systems with Applications*, 37(5):3638–3646, 2010.

S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

E. Hughes. Evolutionary multi-objective ranking with uncertainty and noise. In *Evolutionary Multi-Criterion Optimization (EMO'01)*, volume 1993 of *Lecture Notes in Computer Science*, pages 329–343. Springer, 2001.

E. Hughes. Evolutionary many-objective optimisation: Many once or one many? In *IEEE Congress on Evolutionary Computation (CEC '05)*, volume 1, pages 222–227, 2005.

I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence*, 123(1-2):157–184, 2000.

I. Inza, P. Larrañaga, and B. Sierra. Feature subset selection by Bayesian networks: A comparison with genetic and sequential algorithms. *International Journal of Approximate Reasoning*, 27(2):143–164, 2001a.

I. Inza, M. Merino, P. Larrañaga, J. Quiroga, B. Sierra, and M. Girala. Feature subset selection by genetic algorithms and estimation of distribution algorithms: A case study in the survival of cirrhotic patients treated with TIPS. *Artificial Intelligence in Medicine*, 23(2):187–205, 2001b.

H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *IEEE Congress on Evolutionary Computation (CEC '08) – IEEE World Congress on Computational Intelligence (WCCI '08)*, pages 2419–2426, 2008.

F. Jensen and S. Anderson. Approximations in Bayesian belief universe for knowledge based systems. In *Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI'90)*, pages 162–169. Elsevier, 1990.

F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*, Springer, 2nd edition, 2007.

H. Jia, D. Liu, J. Chen, and J. Guan. Learning Markov equivalence classes of Bayesian network with immune genetic algorithm. In *Third IEEE Conference on Industrial Electronics and Applications (ICIEA '08)*, pages 197–202, 2008.

H.-Y. Jia, D.-Y. Liu, and P. Yu. Learning dynamic Bayesian network with immune evolutionary algorithm. In *Fourth International Conference on Machine Learning and Cybernetics*, volume 5, pages 2934–2938, 2005.

S. Jiang, A. Ziver, J. Carter, C. Pain, A. Goddard, S. Franklin, and H. Phillips. Estimation of distribution algorithms for nuclear reactor fuel management optimisation. *Annals of Nuclear Energy*, 33(11–12):1039–1057, 2006.

Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – A survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

R. Kabli, F. Herrmann, and J. McCall. A chain-model genetic algorithm for Bayesian network structure learning. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 1264–1271. ACM, 2007.

R. Kabli, J. McCall, F. Herrmann, and E. Ong. Evolved Bayesian networks as a versatile alternative to Partin tables for prostate cancer management. In *10th Annual Conference on Genetic and Eevolutionary Computation (GECCO '08)*, pages 1547–1554. ACM, 2008.

H. Kaji, K. Ikeda, and H. Kita. Uncertainty of constraint function in evolutionary multi-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC'09)*, pages 1621–1628, 2009.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

M. Kalisch and P. Bühlmann. Robustification of the PC-algorithm for directed acyclic graphs. *Journal of Computational and Graphical Statistics*, 17(4):773–789, 2008.

H. Karshenas, A. Nikanjam, B. H. Helmi, and A. T. Rahmani. Combinatorial effects of local structures and scoring metrics in Bayesian optimization algorithm. In *First ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09)*, pages 263–270. ACM, 2009.

H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Regularized model learning in estimation of distribution algorithms for continuous optimization problems. Technical Report UPM-FI/DIA/2011-1, Computational Intelligence Group, School of Computer Science, Technical University of Madrid, Madrid, Spain, 2011a.

H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Multi-objective optimization with joint probabilistic modeling of objectives and variables. In *Evolutionary Multi-Criterion Optimization (EMO'11)*, volume 6576 of *Lecture Notes in Computer Science*, pages 298–312. Springer, 2011b.

H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Continuous estimation of distribution algorithms based on factorized Gaussian Markov networks. In Shakya and Santana [2012], pages 157–173.

H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Multi-objective estimation of distribution algorithm based on joint modeling of objectives and variables. *IEEE Transactions on Evolutionary Computation*, Submitted, 2012b.

H. Karshenas, C. Bielza, and P. Larrañaga. Interval-based ranking in noisy evolutionary multi-objective optimization. *Computational Optimization and Applications*, Submitted, 2013a.

H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga. Regularized continuous estimation of distribution algorithms. *Applied Soft Computing*, 13(5):2412–2432, 2013b.

Y. Katsumata and T. Terano. Bayesian optimization algorithm for multi-objective solutions: Application to electric equipment configuration problems in a power plant. In *IEEE Congress on Evolutionary Computation (CEC '03)*, volume 2, pages 1101–1107, 2003.

S. Kern, S. Müller, N. Hansen, D. Büche, J. Očenášek, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms – A comparative review. *Natural Computing*, 3:77–112, 2004.

N. Khan, D. E. Goldberg, and M. Pelikan. Multiple-objective Bayesian optimization algorithm. In *Conference on Genetic and Evolutionary Computation (GECCO '02)*, page 684. Morgan Kaufmann, 2002.

K.-J. Kim, J.-O. Yoo, and S.-B. Cho. Robust inference of Bayesian networks using speciated evolution and ensemble. In *Foundations of Intelligent Systems*, volume 3488 of *Lecture Notes in Computer Science*, pages 185–232. Springer, 2005.

