Conditional log-likelihood MDL and Evolutionary MCMC

PROEFSCHRIFT

ter verkrijging van de graad van
doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof. dr. W.H. Gispen,
ingevolge het besluit van het college van promoties
in het openbaar te verdedigen
op maandag 27 november 2006 des middags te 12.45 uur

door

Mădălina Mihaela Drugan
geboren op 5 november 1975, te Galati, Roemenië

Promotor:	prof. dr. ir. Linda C. van der Gaag
Co-promotor:	dr. ir. Dirk Thierens

This work was carried out in the SIKS graduate school.
SIKS dissertation series number 2006-25

for my family

Contents

Chapter 1

Introduction

Many modern technologies generate and collect a large amount of data that cannot be anymore efficiently processed by human beings. Machine learning, Bayesian inference and information theory are three important areas of research, all of them concerned with the processing of data, that share considerable overlap.

Machine learning, which generically and generally denotes any algorithm that *adapts* its output from its inputs, is an increasingly important field in computer science. The purpose of machine learning is to use data to improve a performance element which determines how the algorithm behaves when given certain inputs. The performance element can be a classifier or the difference between the target and a sampled distribution. By receiving feedback on the performance, the algorithms adapt. Depending on the feedback, we have unsupervised and supervised learning. In supervised learning, the algorithms receive inputs and outputs and search for the function that efficiently maps the input to the output. In unsupervised learning, an algorithm receives only the input data and uses a function to extract useful information from the data. Machine learning is overlapping with statistics since both study and analyze the data. Unlike statistics, machine learning is concerned with the computational complexity of its algorithms which should run on realistic computers. Therefore, for intractable NP-hard problems, machine learning proposes tractable approximative inference algorithms. Two popular textbooks about machine learning algorithms are Michell [67] and Alpaydin [2].

Bayesian inference is statistical inference where evidence (or aprior knowledge) is used to update or to newly infer the probability that a hypothesis may be true. Bayesian inference uses a numerical estimate of the degree of belief in a hypothesis before the data has been observed and calculates a numerical estimate of the degree of belief in the hypothesis after data has been observed. Two textbooks about Bayesian Inference are Box and Tiao [11] and Harney [44]. For example, the evidence is some dataset and we test the hypothesis is that there are certain relationships between the variables in the dataset. Or, the evidence consists of some samples generated with a sampling algorithm, and we use Bayesian inference to estimate some function of these samples. Machine learning algorithms are often used with Bayesian inference for testing alternative hypothesis.

Information theory is a field of mathematics concerning the storage and transmission of data. The data, in order to be sent, is compressed using few bits in such a manner that the receiver can recover it, under the assumption that both sender and receiver understand the encoding. Compression is possible in many real-world application, for example, due to patterns, overlapping and noise. Some advantages of using data compression are noise filtering, pattern exploitation and resource saving. An introduction into information theory we find in Shannon [82]. Information theory is to be quite often used within machine learning algorithms; then, an algorithm is encoding in bits the input to

be transformed in the output. For example, we assume that the variables in a given dataset are not independent: the encoding of the relationships between the variables is more economical than just to send the entire dataset to the receiver. Then, we can use a performance measure to search for the best encoding. Such an example is the two parts Minimum Description Length (MDL) function: with one part we encode the regularities in the data from which we subtract the cost of encoding the data to avoid overfitting.

Like the title, this thesis is thematically split in two parts. Here, we tackle two subjects that can be considered subfields of the above three areas of research: machine learning, Bayesian inference and information theory. The first part is dedicated to a score function to discriminate between two classifiers. Many real-life problems, such as medical diagnosis and troubleshooting of technical equipment, can be viewed as a classification problem. There an instance described by a number of features has to be classified in one of several distinct pre-defined classes. For many of these classification problems, instances of every-day problem solving are recorded in a dataset. Such a dataset often contains regularities and includes more features, or attributes, of the problem's instances than are strictly necessary for the classification task at hand. To represent the relationships between the variables from the dataset, we use Bayesian network classifiers - that are graphical models used to represent the relationships between variables. We use supervised machine learning algorithms to search for the simplest Bayesian network classifiers that most accurately represent the data. In this part, we focus on designing the performance measure for evaluating these classifiers. For that, we adapt a specific function from information theory: an MDL score function. We use Bayesian inference to verify if the relationships in the Bayesian network classifier correspond to the regularities in the dataset.

In the second part, we investigate some sampling algorithms. There are various real-world problems where the search space is huge and the distribution over the search space has unknown properties. To estimate some function of interest of such distribution, we need some algorithm that explores the search space and, in the same time, the properties of the distribution (e.g. how many and how close are the important regions). In this part, we focus on how to integrate evolutionary inspired techniques (e.g. population, recombination and selection) into samplers with the Markov property. We use unsupervised machine learning algorithms to adapt the way that we generate samples from the desired distribution. The performance measure is now some distance measure between the given distribution and the sampled distribution by the algorithm. We use Bayesian inference to estimate some function of the samples generated by the algorithm. Since we aim to design sampling algorithms that generate few samples from which we are able to reconstruct the target distribution, we consider these samplers belonging also to the Information Theory field.

However, these two parts are rather unrelated at the level of abstraction which we approach them here. Therefore, except for the introduction, we refer to each part separately.

1.1 The first part: conditional log-likelihood MDL

In the first part of this thesis, we propose and investigate the properties of a conditional log-likelihood Minimum Description Length (MDL) scoring function. We use it as a performance measure for learning from the instances of a dataset simple Bayesian network classifiers that correctly, but without overfitting, predict the class variable given the other variables.

When constructing a classifier from the dataset, the more or less redundant features may bias the classifier resulting in a relatively poor classification performance. Furthermore, a classifier can be subject to overfitting if in the Bayesian network classifier we represent also the less important relationships between the attributes. We evaluate the relationships between the variables of the dataset

using a MDL like score function.

For constructing a classifier, generally a heuristic algorithm is used that searches the space of possible models for classifiers of high quality. For comparing the qualities of alternative classifiers, we use a measure called the Minimum Description Length for Feature Selection (MDL-FS) function, closely related to the well-known Minimum Description Length (MDL) function. The MDL function is quite popular as quality measure for constructing Bayesian networks. Whereas in Bayesian network classifiers the class variable is the most important variable, in Bayesian networks all the variables are considered equally important. Thus, unlike in the Bayesian networks in general, in the Bayesian network classifiers, the relationships between attributes are interpreted in the presence of the class variable. The function in essence weighs the complexity of a model against its ability to capture the observed joint probability distribution.

In contrast, this conditional log-likelihood MDL, the MDL-FS score, is adjusted to encode the structure of Bayesian network classifiers focusing on the class variable rather than on the other attributes from the dataset. However, using the conditional log-likelihood to learn Bayesian network classifiers is generally known to be computationally infeasible. MDL-FS subtracts the encoding of the Bayesian network over all the variables - that is the class variable and the other attributes - from the encoding of the Bayesian network over the other attributes. This method is not only computationally feasible but also exact when all the relationships between variables in the dataset are considered.

We test the proposed MDL score in a practical setting; we show that its performance is superior to the standard MDL score and other methods in constructing simple Bayesian network classifiers. We study the behavior of the MDL-FS function, in antithesis with the behavior of the MDL function, in learning the structure of selective Bayesian network classifiers from data. The first part of this thesis is contained in Chapter 2, 3, and 4. In the following paragraphs we indicate the main contributions this thesis brings in the conditional log-likelihood MDL domain.

Definition of (noisy) redundant and irreduntant attributes. In Section 2.2, we provide a new definition of the concept of redundant and irredundant attributes that is tailored to Bayesian network classifiers since it is related to the types of dependence that can be expressed by a classifier. In our definition, we combat also the noise that is often present in real datasets.

Design the MDL-FS score function. The MDL-FS function differs from the MDL function only in that it encodes the conditional probability distribution over the class variable given the various attributes. Upon using the function as a measure for comparing the qualities of Bayesian network classifiers, this conditional distribution has to be learned from the available data. Unfortunately, learning a conditional distribution is generally acknowledged to be hard, since it does not decompose over the graphical structure of a Bayesian network classifier as does the joint distribution.

Our MDL-FS function in contrast approximates the conditional distribution by means of an auxiliary Bayesian network over the attributes which capture the strongest relationships between the attributes. With the function, both the structure of the Bayesian network classifier over all variables involved and the structure of the auxiliary network are learned using a less demanding generative method. The conditional log-likelihood of the classifier is approximated by the difference between the unconditional log-likelihood of the classifier and the log-likelihood of the auxiliary network. Furthermore, when we use all the information form the dataset to learn the Bayesian network classifier and the auxiliary structure, we show that we perfectly model the conditional log-likelihood. We introduce this function in Section 3.1.

Analyze the feature selection behavior of the MDL-FS function. We show that, unlike the MDL function, the MDL-FS function serves to identify redundancy and in essence is able to eliminate redundant attributes from different types of Bayesian network classifiers. In Section 3.2 and 3.3, we show that the level of redundancy at which attributes are eliminated with the MDL-FS function is

closely related to the complexity of the auxiliary network used.

Performant Bayesian network classifiers constructed with MDL-FS. In Section 3.1, we show that for a fixed set of attributes, the MDL-FS function is equivalent to the MDL function. The MDL-FS function therefore inherits the tendency of the MDL function to result in good Bayesian network classifiers.

In Section 3.4, we propose a variant of the MDL-FS score which is in fact the difference between the MDL score of a Bayesian network classifier and the MDL score of an auxiliary structure that models the strongest relationships between attributes. Using this variant, the MDL-FS score is the interplay between the (approximation of the) conditional log-likelihood term and the complexity of the model that represents the conditional log-likelihood - that is the complexity of the classifier from which we subtract the complexity of the auxiliary structure. However, for the feature selection task this score has a worse behavior than for the previous variant since the complexity of the conditional log-likelihood is larger than the complexity of the classifier alone.

Experiments. To compare the feature-selection behavior of the MDL-FS and MDL functions in a practical setting, we conducted various experiments in Section 3.6 in which we used different learning algorithms for constructing Naive Bayesian classifiers and TAN classifiers from various datasets. Our results indicate that the MDL-FS function indeed is more suited to the task of feature subset selection than the MDL function as it yields classifiers of comparably good or even significantly better performance with fewer attributes. Furthermore, using the second variant of the MDL-FS score we build performant selective TAN classifiers.

1.2 The second part: evolutionary MCMC

The Markov chain Monte Carlo (MCMC) framework is used for sampling from complicated target distributions (e.g. multi-variable distributions with non-linear correlations between dimensions) that cannot be sampled with simpler, distribution specific, methods. However, there are many distributions where a standard MCMC that does not exploit the properties of the distribution, performs poorly. Ideally, an MCMC algorithm proposes individuals directly from the target distribution such that from few samples we could recover the target distribution. Unfortunately, in practice, this is hardly the case (if we would know the target distribution, we would not need an EMCMC to sample from it).

To improve the efficiency of standard MCMC we can extend them with techniques from Evolutionary Computation (e.g. population, recombination, selection). In the Evolutionary MCMC (EMCMC) framework, population-based MCMCs exchange information between the individual states, such that at population level, they are still MCMCs. In this research work, we generate biased distributions using recombination operators that exploit relationships – such as correlations and commonalities – present in the target distribution. In this perspective, the recombination operators adapt the proposal distribution from the current population. In the second part of this thesis, we investigate how to use recombination and selection in MCMC to obtain efficient samplers.

Like mutation operators, recombination operators have no knowledge about how fit or how likely the individual states are. Therefore, as in standard MCMC sampling, we need a mechanism – for instance, an acceptance rule – to evaluate the suitability of the newly generated individuals.

In the following paragraphs we specify the main contributions this thesis brings to the (E)MCMC domain.

Designing recombination operators for EMCMCs. On one hand we want to “economically” generate new individual states, and on the other hand we want to generate “good” individuals such that the EMCMC algorithm is performing well.

In the real-coded space, we use geometrical transformations - more specifically linear transformations like translation and rotation - of the parents into children to formally express our recombinations. In Section 6.2, we show that our framework greatly extends the use of recombination operators in EMCMC algorithms; we propose new recombination schemes that exploit new types of commonalities and relationships - for instance, correlations - between certain dimensions in the target distribution. In our framework, the recombination operators only require a linear computational effort with the number of dimensions and parents.

We theoretically and experimentally study recombination operators as rotation, translation, and scaling for simplex geometrical figures; we call these operators simplex recombinations. A simplex represents the simplest non-degenerate geometrical shape in a ℓ dimensional space; it consists of $\ell+1$ points and all their interconnecting line segments and polygonal faces.

In Section 6.1, we propose and investigate the properties of some discrete spaces recombination operators.

Combining recombination and mutation operators. Recombination operators are mostly reducible, which means that we cannot generate any state from any other state in several steps. Such a proposal distribution cannot be used with MCMCs. We show how to combine recombination with mutation to obtain irreducible proposal distributions that can be used with MCMC. In Section 6.3 we investigate the properties of such combinations of proposal distributions.

Designing acceptance rules for EMCMCs. We investigate the properties of recombinative and non-recombinative EMCMCs where one candidate state competes against one of its parents with the standard MH acceptance rule. We analytically show that a population of independent MCMCs can outperform the single chain MCMC on a toy problem. Also, when we add recombination operators, we show under what conditions the resulting algorithm can be used as a MCMC to sample from the target distribution. This recombinative EMCMC is the most efficient algorithm we have tested analytically and experimentally.

In general, to sample from the desired distribution the individuals that interact through recombination also need to interact in the acceptance rule. However, such an acceptance rule has a negative effect over the performance of an EMCMC as compared with the standard acceptance rule where only one candidate individual competes against one parent.

Furthermore, the acceptance rules derived from the EC's elitist selection rules, result in elitist acceptance rules that accept with a high probability the two most fit parents and children which disproportionately peaks the target distribution. We control the height of these peaks, and thus the mixing behavior, with a temperature scheduler attached to the individuals from the population such that the higher the fitness the lower the attached temperature. As a result, the more fit individuals remain longer in the population whereas with the less fit ones are replaced in order to further explore the search space. We investigate these acceptance rules in Chapter 7.

Examples. In Chapter 8, we compare the performance of (recombinative) EMCMCs with the standard MCMCs. For the real-coded space, we perform experiments on two abstract mathematical functions that model key properties of practical problems. We compare recombinative and non-recombinative EMCMC and SA algorithms on a *binary quadratic programming problem* (BQP). The results obtained show that recombination improves the mixing of EMCMCs.

1.3 Discussion and future work

We conclude this thesis by gathering our final remarks in Chapter 10. We explore some possible future developments for the two subjects we have discussed.

Part I

Conditional log-likelihood MDL

Chapter 2

Preliminaries

Many real-life problems, such as medical diagnosis and troubleshooting of technical equipment, can be viewed as a classification problem, where an instance described by a number of features has to be classified in one of several distinct pre-defined classes. For many of these classification problems, instances of every-day problem solving are recorded in a dataset. Such a dataset often includes more features, or attributes, of the problem's instances than are strictly necessary for the classification task at hand. When constructing a classifier from the dataset, these more or less redundant features may bias the classifier and as a consequence may result in a relatively poor classification accuracy. By constructing the classifier over just a subset of the features, a less complex classifier is yielded that tends to have a better generalisation performance [59]. Finding a minimum subset of features such that the selective classifier constructed over this subset is optimal for a given performance measure, is known as the *feature subset selection* problem. The feature subset selection problem unfortunately is NP-hard in general [9, 90]. For constructing selective classifiers therefore, generally a heuristic algorithm is employed. Such an algorithm searches the space of possible models for classifiers of high quality. For comparing the qualities of alternative classifiers, typically a quality measure is employed.

In the first part of this thesis, we address the problem of feature subset selection in view of Bayesian network classifiers. We begin by providing a new definition of the concept of redundancy of attributes, where the redundancy is viewed within some allowed amount of noise in the data under study. Several researchers have addressed the concept of redundancy to provide for studying the performance of various algorithms for feature subset selection in general [90, 13, 41, 47, 50, 51]. Our new definition of the concept allows us to study feature selection for different types of Bayesian network classifier more specifically. With our definition we distinguish between different *levels of redundancy* for an attribute by the cardinality of the (sub)sets of attributes given which it is not useful for the classification task. We will argue that these levels of redundancy provide for relating the problem of feature subset selection to the types of dependence that can be expressed by a Bayesian network classifier. By allowing noise for the various levels, our concept of redundancy provides for studying feature selection in a practical setting.