Y. Kim, W. N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6(6):531–556, 2002.

U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2(1):7–17, 1992.

J. A. Kline, A. J. Novobilski, C. Kabrhel, P. B. Richman, and D. M. Courtney. Derivation and validation of a Bayesian network to predict pretest probability of venous thromboembolism. *Annals of Emergency Medicine*, 45(3):282–290, 2005.

K. Koh, S.-J. Kim, and S. P. Boyd. A method for large-scale $\ell_1$-regularized logistic regression. In *22nd AAAI Conference on Artificial Intelligence*, pages 565–571. AAAI Press, 2007.

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

J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.

J. R. Koza, D. Andre, F. H. Bennett, and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999.

J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, 2003.

N. Kramer, J. Schafer, and A.-L. Boulesteix. Regularized estimation of large-scale gene association networks using graphical Gaussian models. *BMC Bioinformatics*, 10(1): 384, 2009.

W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

P. Lanzi, L. Nichetti, K. Sastry, D. Voltini, and D. Goldberg. Real-coded extended compact genetic algorithm based on mixtures of models. In *Linkage in Evolutionary Computation*, volume 157 of *Studies in Computational Intelligence*, pages 335–358. Springer, 2008.

P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Norwell, MA, USA, 2001.

P. Larrañaga and S. Moral. Probabilistic graphical models in artificial intelligence. *Applied Soft Computing*, 11(2):1511–1528, 2011.

P. Larrañaga, C. Kuijpers, R. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 26(4):487–493, 1996a.

P. Larrañaga, R. Murga, M. Poza, and C. Kuijpers. Structure learning of Bayesian networks by hybrid genetic algorithms. In *AI and Statistics V*, volume 112 of *Lecture Notes in Statistics*, pages 165–174. Spriger-Verlag, 1996b.

P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996c.

P. Larrañaga, C. M. H. Kuijpers, M. Poza, and R. H. Murga. Decomposing Bayesian networks: Triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, 7:19–34, 1997.

P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAAIK-IK-4/99, Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.

P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In *Workshop on Optimization by Building and Using Probabilistic Models (OBUPM), Conference on Genetic and Evolutionary Computation (GECCO '00)*, pages 201–204. Morgan Kaufmann, 2000a.

P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Combinatorial optimization by learning and simulation of Bayesian networks. In *16th Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pages 343–352. Morgan Kaufmann Publishers Inc., 2000b.

P. Larrañaga, J. A. Lozano, J. M. Peña, and I. Inza. Editorial of the special issue on probabilistic graphical models in classification. *Machine Learning*, 59:211–212, 2005.

P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on probabilistic graphical models in evolutionary computation. *Journal of Heuristics*, 18(5):795–819, 2012.

P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Information Sciences*, 233 (0):109–125, 2013.

M. Laumanns and J. Očenášek. Bayesian optimization algorithms for multi-objective optimization. In *7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, volume 2439 of *Lecture Notes in Computer Science*, pages 298–307. Springer, 2002.

S. L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, 1992.

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

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B. Methodological*, 50(2):157–224, 1988.

S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17(1):31–57, 1989.

L. Le Cam and G. Lo Yang. *Asymptotics in Statistics: Some Basic Concepts*, Springer, 2nd edition, 2000.

O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004.

J. Lee, W. Chung, and E. Kim. Structure learning of Bayesian networks using dual genetic algorithm. *IEICE Transactions on Information and Systems*, E91-D(1):32–43, 2008.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using $\ell_1$-regularization. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 817–824. MIT Press, 2007.

E. Levina, A. Rothman, and J. Zhu. Sparse estimation of large covariance matrices via a nested LASSO penalty. *Annals of Applied Statistics*, 2(1):245–263, 2008.

B. Li, R. T. Zhong, X. J. Wang, and Z. Q. Zhuang. Continuous optimization based-on boosting Gaussian mixture model. In *18th International Conference on Pattern Recognition (ICPR '06)*, volume 1, pages 1192–1195, 2006.

F. Li and Y. Yang. Using modified Lasso regression to learn large undirected graphs in a probabilistic framework. In *20th National Conference on Artificial intelligence (AAAI'05)*, pages 801–806. AAAI Press, 2005.

H. Li and J. Gui. Gradient directed regularization for sparse Gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7(2):302–317, 2006.

H. Li, Q. Zhang, E. Tsang, and J. A. Ford. Hybrid estimation of distribution algorithm for multiobjective knapsack problem. In *Fourth European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP'04)*, volume 3004 of *Lecture Notes in Computer Science*, pages 145–154. Springer, 2004.

M. Li, D. Goldberg, K. Sastry, and T.-L. Yu. Real-coded ECGA for solving decomposable real-valued optimization problems. In *Linkage in Evolutionary Computation*, volume 157 of *Studies in Computational Intelligence*, pages 61–86. Springer, 2008.

C. Lima, M. Pelikan, D. Goldberg, F. Lobo, K. Sastry, and M. Hauschild. Influence of selection and replacement strategies on linkage learning in BOA. In *CEC 2007, IEEE Congress on Evolutionary Computation*, pages 1083–1090, 2007.

C. Lima, M. Pelikan, F. Lobo, and D. Goldberg. Loopy substructural local search for the Bayesian optimization algorithm. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, volume 5752 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2009.

H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, Norwell, MA, USA, 1998.

H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005.

J. Liu, B. Li, and T. Dillon. An improved naive Bayesian classifier technique coupled with a novel input solution method. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 31(2):249–256, 2001.

A. López Jaimes, C. A. Coello Coello, and D. Chakraborty. Objective reduction using a feature selection technique. In *10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 673–680. ACM, 2008.

J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, Springer, Secaucus, NJ, USA, 2006.

N. Luo and F. Qian. Evolutionary algorithm using kernel density estimation model in continuous domain. In *7th Asian Control Conference (ASCC '09)*, pages 1526–1531, 2009.

L. Malagó, M. Matteucci, and G. Valentini. Introducing $\ell_1$-regularized logistic regression in Markov networks based EDAs. In *IEEE Congress on Evolutionary Computation (CEC'11)*, pages 1581–1588. IEEE, 2011.

C. L. Mallows. Some comments on $C_P$. *Technometrics*, 15(4):661–675, 1973.

B. M. Marlin and K. P. Murphy. Sparse Gaussian graphical models with unknown block structure. In *26th Annual International Conference on Machine Learning (ICML '09)*, pages 705–712. ACM, 2009.

L. Martí, J. Garcia, A. Berlanga, C. A. Coello Coello, and J. M. Molina. On current model-building methods for multi-objective estimation of distribution algorithms: Shortcommings and directions for improvement. Technical Report GIAA2010E001, Department of Informatics, Universidad Carlos III de Madrid, Madrid, Spain, 2010.

L. Martí, J. García, A. Berlanga, C. A. Coello Coello, and J. M. Molina. MB-GNG: Addressing drawbacks in multi-objective optimization estimation of distribution algorithms. *Operations Research Letters*, 39(2):150–154, 2011.

L. Martí, J. García, A. Berlanga, and J. Molina. Multi-objective optimization with an adaptive resonance theory-based estimation of distribution algorithm. *Annals of Mathematics and Artificial Intelligence*, 2012.

M. Mascherini and F. Stefanini. M-GA: A genetic algorithm to search for the best conditional Gaussian Bayesian network. In *International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, volume 2, pages 61–67. IEEE Computer Society, 2005.

I. Matzkevich and B. Abramson. The topological fusion of Bayes nets. In *8th Annual Conference on Uncertainty in Artificial Intelligence (UAI '92)*, pages 191–198. Morgan Kaufmann Publishers Inc., 1992.

R. McKay, N. Hoai, P. Whigham, Y. Shan, and M. O'Neill. Grammar-based genetic programming: A survey. *Genetic Programming and Evolvable Machines*, 11:365–396, 2010.

J. Mehnen, H. Trautmann, and A. Tiwari. Introducing user preference using desirability functions in multi-objective evolutionary optimisation of noisy processes. In *IEEE Congress on Evolutionary Computation (CEC'07)*, pages 2687–2694, 2007.

L. Meier and P. Bühlmann. Smoothing $\ell_1$-penalized estimators for high-dimensional time-course data. *Electronic Journal of Statistics*, 1:597–615, 2007.

N. Meinshausen. Relaxed Lasso. *Computational Statistics & Data Analysis*, 52(1):374–393, 2007.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

A. Mendiburu, R. Santana, and J. A. Lozano. Introducing belief propagation in estimation of distribution algorithms: A parallel framework. Technical Report EHU-KAT-IK-11-07, Intelligent Systems Group, University of the Basque Country, 2007.

O. J. Mengshoel. *Efficient Bayesian Network Inference: Genetic Algorithms, Stochastic Local Search, and Abstraction*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1999.

R. S. Michalski. Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning*, 38:9–40, 2000.

M. Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49(1):8–30, 1961.

T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary computation based on Bayesian classifiers. *International Journal of Applied Mathematics and Computer Science*, 14(3):335–350, 2004.

T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary Bayesian classifier-based optimization in continuous domains. In *6th International Conference on Simulated Evolution and Learning (SEAL '06)*, volume 4247 of *Lecture Notes in Computer Science*, pages 529–536. Springer, 2006.

S. Moral, R. Rumí, and A. Salmerón. Mixtures of truncated exponentials in hybrid Bayesian networks. In *6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU '01)*, volume 2143 of *Lecture Notes in Computer Science*, pages 156–167. Springer, 2001.

M. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In *7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 2, pages 666–670. IEEE Computer Society, 2003.

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

H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In *Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV)*, volume 1141 of *Lecture Notes in Computer Science*, pages 178–187. Springer, 1996.

H. Mühlenbein, T. Mahnig, and A. Ochoa Rodríguez. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247, 1999.

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

J. Muruzábal and C. Cotta. A primer on the evolution of equivalence classes of Bayesian-network structures. In *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 612–621. Springer, 2004.

J. W. Myers, K. B. Laskey, and K. A. DeJong. Learning Bayesian networks from incomplete data using evolutionary algorithms. In *15th Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 476–485. Morgan Kaufmann Publishers, 1999.

R. E. Neapolitan. *Learning Bayesian Networks*, Pearson Prentice Hall, Upper Saddle River, NJ, 2004.

A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *Siam Review*, 40(3):636–666, 1998.