For constructing a selective classifier, generally a heuristic algorithm is used that searches the space of possible models for classifiers of high quality. Because of its simplicity, its intuitive theoretical foundation and its associated ease of computation, the MDL function and its variants [10, 38, 42, 75] have become quite popular as quality measures for constructing Bayesian networks from data, and in fact for constructing Bayesian network classifiers [15, 30]. The function in essence weighs the complexity of a model against its ability to capture the observed probability distribution. While

the MDL function and its variants are accepted as suitable functions for comparing the qualities of alternative Bayesian networks, they are not without criticism when constructing Bayesian network classifiers. The criticism focuses on the observation that the functions capture a joint probability distribution over the variables of a classifier, while it is the conditional distribution over the class variable given the attributes that is of interest for the classification task [30, 55].

For comparing the qualities of alternative classifiers, we propose the *Minimum Description Length for Feature Selection* (MDL-FS) function [21, 20]. The MDL-FS function is closely related to the well-known Minimum Description Length (MDL) function. It differs from the MDL function only in that it encodes the conditional probability distribution over the class variable given the various attributes. Upon using the function as a measure for comparing the qualities of Bayesian network classifiers therefore, this conditional distribution has to be learned from the available data. Unfortunately, learning a conditional distribution is generally acknowledged to be hard, since it does not decompose over the graphical structure of a Bayesian network classifier as does the joint distribution. For learning conditional distributions therefore, usually iterative discriminative methods are employed [37, 56, 79] which tend to be quite demanding from a computational point of view. Our MDL-FS function in contrast approximates the conditional distribution by means of an auxiliary Bayesian network which captures the strongest relationships between the attributes. With the function, both the structure of the Bayesian network classifier over all variables involved and the structure of the auxiliary network over the attributes are learned using a less demanding generative method. The conditional log-likelihood of the classifier then is approximated by the difference between the unconditional log-likelihood of the classifier and the log-likelihood of the auxiliary network.

We study the feature-selection behaviour of the MDL-FS function, compared to that of the MDL function, upon learning Bayesian network classifiers from data. We show that, unlike the MDL function, the MDL-FS function serves to identify redundancy and in essence is able to eliminate redundant attributes from different types of Bayesian network classifier. We show that the level of redundancy at which attributes are eliminated with the MDL-FS function is closely related to the complexity of the auxiliary network used. We further show that for a fixed set of attributes, the MDL-FS function is equivalent to the MDL function. The MDL-FS function therefore inherits the tendency of the MDL function to result in good Bayesian network classifiers.

We propose a variant of the MDL-FS score which is in fact the difference between the MDL score of a Bayesian network classifier and the MDL score of an auxiliary structure that models the strongest relationships between attributes. Using this variant, the MDL-FS score is the interplay between the (approximation of the) conditional log-likelihood term and the complexity of the model that represents the conditional log-likelihood. Here, we consider that the complexity of the conditional log-likelihood term is the complexity of the classifier from which we subtract the complexity of the auxiliary structure. We show that this variant of the MDL-FS score is a sum of redundancy relationships within some allowed noise that depends on the complexity of the used Bayesian structures. However, for the feature selection task this score has a worse behavior than for the previous variant since the complexity of the conditional log-likelihood is larger than the complexity of the classifier alone.

To compare the feature-selection behavior of the MDL-FS and MDL functions in a practical setting, we conducted various experiments in which we used different learning algorithms for constructing Naive Bayesian classifiers and TAN classifiers from various datasets. Our results indicate that the MDL-FS function indeed is more suited to the task of feature subset selection than the MDL function as it yields classifiers of comparably good or even significantly better performance with fewer attributes. Furthermore, using the second variant of the MDL-FS score we build performant selective TAN classifiers.

The first part of the thesis is organised as follows. In Section 2.1 we provide some background on

Bayesian networks and on Bayesian network classifiers more specifically; we further review the MDL function and present our notational conventions. In Section 2.2 we introduce the problem of feature subset selection and provide a formal definition of the concept of redundancy. We introduce our new MDL-FS function and study its relationship with the MDL function in Section 3.1. In Section 3.2, we investigate the feature-selection behavior of the MDL-FS function in general and we compare it with the behavior of the MDL function. In Section 3.3 we study how MDL-FS identifies and eliminates redundant attributes at various levels for two Bayesian network classifiers commonly used in practice: Naive Bayesian and TAN classifiers, respectively. Section 3.4 introduces the second variant of the MDL-FS score and investigate its properties. Section 3.6 reports on the results from our experiments. The first part ends with our concluding observations and remarks in Chapter 4.

2.1 Background

In this section, we provide some preliminaries on Bayesian networks and on Bayesian network classifiers more specifically. We further briefly review the construction of these graphical models from data and, upon doing so, focus on the various quality measures in use. We conclude this section with a discussion of the MDL function.

2.1.1 Bayesian networks and Bayesian network classifiers

We consider a set V of stochastic variables V_i, $i = 1, \ldots, n$, $n \geq 1$. We use $\Omega(V_i)$ to denote the set of all possible (discrete) values of the variable V_i; for ease of exposition, we assume a total ordering on the set $\Omega(V_i)$ and use v_i^k to denote the kth value of V_i. For any subset of variables $S \subseteq V$, we use $\Omega(S) = \times_{V_i \in S}\ \Omega(V_i)$ to denote the set of all joint value assignments to S. A *Bayesian network* over V now is a tuple $\mathcal{B} = (G, P)$ where G is a directed acyclic graph and P is a set of conditional probability distributions. In the digraph G, each vertex models a stochastic variable from V. The set of arcs captures probabilistic independence: for a topological sort of the digraph G, that is, for an ordering $V_1, \ldots, V_n$, $n \geq 1$, of its variables with $i < j$ for every arc $V_i \rightarrow V_j$ in G, we have that any variable V_i is independent of the preceding variables $V_1, \ldots, V_{i-1}$ given its parents in the graphical structure. Associated with the digraph is a set P of probability distributions: for each variable V_i are specified the conditional distributions $P(V_i \mid p(V_i))$ that describe the influence of the various assignments to the variable's parents $p(V_i)$ on the probabilities of the values of V_i itself. The network defines a unique joint probability distribution $P(V)$ over its variables with

$$P(V) = \prod_{V_i \in V} P(V_i \mid p(V_i))$$

Note that the thus defined probability distribution factorises over the network's digraph into separate conditional distributions.

Bayesian network classifiers are Bayesian networks of restricted topology that are tailored to solving classification problems. In a classification problem, instances described by a number of features have to be classified in one of several distinct predefined classes. We consider to this end a set A of stochastic variables A_i, called *attributes*, that are used to describe the features of the instances. We further have a designated variable C, called the *class variable*, that captures the various possible classes. *Bayesian network classifiers* now are defined over the set of variables $A \cup \{C\}$. Like a Bayesian network in general, they include a graphical structure that captures a probabilistic independence relation among the variables involved, and represent a joint probability distribution

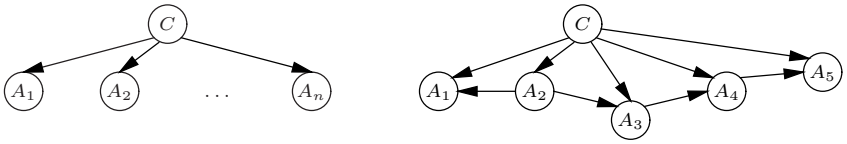

Figure 2.1: A Naive Bayesian classifier (left), and an example TAN classifier (right).

that is factorised in terms of this graphical structure. In this paper, we will focus on Naive Bayesian classifiers and Tree Augmented Network (TAN) classifiers more specifically.

A *Naive Bayesian classifier* over $A \cup \{C\}$ has for its graphical representation a tree-like structure with the variables $A \cup \{C\}$ for its nodes. The class variable is the root of the tree and each attribute has the class variable for its unique parent, as illustrated in Figure 2.1 on the left. The graphical structure of the classifier models the assumption that all attributes $A_i \in A$ are mutually independent given the class variable. The Naive Bayesian classifier specifies a prior probability distribution $P(C)$ for the class variable C and, for each attribute $A_i \in A$, it specifies a conditional distribution $P(A_i \mid C)$ over A_i given C. The joint probability distribution $P(C, A)$ defined by the classifier now equals

$$P(C, A) = P(C) \cdot \prod_{A_i \in A} P(A_i \mid C)$$

A *TAN classifier* over $A \cup \{C\}$ has for its graphical representation a directed acyclic graph in which the class variable is the unique root and in which each attribute has the class variable and at most one other attribute for its parents. The subgraph induced by the set of attributes, moreover, is a directed tree, termed the *attribute tree* of the classifier. An example TAN classifier is shown in Figure 2.1 on the right. Note that, if the class variable and its incident arcs are deleted from the graphical structure, the attributes constitute a tree; if, on the other hand, the arcs between the attributes are removed, we obtain a Naive Bayesian classifier. The TAN classifier specifies a prior probability distribution $P(C)$ for the class variable C and, for each attribute $A_i \in A$, it specifies a conditional distribution $P(A_i \mid p(A_i))$ over A_i given its parents $p(A_i)$ in the graphical structure. The joint probability distribution that is defined by the classifier equals

$$P(C, A) = P(C) \cdot \prod_{A_i \in A} P(A_i \mid p(A_i))$$

A classifier over $A \cup \{C\}$ in essence is a function $\mathcal{C}$: $\Omega(A) \rightarrow \Omega(C)$ that assigns a class value to joint value assignments to the set of attributes A. In the sequel, we assume that our Bayesian network classifiers build upon the *winner-takes-all rule*. Using this rule, they associate with each value assignment a^k to A, a class value c^* with $P(c^* \mid a^k) \geq P(c' \mid a^k)$ for all $c' \in \Omega(C)$; in case multiple class values give the largest conditional probability, the value that ranks highest among these in the ordering on $\Omega(C)$ is associated with a^k.

2.1.2 Learning Bayesian network classifiers

Bayesian network classifiers are typically constructed from a dataset in which instances of every-day problem solving have been recorded along with their associated classes. An *unlabelled instance* over the set of attributes A is an element of $\Omega(A)$. A *labelled instance* is composed of an unlabelled instance and an associated class value; it thus is an element of $\Omega(A \cup \{C\})$. In this paper we assume

labelled as well as unlabelled instances to be complete, that is, we assume that there are no missing values in the dataset nor in the new instances to be presented to the classifiers. For the learning task, we consider a *dataset* D with $N \geq 1$ labelled instances over $A \cup \{C\}$. With D, we associate the *counting function* N_D: $\cup_{S \subseteq A \cup \{C\}} \Omega(S) \rightarrow \mathbb{N}$ that associates with each value assignment s^k to S, the number of instances in D for which $S = s^k$; for $S = \varnothing$, we take the function value of N_D to be equal to N. The dataset D now induces a joint probability distribution $\hat{P}_D(C, A)$, termed the *observed* distribution, over $A \cup \{C\}$, with

$$\hat{P}_D(c^g, a^k) = \frac{N(c^g, a^k)}{N}$$

for all values c^g of C and all value assignments a^k to A. In the sequel, we will omit the subscript D from the counting function N_D and from the observed distribution $\hat{P}_D$ as long as ambiguity cannot occur. Learning a classifier from the dataset now amounts to selecting a classifier, from among a specific family of classifiers, that approximates the observed data. We assume that there might be noise - by means of any errors that interfere in the relationships between class and attributes - in the dataset D. For a review of the impact of the noise over the class variable and the attributes in a dataset we refer to Zhu and Wu [96]. For comparing alternative classifiers various different quality measures are in use. In this section, we review the measures that we will use throughout the paper. Before doing so, we briefly review the basic concepts of entropy and mutual information that underline many of these measures, and state some of their properties; for a more elaborate introduction, we refer the reader to any textbook on information theory.

Entropy and mutual information

The concept of *entropy* originates from information theory and describes the expected amount of information that is required to establish the value of a stochastic variable, or set of stochastic variables, to certainty. For an overview of these concepts we refer to Shannon [82]. We consider a set of variables X and a joint distribution P over X. The entropy $H_P(X)$ of X in P is defined as

$$H_P(X) = -\sum_{X^i \in \Omega(X)} P(X^i) \cdot \log P(X^i)$$

where $\log$ indicates a logarithm to the base 2 and $0 \cdot \log 0$ is taken to be equal to 0. The entropy function attains its *maximum* value for a uniform probability distribution over X. The larger the set $\Omega(X)$ of possible value assignments to X, the larger the maximum attained is; for a binary variable, for example, the maximum equals 1.00, while for a variable with 10 possible values, the maximum entropy is 3.32. The function further attains its *minimum* value for any degenerate distribution P over X with $P(X^j) = 1$ for some value assignment $X^j \in \Omega(X)$ and $P(X^i) = 0$ for all other assignments $X^i \in \Omega(X), i \neq j$. The minimum value equals 0, indicating that there is no uncertainty left as to the true value of X.

We now consider a set of stochastic variables $X \cup Y$ and a joint probability distribution P over $X \cup Y$. The amount of uncertainty as to the true value of X that is expected to remain after observing a value assignment for Y, is captured by the *conditional entropy* $H_P(X \mid Y)$ of X given Y in P; it is defined as

$$H_P(X \mid Y) = -\sum_{X^i \in \Omega(X), Y^j \in \Omega(Y)} P(X^i, Y^j) \cdot \log P(X^i \mid Y^j)$$

The entropy of the set of variables X is never expected to increase by observing a value assignment for the set Y, that is,

$$H_P(X \mid Y) \leq H_P(X)$$

for any (disjoint) sets X, Y; if the sets X and Y are independent in the probability distribution under consideration, then

$$H_P(X \mid Y) = H_P(X)$$

More in general, for any sets Y, Z with $Z \cap Y = \emptyset$, we have that

$$H_P(X \mid Y, Z) \leq H_P(X \mid Y)$$

The conditional entropy of X given Y has its minimum when observing any assignment to Y establishes the assignment for X to certainty.

The concept of *mutual information* is closely related to the concept of entropy. It captures the extent to which two stochastic variables, or sets of variables, are dependent. We consider two sets of variables X and Y, and a joint probability distribution P over $X \cup Y$. The mutual information $I_P(X;Y)$ of X and Y in P now is defined as

$$I_P(X;Y) = \sum_{X^i \in \Omega(X), Y^j \in \Omega(Y)} P(X^i, Y^j) \cdot \log \frac{P(X^i, Y^j)}{P(X^i) \cdot P(Y^j)}$$

The mutual information of two sets X and Y attains its *minimum* value for a probability distribution in which the two sets are independent; the minimum value attained is 0. The mutual information of X and Y is maximal for a probability distribution in which the two sets of variables are perfectly correlated, or maximally dependent. The *maximum* value attained then equals $H_P(X)$. In terms of the entropy function, the mutual information of X and Y equals

$$I_P(X;Y) = H_P(X) + H_P(Y) - H_P(X,Y)$$

The conditional mutual information of two sets X and Y given a third set Z is defined analogously; in terms of entropy, it equals

$$I_P(X;Y \mid Z) = H_P(X \mid Z) - H_P(X \mid Y, Z)$$

Quality measures

The main purpose in constructing a Bayesian network is to approximate, as well as possible, the unknown true joint probability distribution P over the variables involved. Upon constructing the network from data, for this purpose only an observed distribution $\hat{P}$ is available. Alternative networks then are compared by means of a *quality measure* that serves to express how well the represented distribution explains the data. The most commonly used quality measure is the *log-likelihood* measure that assigns to a network, given a particular dataset, a numerical value that is proportional to the probability of the dataset being generated by the joint probability distribution represented by the network. The log-likelihood of a network $\mathcal{B}$ given a dataset D is defined more formally as

$$LL(\mathcal{B} \mid D) = -N \cdot \sum_{V_i \in V} H_{\hat{P}}(V_i \mid p_{\mathcal{B}}(V_i))$$

where V is the set of variables included in the network and $p_{\mathcal{B}}(V_i)$ denotes the set of parents of V_i in the network's digraph. In essence, the entropy factor of the log-likelihood function pertains to the observed joint probability distribution factorised over the network's graphical structure. Writing $\hat{P}_{\mathcal{B}}$ for the thus factorised observed distribution, the log-likelihood thus equals

$$LL(\mathcal{B} \mid D) = -N \cdot H_{\hat{P}_{\mathcal{B}}}(V)$$

Since the joint probability distribution captured by the network factorises into separate conditional distributions, the log-likelihood also decomposes over the network's graphical structure, thereby allowing for ready computation.

While for constructing Bayesian networks in general the main purpose is to approximate the true joint distribution, for a Bayesian network classifier it is the conditional probability distribution of the class variable given the attributes that is of importance. Alternative classifiers therefore are to be compared as to their ability to describe the data for the various different classes. The *conditional log-likelihood* of a classifier $\mathcal{C}$ given a dataset D now is defined as

$$\mathit{CLL}(\mathcal{C} \mid D) = -N \cdot H_{\hat{P}_{\mathcal{C}}}(C \mid A)$$

where $\hat{P}_{\mathcal{C}}$ again denotes the observed joint distribution factorised over the classifier's graphical structure. Since the conditional probability distribution of the class variable given the attributes does not factorise over the graphical structure of a classifier, the conditional log-likelihood measure does not decompose into separate terms as does the unconditional log-likelihood measure. Because of its associated complexity of computation, the conditional log-likelihood measure is not used directly in practice.

An alternative measure that is often used for comparing the qualities of Bayesian network classifiers, is the *classification accuracy* [24, 19]. In essence, we define the classification accuracy of a classifier $\mathcal{C}$ to be the probability of an instance being labelled with its true class value, that is,

$$\mathit{accuracy}(\mathcal{C}) = \sum_{A^k \in \Omega(A)} P(A^k) \cdot \mathit{accuracy}(\mathcal{C}, A^k)$$

where

$$\mathit{accuracy}(\mathcal{C}, A^k) = \begin{cases} 1 & \text{if } \mathcal{C} \text{ returns the true class value for } A^k \\ 0 & \text{otherwise} \end{cases}$$

Note that the joint probability distribution $P(A^k)$ over all possible value assignments to A is readily established from the joint probability distribution over all variables involved:

$$P(A^k) = \sum_{C^g \in \Omega(C)} P(C^g, A^k)$$

Since upon constructing a Bayesian network classifier from data the true joint probability distribution is not known, it is approximated for practical purposes by the observed distribution. The class value included in a labelled instance in the dataset then is taken to be the true class value of the associated unlabelled instance.

The MDL Function

The well-known *minimum description length* (MDL) principle [10, 38, 30, 57, 74] is often employed as the basis for comparing the qualities of Bayesian networks in general. Since in this paper we build upon this principle, we briefly review the, more or less standard, two-parts MDL function.

Let V be a set of stochastic variables as before. Let D be a dataset of N labelled instances over V and let $\hat{P}(V)$ be the joint probability distribution observed in D. Let $\mathcal{B}$ be a Bayesian network over V. Then, the MDL score of the network with respect to the data is defined as

$$\mathit{MDL}(\mathcal{B} \mid D) = \frac{\log N}{2} \cdot |\mathcal{B}| - \mathit{LL}(\mathcal{B} \mid D)$$

where

$$|\mathcal{B}| = \sum_{V_i \in V} (|\Omega(V_i)| - 1) \cdot |\Omega(p_{\mathcal{B}}(V_i))|$$

with $p_{\mathcal{B}}(V_i)$ as before, and where $LL(\mathcal{B} \mid D)$ is the log-likelihood of the network given the data.

The MDL function originates from information theory where it is used for encoding a string of symbols with as few bits as possible [10, 38]. The basic idea underlying the function is the following. Among all the possible Bayesian networks over V, the best network to explain the given data is taken to be the one that minimises the sum of the length in bits of an encoding of the network itself and the length in bits of a description of the data encoded with the help of the network. The term $|\mathcal{B}|$ now captures the length of the encoding of the network and therefore is related to the network's complexity; the associated term $\frac{\log N}{2} \cdot |\mathcal{B}|$ is commonly known as the *penalty term* of the MDL function. The length of the description of the data equals the length of the encoding of the observed joint probability distribution $\hat{P}(V)$ factorised over the graphical structure of $\mathcal{B}$. The log-likelihood term $LL(\mathcal{B} \mid D)$ of the function captures the length of this encoding and therefore is related to the network's ability to explain the data. Often, the MDL function is defined to include a third term, $\log P(\mathcal{B})$, that captures prior information about the likelihood of the various possible networks [10]. In this paper, however, we assume this term to be constant and will not take it into further consideration.

The MDL score of a given network serves to indicate the network's quality with respect to the data under study. The smaller the score, the better the network is. The larger the value of the log-likelihood term, that is, the closer it is to zero, the better the network models the observed probability distribution. As a fully connected network perfectly matches the data, it will have the largest log-likelihood term. Such a network will generally show poor performance, however, as a result of overfitting. The penalty term now counterbalances the effect of the log-likelihood term within the MDL function since it increases in value as a network becomes more densely connected. For a network that is too simple, the values of both the penalty term and the log-likelihood term are rather small. For a network that is too complex, on the other hand, the values of the two terms are both quite large. Although the MDL function is often used by practitioners for comparing the qualities of alternative Bayesian networks, its use for this purpose is not without criticism. Van Allen and Greiner [91], for example, argue that the MDL function tends to underfit the data, unless the dataset under study is quite large or the true probability distribution is quite simple; Kearns et al. [48] express similar concerns.

The MDL function is used not just upon learning Bayesian networks, but upon learning Bayesian network classifiers as well [38, 30, 56]. We recall, however, that upon constructing Bayesian networks in general we are interested in the *joint* probability distribution over their variables. For classifiers, on the other hand, we are not so much interested in the joint distribution $P(C, A)$ over all variables involved but rather in the *conditional* probability distribution

$$P(C \mid A) = \frac{P(C, A)}{P(A)}$$

over the class variable given the attributes. Using the conditional distribution in practice, however, would raise computational as well as representational problems since it does not factorise over the classifier's graphical structure. Since basically only the term $P(C, A)$ of the conditional distribution contributes directly to the actual classification, as $P(A)$ does not include the class variable, generally the joint distribution is used for the learning task.

To conclude, we would like to note that the two-parts MDL function reviewed above is just one of the many forms of the minimum description length principle. An overview and comparison of the

various alternative forms is provided by Hansen and Yu [42]. The alternative MDL functions typically are generalisations of the two-parts function. Many of these alternative functions have been designed for other purposes and are hard to implement in the context of learning Bayesian networks from data. A possible exception is the normalised maximum log-likelihood (NML) function [76, 5] for multimodal (discrete) data, which has been tailored to Naive Bayesian classifiers [53]. Unfortunately, there are no known closed forms of the NML function for expressing more elaborate interactions between the variables than can be captured by this simple family of Bayesian network classifiers. Just like the two-parts MDL function, moreover, the NML function encodes the joint probability distribution over the class variable and the attributes rather than the conditional distribution. In the remainder of the paper, we will argue that the poor feature-selection behaviour of the two-parts MDL function originates from not using the conditional distribution. Our observations can thus be extended to the NML function and in fact to any form of the MDL function that captures the joint distribution over the variables involved.

2.2 Feature Subset Selection

We now define the problem of feature subset selection.We further introduce the concept of redundant attribute which we will use in the sequel for studying the feature-selection behaviour that is induced by various different quality measures.

2.2.1 The problem of feature subset selection

For our motivating example for doing feature selection in the context of the Bayesian network classifiers, we consider the task of constructing a Bayesian network classifier over a set of attributes A that contains two perfectly correlated attributes A_i and A_j where A_j is an exact copy of A_i. As argued before [60], by including both A_i and A_j in for example a Naive Bayesian classifier, A_i (or A_j alternatively) will have twice the influence of the other attributes, which may strongly bias the performance of the classifier. A possible way to improve the classification performance then is to eliminate one of the attributes A_i and A_j from the set A and to construct the classifier over the reduced set of attributes; the resulting classifier is called a *selective classifier*. Eliminating attributes upon constructing a classifier is commonly known as *feature subset selection*. We define the problem of feature subset selection more formally.

Definition 2.1 *Let A be a set of attributes, let C be a class variable, and let D be a set of labelled instances over $A \cup \{C\}$. Let $\mathcal{M}$ be a specific family of Bayesian network classifiers and let $\mathcal{R}$ be a performance measure on $\mathcal{M}$. The* problem of feature subset selection *for A and D given $\mathcal{M}$ and $\mathcal{R}$ is the problem of finding a minimum subset $S \subseteq A$ such that the selective classifier $\mathcal{C} \in \mathcal{M}$ constructed over S maximises performance on D according to the measure $\mathcal{R}$.*

From the definition we have that the problem of feature subset selection is restricted to a specific family of Bayesian network classifiers and to a specific performance measure. Example families of classifiers are the family of Naive Bayesian classifiers and the family of TAN classifiers, as reviewed in the previous section. Examples of performance measures are the classification accuracy and the conditional log-likelihood.

The problem of feature subset selection has been defined by various different researchers in many different ways [9, 90, 39, 28, 29, 17], focusing on different objectives. Many researchers [13, 47, 1, 95, 46, 54], for example, strive to achieve maximum classification accuracy. Tsamardinos and Aliferis [90], Koller and Sahami [51] and Yu and Liu [94], on the other hand, aim to find

a subset of attributes such that the conditional distribution of the class variable given this subset is the same as that given the full set of attributes; Fleuret's closely related goal [27] is to find a subset of attributes that minimises the conditional entropy of the class variable. Hall [41], Kononenko [52], Robnik-Sikonja and Kononenko [78], and Dy and Brodley [25] aim to find a minimum subset of attributes maximising a particular performance measure. For most of these objectives, the problem of feature subset selection is an instance of the more general problem of finding simpler models with better performance, which is known to be NP-hard in general [47]. We take for our objective finding a *minimum subset of attributes* for which *performance*, that is, either classification accuracy or conditional log-likelihood, of the resulting selective classifier is maximised.

Our definition of the problem of feature subset selection is related to the first definition proposed by Tsamardinos and Aliferis [90]. In our notation, they define a feature selection problem to be a tuple $\langle D, A \cup \{C\}, \mathit{alg}, \mathcal{R} \rangle$, where D is a dataset over the variables $A \cup \{C\}$, *alg* is the algorithm used to construct the classifier with, and $\mathcal{R}$ is a performance measure. A solution to the problem then is a subset of attributes $S \subseteq A$ such that the selective classifier over S that is constructed using *alg* maximises performance on D given $\mathcal{R}$. There are a number of differences between the two definitions, however. For example, we prefer sets with a minimum number of attributes, that is, of minimum cardinality; we further do not have any preference among sets with the same number of attributes as long as these give rise to the same classification performance. More importantly, however, we assume in our definition a fixed family of classifiers from among which a model is to be selected. Tsamardinos and Aliferis argue that practitioners would not like to solve a feature-selection task for a fixed family of classifiers and therefore do not restrict their definition. Our main motivation for including a fixed family of classifiers in our definition is that practitioners often are forced to select a model from among a fixed family for computational reasons or for lack of data. Another motivation for restricting our definition to a family of classifiers, is that it provides for studying the feature-selection behaviour of different quality measures in more detail. We further note that we do not specify a particular learning algorithm with our definition of the problem of feature subset selection. Our main motivation for not including a learning algorithm is that we do not want to capture the biases introduced by the heuristics involved into our definition. Defining the problem feature subset selection as a fundamental concept now allows us to study and compare the biases of the various learning algorithms in use.

Finding an appropriate subset of attributes for inclusion in a classifier amounts to searching the space of all possible selective classifiers, given some predefined measure of quality. Since this search space is infeasible large, often a heuristic algorithm is employed for its traversal. Various different algorithms have been proposed to this end. These algorithms essentially take one of two approaches [90, 47, 50, 39]. Within the *filter approach* [51, 90, 94], feature subset selection is performed in a pre-processing step; within the *wrapper approach* [47, 50, 58, 83], feature selection is merged with the learning algorithm. The difference between the two approaches in practice often lies in whether or not the algorithms employ the same measure for the selection of attributes and for measuring performance. In this paper, we will present our fundamental results from both a wrapper and a filter perspective. All algorithms used in this paper are characterised by their starting point(s) in the search space, by the search operator(s) they apply, and by their stopping criterion [9]. Possible starting points in the space of selective classifiers are the *empty classifier* that is built from the empty set of attributes and the *full classifier* that includes all attributes. If the starting point for the search is the empty classifier, then the algorithm typically applies the operator of adding a single attribute; the algorithm is said to perform *forward selection* [41, 50, 83]. If the starting point is the full classifier, on the other hand, the algorithm typically applies the operator of removing a single attribute; it then is said to perform *backward elimination* [51, 94]. The stopping criterion that is commonly employed with the various algorithms, is to stop the traversal of the search space as soon as application of the search

operators does no longer result in classifiers of improved quality. We will return to these algorithmic issues in Section 3.6 where we discuss our experimental results.

2.2.2 The concept of redundancy

Upon constructing a selective classifier, the set of attributes A under study is split into two subsets S and O, with $S \cup O = A$ and $S \cap O = \varnothing$. S is the subset of attributes that are selected to construct the classifier with and O is the subset of attributes that will not be incorporated in the classifier. The attributes included in S are deemed important, whereas the attributes from O are considered to be *redundant* for the classification task. We define our concept of redundancy.

Definition 2.2 *Let $A_i \in A$ be an attribute, let $S \subseteq A \setminus \{A_i\}$ be a subset of attributes, and let C be the class variable as before. Let D be a dataset of labelled instances over $A \cup \{C\}$. We say that A_i is* redundant *for C given S in D, if for every value a_i^k of A_i, for every value c^l of C, and for every value assignment s^j to S such that $N(a_i^k, s^j) > 0$, we have that*

$$\frac{N(a_i^k, s^j, c^l)}{N(a_i^k, s^j)} = \frac{N(s^j, c^l)}{N(s^j)}$$

For $|S| = m$, we say that A_i is redundant for C at level m. *We further say that A_i is* irredundant *for C given S in D if it is not redundant for C given S in D. If, for all subsets S with $|S| = m$, attribute A_i is not redundant for C given S, we say that A_i is irredundant for C* at level m.

We note that, if an attribute A_i is redundant for C given S in D, we have, in terms of probabilities, that $\hat{P}(C \mid A_i, S) = \hat{P}(C \mid S)$, that is, the class variable C is independent of A_i given S in the observed distribution. Moreover, if A_i is redundant for C given S and $N(s^j, c^l) > 0$ for all value assignments c^l and s^j, then

$$\frac{N(a_i^k, s^j, c^l)}{N(s^j, c^l)} = \frac{N(a_i^k, s^j)}{N(s^j)}$$

from which we have that $\hat{P}(A_i \mid S, C) = \hat{P}(A_i \mid S)$.

The following example illustrates our concept of redundancy as well as the different levels at which attributes can be redundant for a class variable.