D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8:159–173, 1998.

J. Očenášek and J. Schwarz. Estimation distribution algorithm for mixed continuous-discrete optimization problems. In *2nd Euro-International Symposium on Computational Intelligence*, pages 227–232. IOS Press, 2002.

J. Očenášek, S. Kern, N. Hansen, and P. Koumoutsakos. A mixed Bayesian optimization algorithm with variance adaptation. In *Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 352–361. Springer, 2004.

A. Ochoa. Opportunities for expensive optimization with estimation of distribution algorithms. In *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation, Learning, and Optimization*, pages 193–218. Springer, 2010.

G. Ochoa, E. Lutton, and E. Burke. The cooperative royal road: Avoiding hitchhiking. In *8th International Conference on Artificial Evolution*, volume 4926 of *Lecture Notes in Computer Science*, pages 184–195. Springer, 2008.

T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multi-objective optimization. In *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 792–802. Springer, 2004a.

T. Okabe, Y. Jin, B. Sendoff, and M. Olhofer. Voronoi-based estimation of distribution algorithm for multi-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC'04)*, volume 2, pages 1594–1601, 2004b.

L. Oliveira, M. Morita, and R. Sabourin. Feature selection for ensembles using the multi-objective optimization approach. In *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*, pages 49–74. Springer, 2006.

L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In *16th International Conference on Pattern Recognition (ICPR'02)*, volume 1, pages 568–571. IEEE Computer Society, 2002.

V. Pareto. The new theories of economics. *The Journal of Political Economy*, 5(4): 485–502, 1897.

J. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research*, 21(1):101–133, 2004.

T. Park and K. R. Ryu. Accumulative sampling for noisy evolutionary multi-objective optimization. In *13th Annual Conference on Genetic and Evolutionary Computation (GECCO'11)*, pages 793–800. ACM, 2011.

M. J. Pazzani. Searching for dependencies in Bayesian classifiers. In *Learning From Data: Artificial Intelligence and Statistics V*, Lecture Notes in Statistics, pages 239–248. Springer-Verlag, 1996.

J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *7th Conference of the Cognitive Science Society*, pages 329–334, 1985.

J. Pearl. Distributed revision of composite beliefs. *Artificial Intelligence*, 33(2):173–215, 1987.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, Springer, 1st edition, 2005.

M. Pelikan and D. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In *Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

M. Pelikan and A. Hartmann. Searching for ground states of Ising spin glasses with hierarchical BOA and cluster exact approximation. In Pelikan et al. [2006a], pages 333–349.

M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing-Engineering Design and Manufacturing*, pages 521–535, 1999.

M. Pelikan and K. Sastry. Fitness inheritance in the Bayesian optimization algorithm. In *Conference on Genetic and Evolutionary Computation (GECCO '04)*, volume 3103 of *Lecture Notes in Computer Science*, pages 48–59. Springer, 2004.

M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In *Conference on Genetic and Evolutionary Computation (GECCO '99)*, volume 1, pages 525–532. Morgan Kaufmann Publishers, 1999.

M. Pelikan, E. Cantú-Paz, and D. E. Goldberg. Bayesian optimization algorithm, population sizing, and time to convergence. In *Conference on Genetic and Evolutionary Computation (GECCO '00)*, pages 275–282. Morgan Kaufmann, 2000.

M. Pelikan, D. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

M. Pelikan, D. Goldberg, and S. Tsutsui. Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert. *Information Sciences*, 156 (3-4):147–171, 2003.

M. Pelikan, K. Sastry, and D. E. Goldberg. Multiobjective hBOA, clustering, and scalability. In *Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 663–670. ACM, 2005.

M. Pelikan, K. Sastry, and E. Cantú-Paz, editors. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Springer, Secaucus, NJ, USA, 2006a.

M. Pelikan, K. Sastry, and D. Goldberg. Multiobjective estimation of distribution algorithms. In Pelikan et al. [2006a], pages 223–248.

M. Pelikan, K. Sastry, and D. Goldberg. Sporadic model building for efficiency enhancement of the hierarchical BOA. *Genetic Programming and Evolvable Machines*, 9(1): 53–84, 2008.

J. M. Peña, J. A. Lozano, and P. Larrañaga. Unsupervised learning of Bayesian networks via estimation of distribution algorithms: An application to gene expression data clustering. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(1):63–82, 2004.

J. M. Peña, J. A. Lozano, and P. Larrañaga. Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. *Evolutionary Computation*, 13(1):43–66, 2005.

M. A. Potter, K. A. D. Jong, and J. J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In *Sixth International Conference on Genetic Algorithms (ICGA95)*, pages 366–372. Morgan Kaufmann Publishers, 1995.

P. Pošík. Preventing premature convergence in a simple EDA via global step size setting. In *10th International Conference on Parallel Problem Solving from Nature (PPSN X)*, volume 5199 of *Lecture Notes in Computer Science*, pages 549–558. Springer, 2008.

P. Pošík. Stochastic local search techniques with unimodal continuous distributions: A survey. In *EvoWorkshops on Applications of Evolutionary Computing (EvoWorkshops '09)*, pages 685–694. Springer-Verlag, 2009.

P. Pošík. BBOB-benchmarking a simple estimation of distribution algorithm with Cauchy distribution. In *11th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '09)*, pages 2309–2314. ACM, 2009.