Example 2.1 *We consider a classification problem with the binary attributes $A = \{A_1, \ldots, A_8\}$ and the binary class variable C. The class variable C is defined as $C = (A_4 \oplus A_1) \vee A_2$, where $\oplus$ denotes the XOR operator and $\vee$ the logical OR operator. Among the nine variables involved, there are some logical relationships and some probabilistic independence relationships. The logical relationships among the attributes are $A_6 = A_1 \vee A_2 \vee A_4$, $A_7 \equiv A_5$, and $A_8 \equiv A_2$. For the independence relationships, we have that A_3 is independent of C given A_2; A_3 further is unconditionally dependent of A_2 and of C. Given these relationships, there are 32 possible instances of the variables involved; these instances are shown in Table 2.1. We now assume that we have a dataset in which each possible instance occurs exactly once. From the dataset, we observe that the attributes A_1 and A_4 both are redundant for the class variable C at level 0; so, for all values $a_1^j \in \Omega(A_1)$, $a_4^k \in \Omega(A_4)$ and $c^l \in \Omega(C)$, we have that $N(c^l, a_1^j)/N(a_1^j) = N(c^l)/N$ and $N(c^l, a_4^k)/N(a_4^k) = N(c^l)/N$. A_1 and A_4 are irredundant for C at all higher levels, since for all subsets $S \subseteq A \setminus \{A_1, A_4\}$ there are values a_1^j, a_4^k, c^l, and $s^i \in \Omega(S)$, for which $N(c^l, a_1^j, a_4^k, s^i)/N(a_1^j, a_4^k, s^i) \neq N(c^l, s^i)/N(s^i)$. The attributes A_5 and A_7 are redundant for the class variable at all possible levels, that is, from level 0 to level $|A| - 1 = 7$. The attribute A_2 is irredundant for C at all levels including level 0. A_3 and*

C	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
0	0	0	1	0	1	0	1	0
1	0	0	1	1	0	1	0	0
1	0	1	1	0	0	1	0	1
1	0	1	1	1	1	1	1	1
1	1	0	1	0	0	1	0	0
0	1	0	0	1	0	1	0	0
1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	0
1	0	1	1	0	1	1	1	1
1	0	1	1	1	0	1	0	1
1	1	0	0	0	1	1	1	0
0	1	0	1	1	1	1	1	0
1	1	1	1	0	0	1	0	1
1	1	1	1	1	0	1	0	1
0	0	0	1	0	0	0	0	0
1	0	0	1	1	1	1	1	0
1	0	1	1	0	1	1	1	1
1	0	1	1	1	0	1	0	1
1	1	0	1	0	1	1	1	0
0	1	0	0	1	1	1	1	0
1	1	1	1	0	0	1	0	1
1	1	1	1	1	0	1	0	1
0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	1	0	0
1	0	1	1	0	0	1	0	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	1	0	0
0	1	0	1	1	0	1	0	0
1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1

Table 2.1: The example dataset illustrating the concept of redundancy

A_8 are irredundant for C at level 0, but redundant at all higher levels given any subset of attributes that contains A_2. The attribute A_6 is irredundant for C at all levels below level 3; at level 3 and higher, it is redundant for C given any subset of attributes that contains A_1, A_2 and A_4. We note that the attributes A_1, A_2 and A_4 serve to completely determine the value of the class variable C. The Bayesian network classifier with the smallest number of attributes giving the highest classification accuracy is shown in Figure 2.2. □

We are not the first to define a concept of redundancy in the context of feature subset selection [9, 90, 41, 47, 50, 51, 28]. The various concepts in use differ in whether they address redundancy with respect to the class variable in terms of single attributes or in terms of sets of attributes. Using a concept of redundancy in terms of single attributes involves studying the relationship between the class variable and each attribute separately. Using a concept of redundancy in terms of subsets of attributes involves investigating all possible subsets of attributes and, as a consequence, is much more demanding from a computational point of view. Caruana and Freitas conducted experiments using various different concepts of redundancy and reported better results from using sets of attributes [13]. Tsamardinos and Aliferis [90] studied redundancy in terms of sets of attributes from a more fundamental perspective and found that the concept does not behave monotonically with respect to taking supersets of attributes, that is, a redundant subset of attributes may become irredundant by including an additional attribute, and vice versa. We have decided, in accordance with this observation, to explicitly distinguish between redundancy at various different levels.

We compare our concept of redundancy to some of the other concepts that have been defined in terms of sets of attributes. John, Kohavi and Pfleger [47], for example, defined the closely related

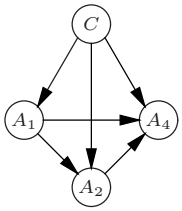

Figure 2.2: The optimal Bayesian network classifier for our example dataset

concepts of relevance and irrelevance. Upon introducing their concepts, they argued that a simple distinction between relevant and irrelevant attributes does not suffice to partition the set of attributes into subsets that provide for studying the differences in feature-selection behaviour of alternative algorithms and quality measures. They therefore introduce two degrees of relevance. They define an attribute A_i to be *strongly relevant* for the class variable C in D, if there exist values a_i^k and c^l for A_i and C respectively, and a value assignment s^j to the set $S = A \backslash \{A_i\}$ such that $\frac{N(a_i^k, s^j, c^l)}{N(a_i^k, s^j)} \neq \frac{N(s^j, c^l)}{N(s^j)}$ The attribute A_i is called *weakly relevant* for the class variable C in D, if it is not strongly relevant for C and there exist values a_i^k and c^l for A_i and C respectively, and a value assignment t^j to some set $T \subset A \setminus \{A_i\}$ such that $\frac{N(a_i^k, t^j, c^l)}{N(a_i^k, t^j)} \neq \frac{N(t^j, c^l)}{N(t^j)}$. Any attribute that is neither strongly nor weakly relevant for the class variable C is called *irrelevant* for C in D. Our concept of redundancy, as introduced in Definition 2.2, is closely related to the concepts defined by John, Kohavi and Pfleger. It is readily shown that an attribute A_i is strongly relevant for the class variable C if and only if it is irredundant for C at level $|A| - 1$. Moreover, the attribute A_i is weakly relevant for C if and only if it is redundant for C at level $|A| - 1$ and irredundant at some level $m < |A| - 1$. Moreover, the attribute A_i is irrelevant for C if and only if it is redundant for C at *all* levels. For the classification problem from Example 2.1, we find for instance that the attributes A_1, A_2 and A_4 are strongly relevant for the class variable; the attributes A_3, A_6 and A_8 are just weakly relevant and the attributes A_5 and A_7 are irrelevant for the classification task. Note that with the concept of John, Kohavi and Pfleger, no distinction is made between the attributes A_3 and A_8 on the one hand and A_6 on the other hand. With our new concept of redundancy, however, different levels of redundancy are identified for these attributes. In contrast with the concept of John, Kohavi and Pfleger, therefore, our concept provides for studying redundancy at *all* possible levels $0, \ldots, |A| - 1$ separately. In the sequel we will illustrate the importance of distinguishing between these levels for learning different types of classifier.

Alternative definitions of redundancy have also been proposed by Tsamardinos and Aliferis [90] and by Koller and Sahami [51]. Tsamardinos and Aliferis relate the conditional probability of the class variable given a set of attributes to conditional independence and build their concept of redundancy on the associated concept of Markov blanket. Given a dataset of labelled instances, the Markov blanket of the class variable is the minimal set of attributes which, upon value assignment, completely substitutes the influences of the other attributes on the class variable; given its Markov blanket, therefore, the class variable is independent of all other attributes. Tsamardinos and Aliferis now showed that, for any probability distribution that is faithful to a Bayesian network, the Markov blanket of the class variable coincides with the set of strongly relevant attributes; a distribution is said to be faithful to a Bayesian network if all dependences and independences embedded in the distribution can be captured by a graphical structure. They further showed that, in a Bayesian network faithfully modelling the distribution, an attribute A_i is weakly relevant for the class variable C if and only if A_i is not strongly relevant and there is an undirected path from A_i to C. The concept of redundancy defined by Tsamardinos and Aliferis thus in essence is equivalent to that of John, Kohavi and Pfleger for any

probability distribution that is faithful to a Bayesian network. For the classification problem from Example 2.1, we have that the observed probability distribution is faithful to a Bayesian network. Using the concept of redundancy defined by Tsamardinos and Aliferis, therefore, would result in the same selection of attributes as mentioned above. For an observed distribution that is not faithful to a Bayesian network, however, the Markov blanket of the class variable may not be unique. Moreover, any of the multiple blankets may then include weakly relevant attributes in addition to strongly relevant ones [94]. Building upon the concept of Markov blanket would then not result in a minimal subset of attributes shielding the influences of the other attributes from the class variable.

2.2.3 The issue of noise

When constructing a selective classifier in practice, the relationship between an attribute A_i and the class variable C, as captured by the available data, is investigated. Using our definition, the attribute can be either redundant or irredundant for the class variable, that is, it can be either (conditionally) independent or dependent of C. The (in)dependences are established from a dataset of instances that are assumed to have been generated from an unknown true probability distribution. As the dataset under study is finite, however, it may not reflect the (in)dependences from the true distribution exactly; the dataset then is said to include *noise* [96]. More specifically, the attribute A_i may be independent of the class variable C in the true distribution, yet appear to be irredundant in the observed distribution, for example, due to chance of observed instances or to the misclassified instances. Attributes that have a very weak dependence of the class variable in the observed distribution therefore, may in fact be independent. To provide for feature subset selection in a practical setting, we introduce the concept of *redundancy within an allowed amount of noise*.

Definition 2.3 *Let $A_i \in A$ be an attribute, let $S \subseteq A \setminus \{A_i\}$ be a subset of attributes, and let C be the class variable as before. Let D be a dataset of N labelled instances over $A \cup \{C\}$. Let $\xi(A_i, C, S, N) > 0$ be a threshold value for the allowed amount of noise. We say that A_i is* redundant *for C given S in D* within the allowed amount of noise $\xi(A_i, C, S, N)$*, if*

$$H_{\hat{P}}(C \mid S) - H_{\hat{P}}(C \mid A_i, S) < \xi(C, A_i, S, N)$$

Otherwise, we say that A_i is irredundant for C given S.

From the above definition, we have that an attribute A_i is said to be redundant for the class variable C given S within some allowed amount of noise ξ, if obtaining a value for A_i serves to reduce the conditional entropy of C given S by at most ξ. Note that if obtaining a value for A_i does not reduce the entropy of C given S at all, that is, if $H_{\hat{P}}(C \mid S) - H_{\hat{P}}(C \mid A_i, S) = 0$, we have that A_i is simply redundant for C given S. Since the conditional entropy depends on the number of values that the variables involved can adopt [52], we have defined the allowed amount of noise to be functionally dependent of A_i, C and S. The function is also taken to be dependent of the number of observed instances, since we would like to allow less noise for larger datasets in which the true (in)dependences are better represented. The threshold function may have many different forms; we will return to this observation in subsequent sections where we analyse the feature-selection behaviour of the MDL function.

Note that $H_{\hat{P}}(C \mid S) - H_{\hat{P}}(C \mid A_i, S) = H_{\hat{P}}(A_i \mid S) - H_{\hat{P}}(A_i \mid C, S)$. We further call the term $H_{\hat{P}}(A_i \mid S) - H_{\hat{P}}(A_i \mid C, S) - \xi(C, A_i, S, N)$ the *amount of irredundancy* the attribute A_i has for C given the attribute set S within the allowed noise level $\xi(C, A_i, S, N)$. We observe that a negative amount of irredundancy corresponds to redundancy of the attribute A_i for C given S within $\xi(C, A_i, S, N)$, whereas a positive amount of irredundancy corresponds with irredundancy.

Zhu and Wu [96] experimentally study the relationship between the (in)dependency between attributes and the class variable and the impact of noise over the performance of a classifier. In their study, they make the assumption that attributes are independent of each other given the class variable. They show that the stronger the relationship between an attribute and the class variable the more impact the noise of this attribute has over the classifier. Our definition of noise also captures the relationships between the class variable and the involved attributes. Furthermore, it generalises the above observation by considering also the dependencies between the attributes given the class variable.

By redefining our concept of redundancy, we explicitly provide for handling a limited amount of noise in a dataset under study. Another approach to the problem of insufficient data is to not allow explicitly for noise, but to use heuristic algorithms for establishing redundancy. Such an algorithm for example never studies a larger number of variables at the same time. In the next sections, we compare the behaviour of our enhanced concept of redundancy with various feature-selection heuristics on the problem from Example 2.1 and on five more realistic datasets.

Chapter 3

MDL for feature selection

In this chapter, we introduce and analyze a MDL function for performing feature selection for Bayesian network classifiers. After introducing the new MDL function in the next section, we investigate the general properties of this new function in Section 3.2. We study the properties of MDL-FS on practical Bayesian network classifiers (e.g. Naive Bayes and TANs) in Section 3.3. We introduce a MDL function suited for learning Bayesian network classifiers in Section 3.4. In the last section of this chapter, we discuss some experimental results.

3.1 An MDL-based quality measure for feature subset selection

The minimum description length principle is often employed as the basis for comparing the qualities of Bayesian network classifiers, and of alternative selective classifiers more specifically. We recall that the MDL function models the joint probability distribution over all the variables of a classifier. As a consequence, the function prefers classifiers in which the relatively strong relationships among all variables are properly captured. We have argued in Section 2.1 however, that for classification purposes we are not so much interested in the joint distribution over all variables, but rather in the conditional distribution over the class variable given the attributes. In the context of feature selection, therefore, we would prefer classifiers in which the strong relationships of the attributes with the class variable are properly captured. Strong relationships among attributes that have no bearing on the class variable are not of interest; in fact, including these relationships could bias the classifier. Because of its use of the joint probability distribution, the MDL function would nonetheless value and capture any strong relationships among the attributes. The MDL function as a consequence is not really suitable for constructing selective Bayesian network classifiers. In fact, the MDL function is able to identify and eliminate attributes that are redundant for the class variable at level 0 only. We will return to this observation in Section 3.2.

Building upon the assumption that the relatively poor feature-selection behaviour of the MDL function originates, to at least some extent, from not using the conditional probability distribution. For that we introduce a new quality measure, called *MDL-FS*, that is tailored to feature selection [21, 20]. The MDL-FS function in essence is based upon the same ideas as the MDL function. Like the MDL function, it captures the joint probability distribution $P(C, A)$ over all variables involved in a log-likelihood term. In addition however, it captures the joint probability distribution $P(A)$ over just the attributes. We note that while the joint distribution $P(C, A)$ factorises over the structure of the classifier under study, the distribution $P(A)$ does not; to allow for ease of computation, the function therefore uses an *auxiliary Bayesian network* to factorise $P(A)$. The function now establishes the differ-

ence between the log-likelihood of the probability distribution $P(C, A)$ and the log-likelihood of the distribution $P(A)$, and thereby effectively models the conditional probability distribution $P(C \mid A)$. Informally speaking, by capturing the difference between the two log-likelihood terms, the strengths of the relationships among the attributes themselves are eliminated from the joint distribution. In contrast with the MDL function, therefore, the MDL-FS function will identify and remove attributes that are redundant for the class variable yet strongly related to one or more other attributes.

In Section 3.1.1, we introduce the MDL-FS function. In Section 3.1.2, we study the relationships of the conditional log-likelihood term of the function and the conditional distribution of the class variable given the set of attributes; we will argue more specifically that the former may be considered an approximation of the latter.

3.1.1 The MDL-FS function for feature selection

We formally define the MDL-FS function.

Definition 3.1 *Let A be a set of attributes and let C be the class variable as before. Let D be a dataset of N labelled instances over $A \cup \{C\}$. Let $\mathcal{C}$ be a Bayesian network classifier over $A \cup \{C\}$ and let $\mathcal{S}$ be a Bayesian network over A. Then,*

$$\textit{MDL-FS}(\mathcal{C}, \mathcal{S} \mid D) = \frac{\log N}{2} \cdot |\mathcal{C}| - \textit{CLL}(\mathcal{C}, \mathcal{S} \mid D)$$

where $|\mathcal{C}|$ is as before and

$$\textit{CLL}(\mathcal{C}, \mathcal{S} \mid D) = \textit{LL}(\mathcal{C} \mid D) - \textit{LL}(\mathcal{S} \mid D)$$

with $\textit{LL}(\mathcal{C} \mid D)$ as before and

$$\textit{LL}(\mathcal{S} \mid D) = -N \cdot \sum_{A_i \in A} H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i))$$

where, for each attribute $A_i \in A$, the set $p_{\mathcal{S}}(A_i)$ is the set of parents of A_i in the graphical structure of the network $\mathcal{S}$.

The basic idea underlying the MDL-FS function is the same as that of the MDL function. The MDL-FS function also includes a *penalty term* to capture the length of an encoding of the classifier and a term that indicates the length of an encoding of the observed probability distribution given the classifier. The latter term in essence captures the observed conditional distribution and is called the *conditional log-likelihood term* of the MDL-FS function. Like the MDL score, the MDL-FS score of a Bayesian network classifier indicates the classifier's quality with respect to the data under study. The smaller the score, the better the classifier is.

For the conditional log-likelihood term, we have that the larger the value of the term $\textit{LL}(\mathcal{C} \mid D)$, that is, the closer it is to zero, the better the classifier models the observed joint probability distribution, as we have argued before in Section 2.1. The term therefore tends to approach zero for classifiers with a complex graphical structure and to be quite small for simpler models. The term $\textit{LL}(\mathcal{S} \mid D)$ also attains its minimum value for the simplest Bayesian network and its maximum value for a fully connected model. The maximum value of the conditional log-likelihood term therefore is obtained for a fully connected Bayesian network classifier and an empty auxiliary Bayesian network without any arcs. Now, to achieve a small score for a classifier, we basically would like to maximise the conditional log-likelihood term of the MDL-FS function. From the above considerations however,

we must conclude that we cannot simply maximise the term, as the function does not suggest any reason for using a more complex auxiliary network than the empty one. Yet, using an empty auxiliary network would not meet our purpose of capturing the relationships among the attributes from the joint distribution: for this purpose, a more complex auxiliary network is required. As the MDL-FS function has no control over the complexity of the auxiliary network to be used, in practical applications we propose to set a specific family of auxiliary networks beforehand. Since we would like to eliminate the influence of $P(A)$ from $P(C, A)$, we should select from this family a maximum log-likelihood networks. We thus have to maximise the log-likelihood of both the classifier and the auxiliary networks to model a conditional log-likelihood term suited for feature subset selection.

Like the MDL function, the MDL-FS function includes a penalty term to counterbalance the effect of the conditional log-likelihood term. From the definition of the MDL-FS function, we observe that this penalty term captures just the complexity of the classifier and not the complexity of the auxiliary network. We have decided not to include a penalty term for the auxiliary network since we are basically interested in the complexity of the classifier only. A difference of penalty terms for the complexities of the classifier and the auxiliary network however, would serve to more evenly counterbalance the difference between the log-likelihoods of the two networks. A quality measure that includes such a difference of penalty terms would amount to taking the difference of the MDL score of the classifier and the MDL score of the auxiliary network. The penalty term for the auxiliary network would then have a negative effect on the feature-selection behaviour of the MDL-FS function in the sense that it would become less selective. We discuss the properties of such an alternative function in Section 3.4.

3.1.2 Comparing the conditional log-likelihood with conditional entropy

Upon reviewing the MDL-FS function, we have argued that its conditional log-likelihood term in essence models the log-likelihood of the conditional probability distribution of the class variable given the attributes. In this section, we show that for a fully connected classifier and a fully connected auxiliary network, the term indeed models the log-likelihood of the conditional distribution. In practical applications, fully connected classifiers have major disadvantages: in addition to the large number of data required for their construction, these classifiers tend to overfit the available data and to show poor generalisation performance. As a consequence, they are hardly ever used in practice. Our result therefore serves to give a fundamental insight only. We will further argue that for classifiers and auxiliary networks that do not accurately capture all information from the data, the conditional log-likelihood term can only be looked upon as an approximation of the log-likelihood of the conditional distribution.

Proposition 3.1 *Let A be a set of attributes and let C be the class variable as before. Let D be a dataset of N labelled instances over $A \cup \{C\}$. Let $\mathcal{C}$ be a fully connected Bayesian network classifier over $A \cup \{C\}$ and let $\mathcal{S}$ be a fully connected Bayesian network over A. Then,*

$$CLL(\mathcal{C}, \mathcal{S} \mid D) = -N \cdot H_{\hat{P}}(C \mid A_1, \ldots, A_n)$$

Proof. Since the Bayesian network classifier $\mathcal{C}$ is fully connected, we have that

$$LL(\mathcal{C} \mid D) = -N \cdot H_{\hat{P}}(C, A_1, \ldots, A_n) =$$

$$= -N \cdot (H_{\hat{P}}(C \mid A_1, \ldots, A_n) + H_{\hat{P}}(A_1 \mid A_2, \ldots, A_n) + \ldots + H_{\hat{P}}(A_n))$$

For the Bayesian network $\mathcal{S}$ moreover, we have that

$$LL(\mathcal{S} \mid D) = -N \cdot H_{\hat{P}}(A_1, A_2, \ldots, A_n)$$

$$= -N \cdot (H_{\hat{P}}(A_1 \mid A_2, \ldots, A_n) + \ldots + H_{\hat{P}}(A_n))$$

For the conditional log-likelihood term of the MDL-FS function, we thus find that

$$CLL(\mathcal{C}, \mathcal{S} \mid D) = -N \cdot H_{\hat{P}}(C \mid A_1, \ldots, A_n)$$

as stated above. □

From the previous proposition, we have that for fully connected classifiers and fully connected auxiliary networks, the conditional log-likelihood term of the MDL-FS function accurately models the log-likelihood of the conditional distribution. As we have argued above, in practical applications classifiers of a simpler complexity than fully connected ones are used. For these classifiers, the previous proposition no longer holds and the conditional log-likelihood term of the MDL-FS function may differ from the log-likelihood of the conditional distribution. A less complex classifier may have a smaller log-likelihood than a fully connected one. The conditional log-likelihood term then decreases compared to the log-likelihood of the conditional distribution. A less complex auxiliary network may also have a smaller log-likelihood than a fully connected one. The conditional log-likelihood term then increases. For a less complex classifier and an associated less complex auxiliary network therefore, the conditional log-likelihood term can be either smaller or larger than the log-likelihood of the conditional distribution. If the auxiliary network models weaker relationships between the attributes than the classifier, for example, the conditional log-likelihood term will be larger than the conditional entropy. If the relationships between the attributes are stronger in the auxiliary network, then the conditional log-likelihood term will be smaller than the log-likelihood of the conditional distribution.

3.2 The feature-selection behaviour of the MDL-FS and MDL functions in general

We begin by studying the feature-selection behaviour of the MDL-FS function for complete Bayesian network classifiers and auxiliary structures, to review in an informal way some of its general properties. Recall that fully connected networks perfectly model the data and are of maximum log-likelihood for that data but, for pragmatical reasons, are seldom used in practice; we will substantiate the reviewed properties in the subsequent sections for more commonly used classifiers. In this section, we study the ability of the MDL and MDL-FS functions to identify and eliminate redundant attributes at different levels. We will argue that the MDL function tends to eliminate from a Bayesian network classifier only attributes that are redundant at level 0 for the class variable as well as for the other attributes. The MDL-FS function overcomes this drawback by comparing the strength of the relationship between an attribute and its parents in the classifier with the strength of the relationship between an attribute and its parents in the auxiliary structure. We show that MDL-FS tends to eliminate from fully connected Bayesian network classifiers the attributes that are redundant for the class variable at the highest level within an allowed amount of noise determined by the penalty term by using a fully connected auxiliary network whereas MDL tends to not eliminate these attributes. We find that the level of the eliminated redundant attributes with the MDL-FS score depends on the complexity of the auxiliary network: with a fully connected auxiliary network, it eliminates redundant attributes at level $|A| - 1$, whereas with an empty auxiliary network, MDL-FS eliminates attributes redundant at level 0 for the class variable and for the other variables from the attributes set.

We investigate the feature selection behavior of the MDL function, in Section 3.2.1; in Section 3.2.2 we study the behaviour of the MDL-FS function.

3.2.1 The feature-selection behaviour of the MDL function

In this section, we investigate the feature-selection behaviour of the MDL function. We will show that the MDL function is able to identify and eliminate only attributes that are redundant at level 0 for the class variable as well as for the other attributes. Before presenting this result, we observe that, to allow for comparing the MDL and/or MDL-FS scores of two classifiers, they both need to capture a joint probability distribution over the same set of variables. When comparing the score of a selective classifier with the score of a classifier that includes more attributes therefore, we look upon the selective classifier as being extended with the deleted attributes by means of nodes without any incident arcs.

Proposition 3.2 *Let A be a set of attributes, let C be the class variable, and let D be a dataset of N labeled instances over $A \cup \{C\}$ as before. Let $\mathcal{C}$ be a Bayesian network classifier over $A \cup \{C\}$ and let $A_i \in A$ be an attribute in $\mathcal{C}$ with the set of parents $p_{\mathcal{C}}(A_i)$ and the set of children $c_{\mathcal{C}}(A_i)$. From $\mathcal{C}$, we construct the selective classifier $\mathcal{C}^-$ by deleting the incident arcs of A_i. Then,*

$$\mathit{MDL}(\mathcal{C}^- \mid D) < \mathit{MDL}(\mathcal{C} \mid D)$$

if and only if

$$\left[H_{\hat{P}}(A_i) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - \frac{\log N}{2 \cdot N} \cdot (|\Omega(A_i)| - 1) \cdot \Big(\prod_{A_j \in p_{\mathcal{C}}(A_i)} |\Omega(A_j)| \; - \; 1 \Big) \right] +$$

$$\sum_{A_k \in c_{\mathcal{C}}(A_i)} \Bigg[\quad H_{\hat{P}}(A_i \mid p_{\mathcal{C}^-}(A_k)) - H_{\hat{P}}(A_i \mid A_k, p_{\mathcal{C}^-}(A_k)) -$$

$$\frac{\log N}{2 \cdot N} \cdot (|\Omega(A_k)| - 1) \cdot \prod_{A_j \in p_{\mathcal{C}}(A_k) \setminus \{A_i\}} |\Omega(A_j)| \cdot (|\Omega(A_i)| - 1) \Bigg] < 0$$

Proof. We begin by observing that, since the two classifiers differ only in the incident arcs for the attribute A_i, the difference of their MDL scores pertains to just A_i and its parents and children. To investigate the difference of the two scores, we now study the differences of their log-likelihood terms and of their penalty terms separately. The difference of the log-likelihood terms for the two classifiers equals

$$LL(\mathcal{C} \mid D) - LL(\mathcal{C}^- \mid D) = -N \cdot \big(H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - H_{\hat{P}}(A_i) \big) -$$
$$N \cdot \sum_{A_k \in c_{\mathcal{C}}(A_i)} \big(H_{\hat{P}}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{C}^-}(A_k)) \big)$$

The difference of the penalty terms of the two classifiers equals

$$\frac{\log N}{2} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) = \frac{\log N}{2} \cdot \Bigg((|\Omega(A_i)| - 1) \cdot \Bigg(\prod_{A_j \in p_{\mathcal{C}}(A_i)} |\Omega(A_j)| \; - \; 1 \Bigg) +$$

$$\sum_{A_k \in c_{\mathcal{C}}(A_i)} (|\Omega(A_k)| - 1) \cdot \prod_{A_j \in p_{\mathcal{C}}(A_k) \setminus \{A_i\}} |\Omega(A_j)| \cdot (|\Omega(A_i)| \; - \; 1) \Bigg)$$

Note that the difference of the two penalty terms is positive since the classifier $\mathcal{C}$ is more complex than the selective classifier $\mathcal{C}^-$. Using

$$\mathit{MDL}(\mathcal{C} \mid D) - \mathit{MDL}(\mathcal{C}^- \mid D) = \frac{\log N}{2} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) - \big(\mathit{LL}(\mathcal{C} \mid D) - \mathit{LL}(\mathcal{C}^- \mid D)\big)$$

and

$$H_{\hat{P}}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{C}^-}(A_k)) = H_{\hat{P}}(A_k, A_i, p_{\mathcal{C}^-}(A_k)) - H_{\hat{P}}(A_i, p_{\mathcal{C}^-}(A_k)) +$$

$$H_{\hat{P}}(A_k, p_{\mathcal{C}^-}(A_k)) - H_{\hat{P}}(p_{\mathcal{C}^-}(A_k)) = H_{\hat{P}}(A_i \mid A_k, p_{\mathcal{C}^-}(A_k)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}^-}(A_k))$$

for each attribute $A_k \in c_{\mathcal{C}}(A_i)$, where $p_{\mathcal{C}}(A_k) = \{A_i\} \cup p_{\mathcal{C}^-}(A_k)$, we now straightforwardly obtain the proposition's inequality. □

By definition we have that the MDL function prefers the selective classifier $\mathcal{C}^-$ over $\mathcal{C}$ if and only if $\mathcal{C}^-$ has a smaller MDL score than $\mathcal{C}$, that is, if and only if $\mathit{MDL}(\mathcal{C}^- \mid D) < \mathit{MDL}(\mathcal{C} \mid D)$ for the dataset D under consideration. The difference $\log N/2 \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$ of the penalty terms for the classifier $\mathcal{C}$ and the selective classifier $\mathcal{C}^-$ is greater than zero since $\mathcal{C}$ has a complexer structure than $\mathcal{C}^-$. The difference $LL(\mathcal{C} \mid D) - LL(\mathcal{C}^- \mid D)$ of the two log-likelihood terms also is greater than 0 because the classifier $\mathcal{C}$ captures the observed joint probability distribution at least as accurately as the selective classifier $\mathcal{C}^-$. The proposition now indicates under which condition the additional complexity of $\mathcal{C}$ is no longer counterbalanced by its increased log-likelihood.

We study the condition stated in the proposition in some closer detail. We observe that the condition basically pertains to the strengths of the relationships of the attribute A_i with the other attributes. Informally speaking, the stronger the relationships of A_i with its neighboring attributes in $\mathcal{C}$, the more the observation of a value assignment to these attributes can contribute to resolving the uncertainty as to the value of A_i. The stronger the relationships of A_i with its neighbouring attributes, therefore, the more likely the inequality stated in the proposition does not hold and the full classifier is preferred over the selective one. The next corollary now quantifies the maximal amount of irredundancy the attribute can have with its neighbours before it is effectively removed by the MDL function.

Corollary 3.1 *Let $\mathcal{C}$, $\mathcal{C}^-$ and A_i be as in Proposition 3.2. The selective classifier $\mathcal{C}^-$ is preferred over the full classifier $\mathcal{C}$ if only if*

- *the attribute A_i is redundant at level 0 for the variables in its set of parents $p_{\mathcal{C}}(A_i)$ in $\mathcal{C}$ within the allowed amount of noise $\xi(A_i, p_{\mathcal{C}}(A_i), \varnothing, N) < \log N/(2 \cdot N) \cdot (|\Omega(A_i)| - 1) \cdot \Big(\prod_{A_j \in p_{\mathcal{C}}(A_i)} |\Omega(A_j)| - 1\Big)$; and*

- *the attribute A_i is redundant for each child attribute $A_k \in c_{\mathcal{C}}(A_i)$ given A_k's other parents in $\mathcal{C}$ within the allowed amount of noise $\xi(A_i, A_k, \varnothing, N) < \log N/(2 \cdot N) \cdot (|\Omega(A_k)| - 1) \cdot \prod_{A_j \in p_{\mathcal{C}}(A_k) \setminus \{A_i\}} |\Omega(A_j)| \cdot (|\Omega(A_i)| - 1)$.*

From the property stated in the corollary, we conclude that, upon feature selection, an attribute is removed by the MDL function *only if* it is redundant at level 0 for *all other variables*, within an amount of noise that is dependent of the structure of the classifier. For Naive Bayesian classifiers, the function will thus serve to remove attributes that are redundant for the class variable C at level 0, since in such a restricted classifier the attributes are assumed to be mutually independent given C. For more complex Bayesian network classifiers, however, attributes that are redundant for the class variable at various levels will not be removed unless these attributes are redundant for all other attributes as well. We conclude that the MDL function is not very well suited for the task of identifying and removing attributes that are redundant for the class variable.

3.2.2 The feature-subsection behaviour of the MDL-FS function

In this section, we study the feature-selection behaviour of the MDL-FS function in detail. Before doing so, we relate the function to the MDL function and show under which conditions the two functions exhibit the same behaviour.

We have argued in Section 3.1 that the MDL-FS function is closely related to the MDL function and differs from this function mainly in that it captures, in addition to the joint probability distribution over the set of all variables, also the joint distribution over just the attributes. We now show that upon comparing classifiers over the same set of variables, the two functions exhibit the same preference behaviour as long as the MDL-FS function uses auxiliary networks that have the same log-likelihood given the data.