P. V. W. Radtke, T. Wong, and R. Sabourin. Solution over-fit control in evolutionary multiobjective optimization of pattern classification systems. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(6):1107–1127, 2009.

P. Ravikumar, G. Raskutti, M. Wainwright, and B. Yu. Model selection in Gaussian graphical models: High-dimensional consistency of $\ell_1$-regularized MLE. In *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, pages 1329–1336, 2009.

P. Ravikumar, M. J. Wainwright, and J. D. Lafferty. High-dimensional Ising model selection using $\ell_1$-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing $\ell_1$-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

I. Rechenberg. *Evolutionsstrategie-Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. PhD thesis, Reprinted by Fromman-Holzboog, 1973.

B. Reiz, L. Csato, and D. Dumitrescu. Prüfer number encoding for genetic Bayesian network structure learning algorithm. In *10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '08)*, pages 239–242. IEEE Computer Society, 2008.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

R. Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V*, volume 622 of *Lecture Notes in Mathematics*, pages 28–43. Springer, 1977.

V. Robles, P. Larrañaga, J. M. Peña, M. Pérez, E. Menasalvas, and V. Herves. Learning semi–naïve Bayes structures by estimation of distribution algorithms. In *Progress in Artificial Intelligence*, volume 2902 of *Lecture Notes in Computer Science*, pages 244–258. Springer, 2003.

V. Robles, P. Larrañaga, J. M. Peña, E. Menasalvas, M. S. Pérez, V. Herves, and A. Wasilewska. Bayesian network multi-classifiers for protein secondary structure prediction. *Artificial Intelligence in Medicine*, 31(2):117–136, 2004.

J. Rodríguez and J. Lozano. Multi-objective learning of multi-dimensional Bayesian classifiers. In *8th International Conference on Hybrid Intelligent Systems (HIS'08)*, pages 501–506, 2008.

C. Rojas-Guzmán and M. A. Kramer. An evolutionary computing approach to probabilistic reasoning on Bayesian networks. *Evolutionary Computation*, 4:57–85, 1996.

T. Romero and P. Larrañaga. Triangulation of Bayesian networks with recursive estimation of distribution algorithms. *International Journal of Approximate Reasoning*, 50 (3):472–484, 2009.

T. Romero, P. Larrañaga, and B. Sierra. 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, 2004.

B. Ross and E. Zuviria. Evolving dynamic Bayesian networks with multi-objective genetic algorithms. *Applied Intelligence*, 26:13–23, 2007.

H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*, Chapman & Hall, London, 2005.

R. Sagarna and J. A. Lozano. Scatter search in software testing, comparison and collaboration with estimation of distribution algorithms. *European Journal of Operational Research*, 169(2):392–412, 2006.

M. Sahami. Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

R. Salinas-Gutiérrez, A. Hernández-Aguirre, and E. Villa-Diharce. Using copulas in estimation of distribution algorithms. In *Advances in Artificial Intelligence (MICAI '09)*, volume 5845 of *Lecture Notes in Computer Science*, pages 658–668. Springer, 2009.

R. P. Sałustowicz and J. Schmidhuber. Probabilistic incremental program evolution: Stochastic search through program space. In *9th European Conference on Machine Learning (ECML '97)*, volume 1224 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1997.

R. Santana. A Markov network based factorized distribution algorithm for optimization. In *14th European Conference on Machine Learning (ECML '03)*, volume 2837 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2003.

R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13:67–97, 2005.

R. Santana. Estimation of distribution algorithms: From available implementations to potential developments. In *13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '11)*, pages 679–686. ACM, 2011.

R. Santana, P. Larrañaga, and J. A. Lozano. Side chain placement using estimation of distribution algorithms. *Artificial Intelligence in Medicine*, 39(1):49–63, 2007.

R. Santana, P. Larrañaga, and J. A. Lozano. Protein folding in simplified models with estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 12(4):418–438, 2008.

R. Santana, C. Bielza, J. A. Lozano, and P. Larrañaga. Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pages 445–452. ACM, 2009a.

R. Santana, P. Larrañaga, and J. A. Lozano. Research topics in discrete estimation of distribution algorithms based on factorizations. *Memetic Computing*, 1(1):35–54, 2009b.

R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armañanzas, and S. Shakya. Mateda-2.0: Estimation of distribution algorithms in MATLAB. *Journal of Statistical Software*, 35(7):1–30, 2010a.

R. Santana, P. Larrañaga, and J. A. Lozano. Learning factorizations in estimation of distribution algorithms using affinity propagation. *Evolutionary Computation*, 18(4): 515–546, 2010b.

R. Santana, A. Mendiburu, N. Zaitlen, E. Eskin, and J. A. Lozano. Multi-marker tagging single nucleotide polymorphism selection using estimation of distribution algorithms. *Artificial Intelligence in Medicine*, 50(3):193–201, 2010c.

R. Santana, H. Karshenas, C. Bielza, and P. Larrañaga. Regularized k-order Markov models in EDAs. In *13th Annual Conference on Genetic and Evolutionary Computation (GECCO'11)*, pages 593–600. ACM, 2011.

K. Sastry and D. E. Goldberg. Probabilistic model building and competent genetic programming. In *Genetic Programming Theory and Practice*, chapter 13, pages 205–220. Kluwer Academic Publishers, 2003.