Proposition 3.3 *Let A be a set of attributes and let C be the class variable as before. Let D be a dataset of labeled instances over $A \cup \{C\}$. Let $\mathcal{C}$ and $\mathcal{C}'$ be two Bayesian network classifiers over $A \cup \{C\}$. Let $\mathcal{S}$ and $\mathcal{S}'$ be two Bayesian networks over A with $LL(\mathcal{S} \mid D) = LL(\mathcal{S}' \mid D)$. Then,*

$$\mathit{MDL}(\mathcal{C} \mid D) - \mathit{MDL}(\mathcal{C}' \mid D) = \mathit{MDL\text{-}FS}(\mathcal{C}, \mathcal{S} \mid D) - \mathit{MDL\text{-}FS}(\mathcal{C}', \mathcal{S}' \mid D)$$

Proof. The property stated in the proposition follows directly from the definitions of the two functions. □

The condition described in the previous proposition hardly ever occurs in a practical setting. Especially in view of feature selection, will it hardly ever be the case that classifiers are compared using (different) auxiliary networks of the same log-likelihood. The importance of the proposition therefore lies mainly in the observation that, with a fixed auxiliary network over a fixed set of attributes, the MDL-FS function will always prefer the same classifier as the MDL function. More specifically, the two functions will exhibit the same preference behaviour if the MDL-FS function uses an empty auxiliary network.

We now turn to the feature-selection behaviour of the MDL-FS function in a more practical setting where classifiers are compared using auxiliary networks of possibly different log-likelihood. To informally review some of the function's properties, we begin by studying the MDL-FS score of a Bayesian network classifier $\mathcal{C}$ over $A \cup \{C\}$ and an auxiliary Bayesian network $\mathcal{S}$ over A. We rewrite this score as a sum of terms for the class variable and for each attribute A_i separately:

$$\begin{aligned}\mathit{MDL\text{-}FS}(\mathcal{C}, \mathcal{S} \mid D) = N \cdot \left[H_{\hat{P}}(C) + \frac{\log N}{2 \cdot N} \cdot (|\Omega(C)| - 1) \right] + \\ N \cdot \sum_{A_i \in A} \Big[\quad H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) + \\ \frac{\log N}{2 \cdot N} \cdot (|\Omega(A_i)| - 1) \cdot |\Omega(p_{\mathcal{C}}(A_i))| \Big]\end{aligned}$$

where $p_{\mathcal{C}}(A_i)$ and $p_{\mathcal{S}}(A_i)$ are the sets of parents of A_i in the networks $\mathcal{C}$ and $\mathcal{S}$, respectively. We observe that strong relationships between the attribute A_i and its parents in the classifier $\mathcal{C}$, that is, $H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i))$ going to 0, would decrease the score, while strong relationships between A_i and its parents in $\mathcal{S}$, that is, $H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i))$ going to 0, would increase the score. In view of feature subset selection, therefore, the stronger the relationships of the attribute A_i with its parents in the classifier and the weaker the relationships with its parents in the auxiliary network, the less likely the attribute

is to be removed. To study the differences in strength of these relationships in more detail, we express the difference $H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i))$ in terms of the amounts of irredundancy that the attribute A_i has with its parents in the two networks. Let $p_{\mathcal{C}\cap\mathcal{S}}(A_i) = p_{\mathcal{C}}(A_i) \cap p_{\mathcal{S}}(A_i)$ be the set of parents of A_i in both the classifier and the auxiliary network. Then,

$$H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) = \big(H_{\hat{P}}(A_i \mid p_{\mathcal{C}\cap\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i))\big) - \big(H_{\hat{P}}(A_i \mid p_{\mathcal{C}\cap\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i))\big)$$

The two terms capturing the amount of irredundancy for attribute A_i both are positive. The amount of irredundancy $H_{\hat{P}}(A_i \mid p_{\mathcal{C}\cap\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i))$ describes how "far" the attribute A_i is from being redundant for the set of variables $p_{\mathcal{C}}(A_i) \setminus p_{\mathcal{S}}(A_i)$ given $p_{\mathcal{C}\cap\mathcal{S}}(A_i)$; note that the set $p_{\mathcal{C}}(A_i) \setminus p_{\mathcal{S}}(A_i)$ includes the class variable and all attributes that are parents of A_i in $\mathcal{C}$ but not in $\mathcal{S}$. The closer to 0 this term is, the larger the MDL-FS score will be and the more likely the attribute will be removed upon feature subset selection. The term $H_{\hat{P}}(A_i \mid p_{\mathcal{C}\cap\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i))$, on the other hand, indicates how "far" the attribute A_i is from being redundant for the set of variables $p_{\mathcal{S}}(A_i) \setminus p_{\mathcal{C}}(A_i)$ given $p_{\mathcal{C}\cap\mathcal{S}}(A_i)$); note that the set $p_{\mathcal{S}}(A_i) \setminus p_{\mathcal{C}}(A_i)$ includes all attributes that are parents of A_i in $\mathcal{S}$ but not in $\mathcal{C}$. The closer to 0 this term is, the smaller the MDL-FS score will be and the less likely the attribute is to be removed. We conclude that the difference $H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i))$ represents the amount of irredundancy of the attribute A_i for the class variable and its exclusive parent attributes in the classifier given its parent attributes in the auxiliary network.

We now begin by studying the feature-selection behaviour of the MDL-FS function for complete Bayesian network classifiers using complete auxiliary networks. To pertain to the same joint proposal distributions like the full classifier and auxiliary network, we again extend the selective networks with the deleted attributes by means of nodes without incident arcs.

Proposition 3.4 *Let A be a set of attributes, let C be the class variable, and let D be a dataset of N labeled instances over $A \cup \{C\}$ as before. Let $\mathcal{C}$ be a Bayesian network classifier over the variables $A \cup \{C\}$ and let A_i be an attribute in $\mathcal{C}$ with the set of parents $p_{\mathcal{C}}(A_i)$ and the set of children $c_{\mathcal{C}}(A_i)$. From $\mathcal{C}$, we construct the selective classifier $\mathcal{C}^-$ by deleting the incident arcs of A_i. In addition, let $\mathcal{S}$ be an auxiliary network over the attributes A and let $p_{\mathcal{S}}(A_i)$ be the set of parents and $c_{\mathcal{S}}(A_i)$ be the set of children of A_i in $\mathcal{S}$. Let $\mathcal{S}^-$ be the selective auxiliary network that is obtained from $\mathcal{S}$ by deleting the incident arcs of A_i. Then ,*

$$\textit{MDL-FS}(\mathcal{C}^-, \mathcal{S}^- \mid D) < \textit{MDL-FS}(\mathcal{C}, \mathcal{S} \mid D)$$

if and only if

$$\left[H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - \frac{\log N}{2} \cdot (|\Omega(A_i)| - 1) \cdot \left(|\Omega(C)| \cdot \prod_{A_j \in p_{\mathcal{C}}(A_i)} |\Omega(A_j)| - 1\right)\right] + \sum_{A_k \in c_{\mathcal{C}}(A_i)} \Big[H_{\hat{P}}(A_i \mid p_{\mathcal{C}^-}(A_k)) - H_{\hat{P}}(A_i \mid A_k, p_{\mathcal{C}^-}(A_k)) - \frac{\log N}{2 \cdot N} \cdot (|\Omega(A_k)| - 1) \cdot \prod_{A_j \in p_{\mathcal{C}^-}(A_k) \setminus \{A_i\}} |\Omega(A_j)| \cdot (|\Omega(A_i)| - 1)\Big] < \sum_{A'_k \in c_{\mathcal{S}}(A_i)} [H_{\hat{P}}(A_i \mid p_{\mathcal{S}^-}(A'_k)) - H_{\hat{P}}(A_i \mid A'_k, p_{\mathcal{S}^-}(A'_k))]$$

where $p_{\mathcal{C}^-}(A_k)$, with $p_{\mathcal{C}}(A_k) = \{A_i\} \cup p_{\mathcal{C}^-}(A_k)$, and $p_{\mathcal{S}^-}(A_k)$, with $p_{\mathcal{S}}(A'_k) = \{A_i\} \cup p_{\mathcal{S}^-}(A'_k)$, are the parents sets of A_k and A'_k in $\mathcal{C}^-$ and $\mathcal{S}^-$ graphical structures, respectively.

Proof. To investigate the difference between the two MDL-FS scores, we study the differences of the log-likelihood terms of the classifiers, the log-likelihood terms of the auxiliary networks and of the penalty terms separately. Since we modify only locally the Bayesian structures, the difference in the MDL-FS score will be reflected only by the locally modified parts. The difference of the two log-likelihood terms of the classifiers and of the auxiliary networks equals

$$LL(\mathcal{C} \mid D) - LL(\mathcal{C}^- \mid D) = -N \cdot \left(H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - H_{\hat{P}}(A_i)\right) -$$

$$N \cdot \sum_{A_k \in c_{\mathcal{C}}(A_i)} \left(H_{\hat{P}}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{C}^-}(A_k))\right)$$

$$LL(\mathcal{S} \mid D) - LL(\mathcal{S}^- \mid D) = -N \cdot \left(H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) - H_{\hat{P}}(A_i)\right) -$$

$$N \cdot \sum_{A'_k \in c_{\mathcal{S}}(A_i)} \left(H_{\hat{P}}(A'_k \mid p_{\mathcal{S}}(A'_k)) - H_{\hat{P}}(A'_k \mid p_{\mathcal{S}^-}(A'_k))\right)$$

The difference of the penalty terms of the two MDL-FS scores equals

$$\frac{\log N}{2} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) = \frac{\log N}{2} \cdot (|\Omega(A_i)| - 1) \cdot \left(|\Omega(C)| \cdot \prod_{A_j \in p_{\mathcal{C}}(A_i)} |\Omega(A_j)| \; - \; 1 \right) +$$

$$\sum_{A_k \in c_{\mathcal{C}}(A_i)} \frac{\log N}{2} \cdot (|\Omega(A_k)| - 1) \cdot \prod_{A_j \in p_{\mathcal{C}}(A_k) \setminus \{A_i\}} |\Omega(A_j)| \cdot (|\Omega(A_i)| \; - \; 1)$$

We now have

$$MDL\text{-}FS(\mathcal{C}, \mathcal{S} \mid D) - MDL\text{-}FS(\mathcal{C}^-, \mathcal{S}^- \mid D) =$$

$$\frac{\log N}{2} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) - \left(CLL(\mathcal{C}, \mathcal{S} \mid D) - CLL(\mathcal{C}^-, \mathcal{S}^- \mid D)\right)$$

from which directly results the proposition's inequality by substituting each child $A'_k \in c_{\mathcal{S}}(A_i)$, where $p_{\mathcal{S}}(A'_k) = \{A_i\} \cup p_{\mathcal{S}^-}(A'_k)$, in the equation

$$H_{\hat{P}}(A'_k \mid p_{\mathcal{S}}(A'_k)) - H_{\hat{P}}(A'_k \mid p_{\mathcal{S}^-}(A'_k)) H_{\hat{P}}(A_i \mid A'_k, p_{\mathcal{S}^-}(A'_k)) - H_{\hat{P}}(A_i \mid p_{\mathcal{S}^-}(A'_k)))$$

and each child $A_k \in c_{\mathcal{C}}(A_i)$, where $p_{\mathcal{C}}(A_k) = \{A_i\} \cup p_{\mathcal{C}^-}(A_k)$, in the equation

$$H_{\hat{P}}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{C}^-}(A_k)) H_{\hat{P}}(A_i \mid A_k, p_{\mathcal{C}^-}(A_k)) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}^-}(A_k))$$

those proves are similar with the one from Proposition 3.2. □

By definition we have that the MDL-FS function prefers $\mathcal{C}$ over $\mathcal{C}^-$ if and only if $\mathcal{C}$ has a smaller MDL-FS score than $\mathcal{C}^-$, that is, if and only if $MDL\text{-}FS(\mathcal{C}, \mathcal{S} \mid D) - MDL\text{-}FS(\mathcal{C}^-, \mathcal{S}^- \mid D) < 0$ for the dataset D. Again the difference between the penalty terms is greater than zero because the full Bayesian network classifier is more complex than the selective one. When A_i does not have any children in $\mathcal{C}$, the difference between the two conditional log-likelihoods is equal with the A_i's term in the conditional log-likelihood of the full connected classifier, $N \cdot (H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)))$. When A_i has children (e.g. A_k) in $\mathcal{C}$, we add a term that indicates the amount

of irredundancy of A_i for A_k given A_k's parents in $\mathcal{C}$ except for A_i; when A_i has children (e.g. A'_k) in $\mathcal{S}$, we subtract a term that indicates the amount of irredundancy of A_i for A'_k given A'_k's parents in $\mathcal{S}$ except for A_i. Thus, the difference of conditional log-likelihoods increases with the strength of the relationships between the attribute and the other variables in the classifier, and decreases with its strength of the relationship between the attribute and the other attributes in the auxiliary network.

Informally speaking, the stronger one or more of the relationships between A_i and the other variables $A \setminus \{A_i\}$ in the classifier are and the weaker the relationships of A_i in the auxiliary network are, the more the full classifier is preferred over the selective classifier with the MDL-FS function. It is interesting to note that the feature selection behaviour of the MDL-FS function can be derived directly from the function itself as we have presented in the first part of this section. Thus only the parents sets of an attribute in the classifier and in the auxiliary structure determine the amount of (ir)redundancy an attribute has. The children of an attribute compare the amount of irredundancy the attribute has for them given the other parents of these children in the classifier with the amount of irredundancy the attribute has for their children given the other parents of these children in the auxiliary structure. As a consequence, the feature selection properties in removing or not a redundant attribute with the MDL-FS function is correlated to the complexity of the parents set of the attribute in the classifier and in the auxiliary structure, and not to the attribute's children sets.

From these observations, we conclude that *the MDL-FS function is more suited for the task of feature selection since it can serve to identify and remove redundant attributes at various levels.* Whereas with the MDL function we can eliminate only the attributes that are redundant at level 0 for all other variables from the dataset, $\{C\} \cup A \setminus \{A_i\}$, and thus it can be only used for Naive Bayesian classifiers, the MDL-FS function can be used to eliminate redundant attribute at various levels from more complex classifiers. In the following, we practically illustrate the use of the MDL-FS score in reducing redundant attributes at various levels from Bayesian network classifiers of interest.

In the following paragraph, we show that *the level of redundancy for which the MDL-FS function reduces attributes depends on the complexity of the auxiliary structure*. We consider two cases: (i) a fully connected auxiliary network and an arbitrary complex Bayesian network classifier, and (ii) a fully connected Bayesian network classifier and an arbitrary complex auxiliary network. Since the eventual children of A_i in the classifier indicate how irredundant A_i is for its children rather than for the class variable, in the following discussion, we consider only the parents sets of A_i in a classifier or/and an auxiliary network. When A_i have also children in $\mathcal{C}$ or/and $\mathcal{S}$, we referee to the discussion following Proposition 3.4 for understanding the behaviour of the MDL-FS score.

In the following corollary of Proposition 3.4, we show that the attributes that are redundant for the class variable at level $|A| - 1$ are eliminated from any classifier, where A_i has no children, if the auxiliary network is a fully connected Bayesian network. However, since they might be relevant attributes at the lower levels captured by the correlations in the classifier, their removal might *not* be beneficial for the classification task.