K. Sastry, M. Pelikan, and D. Goldberg. Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *IEEE Congress on Evolutionary Computation (CEC '04)*, volume 1, pages 720–727, 2004.

K. Sastry, D. Goldberg, and M. Pelikan. Limits of scalability of multiobjective estimation of distribution algorithms. In *IEEE Congress on Evolutionary Computation (CEC'05)*, volume 3, pages 2217–2224, 2005.

J. Schäfer and K. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005a.

J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1):online, DOI: 10.2202/1544–6115.1175, 2005b.

M. Schmidt. Least squares optimization with L1-norm regularization. Technical report, University of British Columbia, 2005.

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using L1-regularization paths. In *22nd National Conference on Artificial Intelligence (AAAI '07)*, volume 2, pages 1278–1283. AAAI Press, 2007.

J. R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics, 1995.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

J. Schwarz and J. Očenášek. Multiobjective Bayesian optimization algorithm for combinatorial problems: Theory and practice. *Neural Network World*, 11(5):423–442, 2001.

H.-P. Schwefel. *Evolution and Optimum Seeking*, John Wiley & Sons, New York, NY, USA, 1995.

M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*, volume 1498 of *Lecture Notes in Computer Science*, pages 418–427. Springer, 1998.

R. Shachter and M. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI'89)*, pages 311–318. Elsevier Science, 1989.

S. Shakya. *DEUM: A Framework for an Estimation of Distribution Algorithm Based on Markov Random Fields*. PhD thesis, Robert Gordon University, 2006.

S. Shakya and J. McCall. Optimization by estimation of distribution with DEUM framework based on Markov random fields. *International Journal of Automation and Computing*, 4:262–272, 2007.

S. Shakya and R. Santana, editors. *Markov Networks in Evolutionary Computation*, Springer, Berlin, Heidelberg, 2012.

S. Shakya, R. Santana, and J. A. Lozano. A Markovianity based optimisation algorithm. *Genetic Programming and Evolvable Machines*, 13(2):159–195, 2012.

Y. Shan, R. McKay, D. Essam, and H. Abbass. A survey of probabilistic model building genetic programming. In Pelikan et al. [2006a], pages 121–160.

P. P. Shenoy. A re-definition of mixtures of polynomials for inference in hybrid Bayesian networks. In *11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU '11)*, volume 6717 of *Lecture Notes in Artificial Intelligence*, pages 98–109. Springer, 2011.

P. P. Shenoy and J. C. West. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657, 2011.

S. Shi, P. Suganthan, and K. Deb. Multiclass protein fold recognition using multiobjective evolutionary algorithms. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'04)*, pages 61–66, 2004.

V. A. Shim, K. C. Tan, and C. Y. Cheong. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(5):682–691, 2012a.

V. A. Shim, K. C. Tan, and K. K. Tan. A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC'12)*, pages 1–8, 2012b.

V. A. Shim, K. C. Tan, and J. Y. Chia. Multi-objective optimization with estimation of distribution algorithm in a noisy environment. *Evolutionary Computation*, 21(1): 149–177, 2013.

S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68 (2):399–410, 1994.

B. Sierra and P. Larrañaga. Predicting survival in malignant skin melanoma using Bayesian networks automatically induced by genetic algorithms: An empirical comparison between different approaches. *Artificial Intelligence in Medicine*, 14(1-2):215–230, 1998.

B. Sierra, N. Serrano, P. Larrañaga, E. J. Plasencia, I. Inza, J. J. Jiménez, P. Revuelta, and M. L. Mora. Using Bayesian networks in the construction of a bi-level multi-classifier: A case study using intensive care unit patients data. *Artificial Intelligence in Medicine*, 22(3):233–248, 2001.

G. Soares, F. Guimaraes, C. Maia, J. Vasconcelos, and L. Jaulin. Interval robust multi-objective evolutionary algorithm. In *IEEE Congress on Evolutionary Computation (CEC'09)*, pages 1637–1643, 2009.

H. Soh and M. Kirley. moPGA: Towards a new generation of multi-objective genetic algorithms. In *IEEE Congress on Evolutionary Computation (CEC'06)*, pages 1702–1709, 2006.

T. P. Speed and H. T. Kiiveri. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14(1):138–150, 1986.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*, The MIT Press, 2nd edition, 2001.

N. Spolaôr, A. Lorena, and H. Lee. Multi-objective genetic algorithm evaluation in feature selection. In *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 462–476. Springer, 2011.

N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

N. Sriwachirawat and S. Auwatanamongkol. On approximating K-MPE of Bayesian networks using genetic algorithm. In *IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6, 2006.

K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1):63–100, 2004.

C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Third Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 197–206. University of California Press, 1956.

F. H. Stillinger, T. Head-Gordon, and C. L. Hirshfeld. Toy model for protein folding. *Physical Review E*, 48(2):1469–1477, 1993.

M. Sugiyama, M. Kawanabe, and K.-R. Müller. Trading variance reduction with unbiasedness: The regularized subspace information criterion for robust model selection in kernel regression. *Neural Computation*, 16(5):1077–1104, 2004.

J. Sun, Q. Zhang, and E. Tsang. DE/EDA: A new evolutionary algorithm for global optimization. *Information Sciences*, 169(3-4):249–262, 2005.