Corollary 3.2 *Consider the fully connected auxiliary network $\mathcal{S}$ over the set of attributes A and the selective auxiliary network $\mathcal{S}^-$ over the set of attributes $A \setminus \{A_i\}$ as described before. Let's consider further a Bayesian network classifier $\mathcal{C}_1$ over $A \cup \{C\}$ where A_i has the parents set $p_{\mathcal{C}_1}(A_i) \subset A \setminus \{A_i\}$ and no children; from the classifier $\mathcal{C}_1$, we construct the selective Bayesian classifier $\mathcal{C}_1^-$ over $\{C\} \cup A \setminus \{A_i\}$ by deleting A_i and its incident arcs. If A_i is redundant for C at level $|A| - 1$ within the allowed noise $\xi(A_i; p_{\mathcal{C}_1}(A_i); N) < \log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$ then the selective classifier $\mathcal{C}_1^-$ is preferred to the full classifier $\mathcal{C}_1$.*

Proof. Because the attribute A_i is connected to all other attributes in a fully connected auxiliary structure $\mathcal{S}$, we can rewrite $\mathcal{S}$ such that A_i has no children without changing the dependencies. Therefore, without loss of generality, we can assume, for ease of exposition, that A_i has

the set of variables $A \setminus \{A_i\}$ for its parents in the graphical structure of $\mathcal{S}$. If A_i is redundant for C given $A \setminus \{A_i\}$ within the allowed noise level $\xi(A_i; p_{\mathcal{C}}(A_i); N) < \log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$, then $H_{\hat{P}}(A_i \mid A \setminus \{A_i\}) - H_{\hat{P}}(A_i \mid C, A \setminus \{A_i\}) \leq \log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$, where $\log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|) = \log N/(2 \cdot N) \cdot (|\Omega(A_i)| - 1) \cdot \left(\prod_{A_j \in A \setminus \{A_i\}} |\Omega(A_j)| - 1\right)$. From Proposition 3.4, we have that the selective classifier is preferred to the full one if and only if $H_{\hat{P}}(A_i \mid A \setminus \{A_i\}) - H_{\hat{P}}(A_i \mid p_{\mathcal{C}_1}(A_i)) < \log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$. Since $p_{\mathcal{C}_1}(A_i) \subseteq \{C\} \cup A \setminus \{A_i\}$, we have $H_{\hat{P}}(A_i \mid C, A \setminus \{A_i\}) \leq H_{\hat{P}}(A_i \mid p_{\mathcal{C}_1})$. From the above two inequalities, we conclude the stated property. □

A direct consequence to the above property is that the redundant attributes at level $|A| - 1$ are effectively removed from a fully connected Bayesian network classifier with the MDL-FS score using a fully connected auxiliary network. Again, without loss of generality, we can assume, for ease of exposition, that A_i has the set of variables $(A \setminus \{A_i\}) \cup \{C\}$ for its parents in the graphical structure of $\mathcal{C}$. In this case, the difference of the two conditional log-likelihood terms is greater than or equal to zero because the amount of uncertainty of an attribute in the fully connected Bayesian network classifier is less or equal to the amount of uncertainty of the same attribute in the fully connected auxiliary network. Thus, the full classifier $\mathcal{C}$ captures the observed conditional probability distribution $\hat{P}(C \mid A)$ at least as accurately as the selective classifier $\mathcal{C}^-$. Informally speaking, the stronger the relationship between the possible eliminated attribute A_i and the class variable C, the more inclined the MDL-FS function is to prefer the full classifier over the selective one. Since the relationship between A_i, the class variable and the other variables in the attribute set is captured with a fully connected Bayesian network classifier, we consider that the redundant attribute A_i at level $|A| - 1$ is correctly eliminated with the MDL-FS score and would be incorrectly kept with the MDL score.

In opposition, for a simpler auxiliary network than the fully connected one, a redundant attribute at level $|A| - 1$ might not be eliminated with MDL-FS from a fully connected Bayesian network classifier where A_i has no parents.

Corollary 3.3 *Consider a fully connected Bayesian network classifier $\mathcal{C}$ over $A \cup \{C\}$ and a selective Bayesian classifier $\mathcal{C}^-$ over $A \setminus \{A_i\} \cup \{C\}$ as before. Let's now consider an auxiliary network $\mathcal{S}_1$ over the set of attributes A, where the attribute A_i has the parent set $p_{\mathcal{S}_1}(A_i) \subset A \setminus \{A_i\}$ and no children; from $\mathcal{S}_1$, we construct the selective auxiliary network $\mathcal{S}_1^-$ over the set of attributes $A \setminus \{A_i\}$ by deleting A_i and its incident arcs. An attribute A_i is eliminated from C if and only if it is redundant for the class variable and the attributes that are not its parents in the auxiliary structure but are parents in the classifier, $H_{\hat{P}}(A_i \mid p_{\mathcal{S}_1}(A_i)) - H_{\hat{P}}(A_i \mid C, A \setminus \{A_i\}) < \log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$.*

Proof. The proof follows directly from Proposition 3.4. □

Informally speaking, the MDL-FS score eliminates an attribute if the redundancy for C given S is an approximation of the redundancy for C at level $|A| - 1$. The above property pertains to the strength of the relationship between A_i and its parents in the classifier as compared to its parents in the auxiliary structure; the stronger the relationship with the class variable C and the remaining attributes $A \setminus \{A_i\}$ as compared with the subset S, the less inclined the MDL-FS score function to eliminate the redundant attribute A_i. We note that the MDL-FS function eliminates only redundant attributes at level 0 for the class variable and for the attributes when an empty network is used for the auxiliary structure.

To illustrate that, in fact, the MDL-FS function, unlike the MDL score, removes attributes redundant at level $|A| - 1$ from full Bayesian network classifiers with a complete Bayesian auxiliary structure, upon feature selection, we consider again the classification problem from Example 2.1.

Example 3.1 *Let's consider the dataset D from Example 2.1 by copying it 128 times - then $N = 4096$ - and $A = \{A_1, \ldots, A_8\}$. Let's consider a complete Bayesian network classifier $\mathcal{C}$ and a selective*

one $\mathcal{C}^-$ as before. Let's consider a complete Bayesian network $\mathcal{S}$ and a selective one $\mathcal{S}^-$ as before. Suppose that $A_i \equiv A_5$, where A_5 is redundant for C at all levels. Since $H_{\hat{P}}(A_5 \mid C, A \setminus \{A_5\}) = H_{\hat{P}}(A_5 \mid A \setminus \{A_5\})$, from the above proposition, the selective Bayesian classifier $\mathcal{C}^-$ is preferred to the full Bayesian classifier $\mathcal{C}$ when we use MDL-FS. Thus, A_5 is correctly removed from the classifier.

When we use MDL, since $A_5 \equiv A_7$ and, then, $H_{\hat{P}}(A_5 \mid C, A \setminus \{A_5\}) = H_{\hat{P}}(A_5 \mid A_7) = 0$, the full Bayesian classifier $\mathcal{C}$ is preferred to the selective Bayesian classifier $\mathcal{C}^-$ because $-H_{\hat{P}}(A_5) = -1 < -\log(4096)/(2 \cdot 4096) \cdot (|\Omega(A_5)| - 1) \cdot \left(2^8 - 1\right) \approx -0.37$. Although A_5 is redundant for C at all levels, we have that A_5 is wrongly kept in the classifier. Furthermore, an arc between A_5 and A_7 will be always considered by any algorithm for constructing Bayesian network classifiers by maximizing the log-likelihood term, because $H_{\hat{P}}(A_5 \mid A_7, S) = 0$, for any subset of attributes $S \subseteq A \setminus \{A_5, A_7\}$; thus, an arc between A_5 and A_7 will represent the most powerful dependence in the Bayesian network classifier. But, since the MDL-FS score uses an auxiliary structure that includes also this arc, MDL-FS eliminates the influence of A_5 from the classifier, whereas the MDL score wrongly keeps it in the classifier. Similar conclusions we draw for A_6 - when the set of parents includes $\{A_1, A_2, A_4\}$ -, A_3 - when the set of parents includes A_2-, and A_8 - when the set of parents include A_2.

We conclude that when the MDL score is employed none of the attributes will be removed from $\mathcal{C}$. When the MDL-FS score is employed, the remaining attributes A_1, A_2 and A_4 are irredundant for the class variable at level $|A| - 1$ and are not removed from the classifier. Then, the fully connected classifier has the same conditional log-likelihood as the optimal classifier and the same number of attributes. We have that the conditional log-likelihood when a complete network is used for the auxiliary network is equal to the conditional entropy $H_{\hat{P}}(C \mid A_1, A_2, A_4) = 0$. □

Now, we show that using the above analysis we can predict the minimum size of the dataset from which a Bayesian network classifier can be constructed using the MDL-FS score. We say that a dataset is too small to construct a specific class of Bayesian network classifiers from, if for any attribute A_i from the dataset, A_i is eliminated from the classifier regardless the strength of the relationship with other attributes. With the MDL-FS score, the size of the dataset depends on the number of values for the attribute under discussion and its parents, and the difference between conditional entropies of an attribute A_i in the fully connected classifier and in the fully connected auxiliary network, $H_{\hat{P}}(A_i \mid C, A \setminus \{A_i\}) - H_{\hat{P}}(A_i \mid A \setminus \{A_i\})$. We note that this difference is lower bounded by $-H_{\hat{P}}(A_i \mid A \setminus \{A_i\}) \geq -H_{\hat{P}}(A_i)$. Thus, if there is an attribute A_i for $-H_{\hat{P}}(A_i) > -\log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$ we say that the dataset is not large enough for constructing the specific type of Bayesian network classifier from it since A_i will be always removed from the full Bayesian network classifier regardless of its relationship with the classifier or with the other attributes.

Note that the size of the dataset needed for the MDL score is the same as for the MDL-FS score. In the above analysis, the difference of the log-likelihood between classifiers has the same lower bound $-H_{\hat{P}}(A_i) \leq H_{\hat{P}}(A_i \mid C, A \setminus \{A_i\}) - H_{\hat{P}}(A_i)$ and the same right term $-\log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$.

Example 3.2 *Let's consider again the dataset from the previous example where the 32 instances from Example 2.1 are copied 128 times and the full Bayesian network classifier $\mathcal{C}$ and the selective one $\mathcal{C}^-$ from the previous example. We recall from Section 2.1 that an attribute A_i's entropy attains its maximum value for a uniform probability distribution over its values. The maximum value thus is found for a distribution with $N(A_i^1) = N(A_i^2)$; the maximum equals 1.00. We obtain the entropy's highest bound for A_1, A_2, A_4, A_5, A_7 and A_8 because they have half instances 1 and half 0. The lowest bound of the entropy for an attribute from Example 2.1, depends of the 32 instances we have listed in Table 2.1 and it is $\log 32/(2 \cdot 32) \cdot \left(2^{7+1} - 1\right) \approx 19.92$. If we assume that the 32 instances include the two possible values of A_i at least once, then the minimum value for the entropy is attained*

for A_6 with $H(A_6) \approx 0.54$. To not eliminate A_6 from a full Bayesian network classifier just because there are not enough data in the dataset, we need a dataset of size $N = 4096 = 32 \cdot 128$. We have that $\log 4096/(2 \cdot 4096) \cdot \left(2^{7+1} - 1\right) \approx 0.37$.

In general, the classifiers which are build in practice, are less complex, with a smaller set of parents for one attribute. For example, if A_6 has one parent or two parents, 32 instances are enough for Example 2.1. Since A_6 is the attribute with the lowest entropy, we can use the dataset of size 32 (or larger) to construct Naive Bayesian or TAN classifiers. □

To conclude the example, we note that for datasets with larger numbers N of instances, we have that the term $\log N/(2 \cdot N)$ approaches 0. The closer this term gets to zero, the less noise the function will allow for a redundant attribute. A larger number of values per variable, on the other hand, would substantially increase the allowed noise $\xi(A_i; C, A \setminus \{A_i\}; N) < \log N/(2 \cdot N) \cdot (|\mathcal{C}| - |\mathcal{C}^-|)$.

3.3 Learning Bayesian network classifiers with MDL-FS

While in the previous section we have investigated general properties of the feature-selection behavior of the two functions, in this section we study their properties in a practical setting. We use the MDL-FS function for constructing Naive Bayesian and TAN classifiers from data using tree structured auxiliary networks of maximum log-likelihood. Within the approach adopted in this thesis, the basic idea is that the search space of alternative classifiers is traversed by a heuristic algorithm that generates in a greedy way various graphical structures, as outlined in Section 2.2. The MDL-FS function is used for comparing the qualities of the classifiers that are supplemented by tree structured auxiliary networks of maximum log-likelihood. As soon as the algorithm cannot construct a new classifier that improves upon the MDL-FS score of the currently best classifier, the algorithm is halted. In the following, we investigate the feature-selection behavior of the MDL-FS function in this context.

In Section 3.3.1 we learn Naive Bayesian classifiers with the MDL function and we show that only redundant attributes at level 0 for the class variable are reduced. In Section 3.3.2 we learn Naive Bayesian classifiers with the MDL-FS function using a tree structured auxiliary network of maximum log-likelihood; and we show that this time attributes redundant at level 0 and 1 are eliminated. Similarly, in Section 3.3.3 we learn TAN classifiers with the MDL function and we show that only redundant attributes at level 0 for the rest of the variables (the class variable and the other attributes) are eliminated. In Section 3.3.4, we show that the MDL-FS function serves to identify and eliminates redundant attributes at level 1 from TANs when we use tree structured auxiliary networks of maximum log-likelihood.

3.3.1 The MDL score for selective Naive Bayesian classifiers

Learning a Naive Bayesian classifier over a given set of attributes is straightforward as the classifier's graphical structure is uniquely defined. At the beginning of the learning process, we compute the conditional entropies for each attribute given the class variable. Such an algorithm has a time complexity of $O(n \cdot N)$.

Learning a selective Naive Bayesian classifier, on the other hand, amounts to selecting a graphical structure from among exponentially many alternatives. We recall that the forward-selection algorithm for this purpose starts with the empty Naive Bayesian classifier and iteratively adds single attributes that upon removal serve to maximally decrease the MDL score of the classifier. The algorithm stops as soon as adding a single attribute can no longer decrease the classifier's score [50, 60]. As we already have stated in Section 3.2.1 in Proposition 3.2, the MDL function tends to eliminate from a Naive Bayesian classifier *only* attributes redundant at level 0. Since a Naive Bayesian classifier

cannot express the information contributed by an attribute at a level higher than level 0, we may look upon an attribute's contribution at level 0 as an *approximation* of its contribution at level $|A| - 1$. Thus, the redundant attributes for the class variable at level 0 are correctly removed from a Naive Bayesian classifier.

3.3.2 The MDL-FS score for selective Naive Bayesian classifiers

This section presents the algorithm we use to construct a selective Naive Bayesian classifier with MDL-FS from the data and we study the feature selection behavior of such classifier. Previously, we argued that to be able to exploit the underlying idea of the MDL-FS function, a more complex auxiliary network than the empty structure needs to be used. The auxiliary network should not have a structure too complex, however, because of the number of instances and the computational effort it requires for its construction. In the following, we show how to learn *tree-structured auxiliary networks* with the MDL-FS function. The use of tree-structured auxiliary networks was motivated by an efficient learning algorithm from Chow and Liu [16], that is guaranteed to result in a tree-structured network of maximum log-likelihood as described in Section 2.1. Chow and Liu's algorithm has a time complexity of $O(n^3 \cdot N)$, which is given by a preprocessing step, where the conditional mutual information between each pair of attributes is computed [30].

The forward-selection algorithm for learning selective Naive Bayesian classifiers starts with the empty Naive Bayesian classifier and auxiliary network. The algorithm iteratively adds single attributes, where in each iteration it computes a Naive Bayesian classifier and a maximum log-likelihood tree over the selected set of attributes. The MDL-FS function is used for selecting the attributes to be added as well as for a stopping criterion: the algorithm stops as soon as adding a single attribute cannot result in a classifier of smaller score.

In the following, we study the behavior of the MDL-FS function when constructing selective Naive Bayesian classifiers from a given dataset using a tree structure auxiliary network of maximum log-likelihood. Proposition 3.4 from Section 3.2.2 does not cover this case; now, after deleting an attribute, we construct the selective tree structure auxiliary network of maximum log-likelihood over the remaining set of attributes. The following propositions pertain to an attribute that is either an internal node or a leaf in the auxiliary tree under consideration. Similar observations also hold for the attribute that constitutes the root of the tree.