J. Sun, Q. Zhang, J. Li, and X. Yao. A hybrid estimation of distribution algorithm for CDMA cellular system design. *International Journal of Computational Intelligence and Applications*, 7(2):187–200, 2008.

A. Syberfeldt, A. Ng, R. I. John, and P. Moore. Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. *European Journal of Operational Research*, 204(3):533–544, 2010.

K. Tan and C. Goh. Handling uncertainties in evolutionary multi-objective optimization. In *Computational Intelligence: Research Frontiers*, volume 5050 of *Lecture Notes in Computer Science*, pages 262–292. Springer, 2008.

H. Tang, V. A. Shim, K. C. Tan, and J. Y. Chia. Restricted Boltzmann machine based algorithm for multi-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC'10)*, pages 1–8, 2010.

J. Teich. Pareto-front exploration with uncertain objectives. In *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2001.

D. Thierens. The linkage tree genetic algorithm. In *Parallel Problem Solving from Nature (PPSN XI)*, volume 6238 of *Lecture Notes in Computer Science*, pages 264–273. Springer, 2011.

D. Thierens and P. A. N. Bosman. Multi-objective mixture-based iterated density estimation evolutionary algorithms. In *Conference on Genetic and Evolutionary Computation (GECCO '01)*, pages 663–670. Morgan Kaufmann, 2001.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B (Methodological)*, 58(1):267–288, 1996.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused Lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

A. Tikhonov and V. Arsenin. *Solutions of ill-posed problems*, Winston, Washington DC, USA, 1977.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006.

S. Tsutsui. Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In *Parallel Problem Solving from Nature (PPSN VII)*, volume 2439 of *Lecture Notes in Computer Science*, pages 224–233. Springer, 2002.

S. Tsutsui, M. Pelikan, and D. E. Goldberg. Evolutionary algorithm using marginal histogram in continuous domain. In *Optimization by Building and Using Probabilistic Models (OBUPM) Workshop – Conference on Genetic and Evolutionary Computation (GECCO '01)*, pages 230–233. 2001.

S. Tsutsui, M. Pelikan, and D. Goldberg. Node histogram vs. edge histogram: A comparison of PMBGAs in permutation domains. Technical Report 2006009, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), Department of Mathematics and Computer Science, University of Missouri–St. Louis, 2006.

A. Tucker, X. Liu, and A. Ogden-Swift. Evolutionary learning of dynamic probabilistic models with large time lags. *International Journal of Intelligent Systems*, 16(5):621–645, 2001.

A. Tucker, X. Liu, and D. Garway-Heath. Spatial operators for evolving dynamic Bayesian networks from spatio-temporal data. In *International Conference on Genetic and Evolutionary Computation (GECCO '03): Part II*, volume 2724 of *Lecture Notes in Computer Science*, pages 2360–2371. Springer, 2003.

D. E. Tyler. A distribution-free M-estimator of multivariate scatter. *Annals of Statistics*, 15(1):234–251, 1987.

T. Ulrich, D. Brockhoff, and E. Zitzler. Pattern identification in Pareto-set approximations. In *10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 737–744. ACM, 2008.

S. I. Valdez-Peña, A. Hernández-Aguirre, and S. Botello-Rionda. Approximating the search distribution to the selection distribution in EDAs. In *11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pages 461–468. ACM, 2009.

S. van Dijk and D. Thierens. On the use of a non-redundant encoding for learning Bayesian networks from data with a GA. In *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 141–150. Springer, 2004.

S. van Dijk, D. Thierens, and L. van der Gaag. Building a GA from design principles for learning Bayesian networks. In *Fifth Annual Conference on Genetic and Evolutionary Computation (GECCO '03): Part I*, volume 2723 of *Lecture Notes in Computer Science*, pages 886–897. Springer, 2003.

R. van Engelen. Approximating Bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):916–920, 1997.

I. Vatolkin, M. Preuß, G. Rudolph, M. Eichhoff, and C. Weihs. Multi-objective evolutionary feature selection for instrument recognition in polyphonic audio mixtures. *Soft Computing*, 16(12):2027–2047, 2012.

S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens. Analyzing the effect of objective correlation on the efficient set of MNK-landscapes. In *5th International Conference on Learning and Intelligent Optimization (LION'5)*, volume 6683 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2011.

D. Vidaurre, C. Bielza, and P. Larrañaga. Learning an $l_1$-regularized Gaussian Bayesian network in the equivalence class space. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 40:1231–1242, 2010.

M. Wagner, A. Auger, and M. Schoenauer. EEDA: A new robust estimation of distribution algorithm. Technical Report 5190, Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt, France, 2004.

M. J. Wainwright, P. Ravikumar, and J. D. Lafferty. High-dimensional graphical model selection using $\ell_1$-regularized logistic regression. In *Advances in Neural Information Processing Systems, 19*, pages 1465–1472. MIT Press, 2006.

H. Wang, K. Yu, X. Wu, and H. Yao. Triangulation of Bayesian networks using an adaptive genetic algorithm. In *Foundations of Intelligent Systems*, volume 4203 of *Lecture Notes in Computer Science*, pages 127–136. Springer, 2006.

L. Wang and C. Fang. An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research*, 39(2):449–460, 2012.

L.-F. Wang and J.-C. Zeng. Estimation of distribution algorithm based on copula theory. In *Exploitation of Linkage Learning in Evolutionary Algorithms*, volume 3 of *Evolutionary Learning and Optimization*, pages 139–162. Springer, 2010.

L.-F. Wang, J.-C. Zeng, and Y. Hong. Estimation of distribution algorithm based on Archimedean copulas. In *First ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09)*, pages 993–996. ACM, 2009.

X. Wang and H. Wang. Evolutionary optimization with Markov random field prior. *IEEE Transactions on Evolutionary Computation*, 8(6):567–579, 2004.

T. Weise, S. Niemczyk, R. Chiong, and M. Wan. A framework for multi-model EDAs with model recombination. In *Applications of Evolutionary Computation*, volume 6624 of *Lecture Notes in Computer Science*, pages 304–313. Springer, 2011.

W. X. Wen. Optimal decomposition of belief networks. In *Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI '90)*, pages 209–224. Elsevier Science Inc., 1991.

D. M. Witten and R. Tibshirani. Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society. Series B. Methodological*, 71(3):615–636, 2009.

M. L. Wong and K. S. Leung. An efficient data mining method for learning Bayesian networks using an evolutionary algorithm-based hybrid approach. *IEEE Transactions on Evolutionary Computation*, 8(4):378–404, 2004.

M. L. Wong, W. Lam, and K. S. Leung. Using evolutionary programming and minimum description length principle for data mining of Bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):174–178, 1999.

M. L. Wong, S. Y. Lee, and K. S. Leung. Data mining of Bayesian networks using cooperative coevolution. *Decision Support Systems*, 38(3):451–472, 2004.

P. Woźniak. Preferences in evolutionary multi-objective optimisation with noisy fitness functions: Hardware in the loop study. In *International Multiconference on Computer Science and Information Technology*, pages 337–346, 2007.

J. Xiao, Y. Yan, and J. Zhang. HPBILc: A histogram-based EDA for continuous optimization. *Applied Mathematics and Computation*, 215(3):973–982, 2009.

K. Yanai and H. Iba. Estimation of distribution programming based on Bayesian network. In *IEEE Congress on Evolutionary Computation (CEC '03)*, volume 3, pages 1618–1625, 2003.

J. Yang, H. Xu, Y. Cai, and P. Jia. Effective structure learning for EDA via L1-regularized Bayesian networks. In *12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, pages 327–334. ACM, 2010.

P. Yates and M. Reimers. RCMAT: A regularized covariance matrix approach to testing gene sets. *BMC Bioinformatics*, 10(1):300, 2009.

R. Yehezkel and B. Lerner. Bayesian network structure learning by recursive autonomy identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009.

B. Yuan and M. Gallagher. On the importance of diversity maintenance in estimation of distribution algorithms. In *Sixth Annual Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 719–726. ACM, 2005.

B. Yuan and M. Gallagher. Convergence analysis of $\mathrm{UMDA_C}$ with finite populations: A case study on flat landscapes. In *11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pages 477–482. ACM, 2009.

M. Yuan. Efficient computation of $\ell_1$ regularized estimates in Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 17(4):809–826, 2008.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

D. Zaharie, D. Lungeanu, and S. Holban. Feature ranking based on weights estimated by multiobjective optimization. In *IADIS European Conference on Data Mining*, pages 124–128, 2007.

Q. Zhang. 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, 2004.

Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

Q. Zhang and H. Mühlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):127–136, 2004.

Q. Zhang, A. Zhou, and Y. Jin. RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008.

Q. Zhang, W. Liu, and H. Li. The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *IEEE Congress on Evolutionary Computation (CEC'09)*, pages 203–208, 2009a.

Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-487, The Shool of Computer Science and Electronic Engineering, University of Essex, UK, 2009b.

Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2010.

X. Zhang, B. Lu, S. Gou, and L. Jiao. Immune multiobjective optimization algorithm for unsupervised feature selection. In *Applications of Evolutionary Computing (EvoWorkshops 2006)*, volume 3907 of *Lecture Notes in Computer Science*, pages 484–494. Springer, 2006.

Y. Zhang and X. Li. Estimation of distribution algorithm for permutation flow shops with total flowtime minimization. *Computers & Industrial Engineering*, 60(4):706–718, 2011.

X. Zhong and W. Li. A decision-tree-based multi-objective estimation of distribution algorithm. In *International Conference on Computational Intelligence and Security (CIS'07)*, pages 114–118. IEEE Computer Society, 2007.

Z. Zhu, Y.-S. Ong, and M. Dash. Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition*, 40(11):3236–3248, 2007.

Z. Zhu, Y.-S. Ong, and J.-L. Kuo. Feature selection using single/multi-objective memetic frameworks. In *Multi-Objective Memetic Algorithms*, volume 171 of *Studies in Computational Intelligence*, pages 111–131. Springer, 2009.

E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer, 2004.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems (EUROGEN '01)*, pages 95–100. International Center for Numerical Methods in Engineering, 2001.

E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

E. Zitzler, J. Knowles, and L. Thiele. Quality assessment of Pareto set approximations. In *Multiobjective Optimization*, volume 5252 of *Lecture Notes in Computer Science*, pages 373–404. Springer, 2008.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B (Methodological)*, 67(2):301–320, 2005.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.