Proposition 3.5 *Let $\mathcal{C}$ be a Naive Bayesian classifier over $A \cup \{C\}$. Let $\mathcal{C}^-$ be the selective Naive Bayesian classifier over $(A \setminus \{A_i\}) \cup \{C\}$ that is constructed from $\mathcal{C}$ by deleting A_i and its incident arcs. In addition, let $\mathcal{S}$ be a tree-structured Bayesian network over A and let $A_i \in A$ be an attribute that, in the graphical structure of $\mathcal{S}$, has the attribute $A_j \in A \setminus \{A_i\}$ for its parent. Let $\mathcal{S}^-$ be a tree-structured Bayesian network over $A \setminus \{A_i\}$. Then*

$$\textit{MDL-FS}(\mathcal{C}^-, \mathcal{S}^- \mid D) < \textit{MDL-FS}(\mathcal{C}, \mathcal{S} \mid D)$$

if and only if

$$H_{\hat{P}}(A_i \mid A_j) - H_{\hat{P}}(A_i \mid C) < \frac{\log N}{2 \cdot N} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) + \sum_{A_k \in A \setminus \{A_i\}} \left(H_{\hat{P}}(A_k \mid p_{\mathcal{S}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{S}^-}(A_k))\right)$$

Proof. The difference between the log-likelihood of the classifiers and of the auxiliary structures are

$$LL(\mathcal{C} \mid D) - LL(\mathcal{C}^- \mid D) = -N \cdot \left(H_{\hat{P}}(A_i \mid C) - H_{\hat{P}}(A_i)\right)$$

$$LL(\mathcal{S} \mid D) - LL(\mathcal{S}^- \mid D) = -N \cdot \left(H_{\hat{P}}(A_i \mid A_j) - H_{\hat{P}}(A_i)\right)) \\ - N \cdot \sum_{A_k \in A \setminus \{A_i\}} \left(H_{\hat{P}}(A_k \mid p_{\mathcal{S}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{S}^-}(A_k))\right)$$

Using the same line of reasoning as before, the stated inequality follows directly. □

From the previous proposition, we have that under certain conditions, the MDL-FS function prefers the selective Naive Bayesian classifier over the full Naive Bayesian classifier. To interpret these conditions, from the selective auxiliary network $\mathcal{S}^-$, we now construct another full network $\mathcal{S}'$ by adding the attribute A_i as a child of A_j. We note that the new network $\mathcal{S}'$ also is a tree-structured network; it may not be of maximum log-likelihood, however. Since $\mathcal{S}'$ differs from $\mathcal{S}^-$ in just the parent of A_i, we have that

$$LL(\mathcal{S}^- \mid D) = LL(\mathcal{S}' \mid D) - N \cdot \left(H_{\hat{P}}(A_i) - H_{\hat{P}}(A_i \mid A_j)\right)$$

By replacing the log-likelihood of $\mathcal{S}^-$ with the log-likelihood of $\mathcal{S}'$ in the difference of log-likelihoods $LL(\mathcal{S} \mid D) - LL(\mathcal{S}^- \mid D)$, we obtain

$$LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D) = -N \cdot \sum_{A_k \in A \setminus \{A_i\}} \left(H_{\hat{P}}(A_k \mid p_{\mathcal{S}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{S}^-}(A_k))\right).$$

Based upon these observations, we find that the full classifier $\mathcal{C}$ is preferred over the selective classifier $\mathcal{C}^-$ if and only if

$$H_{\hat{P}}(A_i \mid C) - H_{\hat{P}}(A_i \mid A_j) < \frac{\log N}{2 \cdot N} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) - \frac{LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D)}{N}$$

Whether or not the full classifier is preferred over the selective one, therefore, basically depends on the terms $H_{\hat{P}}(A_i \mid C) - H_{\hat{P}}(A_i \mid A_j)$ and $1/N \cdot (LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D))$. For the latter term we observe that, since $\mathcal{S}$ is a tree-structured network of maximum log-likelihood given D, we have that $LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D) \geq 0$. Now, the full auxiliary network $\mathcal{S}'$ in general may not be of maximum log-likelihood. Because it was constructed from a selective network of maximum log-likelihood, however, it is expected to have a relatively large log-likelihood. Especially for larger values of N, therefore, the impact of the term $1/N \cdot (LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D))$ on the right-hand side of the inequality is expected to be small. The difference in entropy in the left-hand side of the inequality depends on the strength of the relationship of A_i with the class variable C on the one hand, and on the strength of its relationship with the attribute A_j on the other hand. Informally speaking, the stronger the relationship of A_i with A_j and the weaker its relationship with C, the more inclined the MDL-FS function is to remove A_i.

Building upon the above discussion, we now formally prove that, for the class of Naive Bayesian classifiers, the MDL-FS function serves to identify and remove attributes that are redundant for the class variable at level 0, and in addition, with a tree-structured auxiliary network, removes attributes redundant at level 1.

Proposition 3.6 *Let A be a set of attributes and let $A_i \in A$; let C be the class variable as before. Let D be a dataset of N labeled instances over $A \cup \{C\}$. Now, let $\mathcal{C}$ be a Naive Bayesian classifier over $A \cup \{C\}$ and let $\mathcal{C}^-$ be the selective Naive Bayesian classifier over $(A \setminus \{A_i\}) \cup \{C\}$ that is constructed from $\mathcal{C}$ by deleting A_i and its incident arc. Let $\mathcal{S}$ be a tree-structured Bayesian network over A and let A_j be the parent of A_i in the graphical structure of $\mathcal{S}$, and A_i has no children. Let $\mathcal{S}^-$ be a tree-structured Bayesian network over $A \setminus \{A_i\}$.*

If A_i is redundant for C at level 0 and/or 1 within the allowed noise $\xi(A_i; C) < \log N/(2 \cdot N) \cdot (|\Omega(A_i)| - 1) \cdot (|\Omega(C)| - 1)$, then, with the MDL-FS score, the selective Bayesian classifier $\mathcal{C}^-$ is preferred over the full classifier $\mathcal{C}$.

Proof. When the attribute A_i is redundant for the class variable C at level 0 within the allowed noise $\xi(A_i;C)$, we have that $H_{\hat{P}}(A_i) - H_{\hat{P}}(A_i \mid C) < \xi(A_i;C)$. From the properties of the entropy function, moreover, we have that $H_{\hat{P}}(A_i) \geq H_{\hat{P}}(A_i \mid A_j)$. We thus find that the difference $H_{\hat{P}}(A_i \mid C) - H_{\hat{P}}(A_i \mid A_j)$ is positive. We conclude that A_i is removed.

Similarly, assume that the attribute A_i is redundant for C at level 1 given A_j within the allowed noise $\xi(A_i;C)$. We have that $H_{\hat{P}}(A_i \mid A_j) - H_{\hat{P}}(A_i \mid C, A_j) < \xi(A_i;C)$. From $H_{\hat{P}}(A_i \mid C) \geq H_{\hat{P}}(A_i \mid A_j, C)$, we now find that the difference $H_{\hat{P}}(A_i \mid C) - H_{\hat{P}}(A_i \mid A_j)$ is positive. Thus, also in this case, A_i is removed. □

From the previous proposition, we have that, using a tree-structured auxiliary network of maximum log-likelihood, the MDL-FS function will always identify and remove from a Naive Bayesian classifier any attribute that is redundant for the class variable at level 0 and/or level 1 within an allowed noise level determined by the penalty term. Even though a Naive Bayesian classifier can only capture the information contributed by an attribute to the class variable at level 0, in practice, it is expected that the function will identify many of the attributes that are redundant at level 1, since it is quite likely that an attribute A_i that is redundant for the class variable given A_j, will be connected to A_j in the auxiliary network. More specifically, the function identifies the attributes that are redundant for the class variable given their parent in the auxiliary network because it models the most important relationships among the attributes, from which it can identify redundancy at level 1 to at least some extent. For example, the MDL-FS function can identify attributes that are copies of one another. We recall from Section 2.2 that the inclusion of perfectly correlated attributes may bias the accuracy of the classifier and should in fact be removed.

We observe that the MDL-FS function, when using a tree structured auxiliary structure, tends not to eliminate any other attributes from a Naive Bayesian classifier than the ones reviewed above. To investigate whether or not such attributes should indeed be removed, we consider the following two types of attribute: (i) A_i is irredundant at level 0, A_i is irredundant at level 1, redundant at level 2 and irredundant at level $|A| - 1$; and (ii) A_i is irredundant at level 0, A_i is irredundant at level 1, redundant at level 2, and redundant at level $|A| - 1$. As we mentioned before, in situation (ii), attribute A_i should be removed and in situation (i), attribute A_i should in essence contribute to the classification at level $|A| - 1$. Even though a Naive Bayesian classifier can only express the information contributed by an attribute at level 0 for the class variable, a tree structured Bayesian network over A as auxiliary structure can express the attribute's dependency at level 0 with other attributes from A. We now may look upon the attribute's contribution in the auxiliary structure as an *approximation* of its contribution at level 1; furthermore we may look upon this as an *approximation* of its contribution at level $|A| - 1$. Then, the attribute A_i should not be removed from the Naive Bayesian classifier. We note that, for identifying redundancy at a higher level, an auxiliary network of higher complexity is required. To conclude, we illustrate the basic idea by means of an example.

Example 3.3 *We consider again the classification problem from Example 2.1. We note $A = \{A_1, \ldots, A_8\}$. Let's consider a Naive Bayesian classifier $\mathcal{C}$ and a selective Naive Bayesian classifier $\mathcal{C}^-$ as before. Associated to these classifiers are a tree-structured Bayesian network $\mathcal{S}$ and a selective tree-structured Bayesian network $\mathcal{S}^-$ as before.*

We have that A_5 and A_7, where $A_5 \equiv A_7$, are redundant for the class variable C at level 0 and higher. From Proposition 3.2, with the MDL score, A_5 is removed from $\mathcal{C}$. Also according with the previous proposition, with the MDL-FS score, the selective classifier $\mathcal{C}^-$ is preferred over the full classifier $\mathcal{C}$, and A_5 is effectively removed. We note that A_5 is removed regardless of the strengths of its relationships with the other attributes. A similar observation holds for the attribute $A_7 \equiv A_5$.

We further recall from Example 2.1 that the attributes A_1 and A_4 are redundant for the class variable at level 0 but irredundant at higher levels than 1. A full Naive Bayesian classifier over $A \cup \{C\}$

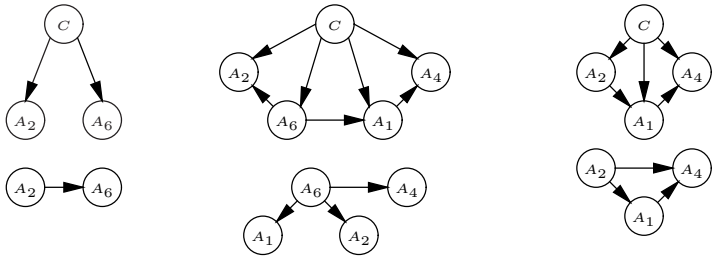

Figure 3.1: The selective Naive Bayesian classifier and its associated tree structured auxiliary network (left), the selective TAN classifier and its tree structured auxiliary network (middle) and the selective TAN classifier and its complete auxiliary network (right), constructed with the MDL-FS function.

only includes the prior probability distribution $P(C)$ and the conditional probability distributions $P(A_1 \mid C)$ and $P(A_4 \mid C)$. The XOR operator that captures the combined influence of A_1 and A_4 on C cannot be modeled by a Naive Bayesian classifier. Although these attributes cannot bias the classification as it does not contribute any information to the class variable C at level 0, it adds to the complexity of the classifier and therefore they are correctly removed by the MDL and MDL-FS functions.

Suppose that $A_i \equiv A_8$, which is redundant for C given A_2 and irredundant at level 0. Then $A_j \equiv A_2$ since $A_8 \equiv A_2$ and then the mutual information of A_8 and A_2 is maximal. The MDL-FS function prefers the selective classifier $\mathcal{C}^-$ over the selective classifier $\mathcal{C}$; the attribute A_i is thus removed from the classifier. Unlike the MDL-FS score, the MDL function prefers $\mathcal{C}$ over $\mathcal{C}^-$ whenever $H_{\hat{P}}(A_2) - H_{\hat{P}}(A_2 \mid C) > \xi(A_2; C)$. For $N = 32$, for example, we find that $\xi(A_2; C) < \log 32/(2 \cdot 32) \cdot (|\Omega(A_2)| - 1) \cdot (|\Omega(C)| - 1) \approx 0.08$. We further have that $H_{\hat{P}}(A_2) - H_{\hat{P}}(A_2 \mid C) \approx 0.16$. We conclude that the full classifier $\mathcal{C}$ therefore is always preferred over the selective classifier $\mathcal{C}^-$ and the attribute A_2 is not removed. Similar observations hold for the attribute A_3 that is strongly connected to A_2.

With MDL-FS, after eliminating the redundant attributes at level 0 and 1, there are only two left: A_2 and A_6. Suppose that $A_i \equiv A_2$, where A_2 is irredundant for C at level 0 and higher, and $A_j \equiv A_6$. The MDL-FS function now prefers the full classifier $\mathcal{C}$ over the selective classifier $\mathcal{C}^-$ because, for $N = 32$, for example, we have that $H_{\hat{P}}(A_2 \mid A_6) - H_{\hat{P}}(A_2 \mid C) = 0.87 - 0.69 > \xi(A_2; C) = 0.08$. The attribute A_2 is thus not removed from the classifier. Similar observations hold for the attribute A_6 that is irredundant for C at level 0 up to 3.

The selective Naive Bayesian classifier yielded for the example dataset by the MDL-FS function is shown in Figure 3.1 on the left. The conditional log-likelihood of these Naive Bayesian classifier is proportional with $CLL(\mathcal{C}, \mathcal{S} \mid D)/N \approx 0.47$. This score it higher than the conditional log-likelihood of the selective Naive Bayesian classifier selected by the MDL score -0.02. With a tree-structured auxiliary network of maximum log-likelihood, therefore, the MDL-FS function serves to remove attributes that are redundant at level 0 and/or at level 1 upon feature selection. We observe that the conditional log-likelihood of the selective Naive Bayesian selected by the MDL-FS score is higher than the conditional entropy of the class variable given the attribute set A; the conditional log-likelihood of the Naive Bayesian classifiers when considering a tree structured auxiliary network of maximum log-likelihood is positive when the auxiliary network has a higher score than the Naive Bayesian classifiers. □

We would like to note that, in practical applications, generally good feature-selection results are obtained with the MDL function for Naive Bayesian classifiers [50]. Apparently, the function's

ability to identify and remove attributes that are redundant for the class variable at level 0 suffices to yield relatively simple classifiers of good accuracy. However, the MDL-FS score is reducing even more redundant attributes - that are attributes redundant at level 1 - since the contribution of these attributes are captured by the tree structured auxiliary network. Thus, we consider that MDL-FS is more suited for the task of constructing selective Naive classifier from data with the reduction of redundant attributes at chosen levels.

3.3.3 The MDL score for selective TAN classifiers

Learning a TAN classifier over a given set of attributes is more involved than learning a Naive Bayesian classifier, because the graphical structure of the TAN classifier is not unique. A well-known search algorithm for learning TAN classifiers[30] starts with a Naive Bayesian classifier and iteratively inserts undirected edges between pairs of attributes, under the constraint of acyclicity; the selection of the edges to be inserted is based upon the conditional mutual information of two attributes given the class variable. The algorithm stops adding edges as soon as the undirected graphical structure over the attributes constitutes a tree. After randomly selecting a root for the tree, the edges in the structure are oriented from the root towards the leaves. The resulting TAN classifier is guaranteed to have *maximum log-likelihood* given the data. In the sequel, we assume a TAN classifier to be of maximum log-likelihood unless explicitly stated otherwise. The time complexity of this algorithm is $O(n^3 \cdot N)$ and is given by the preprocessing step, where conditional mutual informations between each pair of attributes are computed [30].

The forward-selection algorithm for constructing a selective TAN classifier now starts with the empty TAN classifier and iteratively adds single attributes. In each iteration, it computes a TAN classifier over the selected set of attributes by means of the algorithm described above. The MDL function again is used for selecting the attributes to be added as well as for a stopping criterion: the algorithm stops as soon as adding a single attribute cannot result in a TAN classifier of higher score.

In contrast with Naive Bayesian classifiers, TAN classifiers can express information at level 1: they can model the relationship of an attribute with the class variable conditional on a single other attribute. Although similar, the following property is not a direct consequence of Proposition 3.2; when deleting an attribute from a TAN classifier, now, the selective classifier is also a TAN classifier of maximum log-likelihood which might be different from the selective classifier obtained from the full TAN by deleting the given attribute and its incident arcs. This proposition and the following up discussion pertain to an attribute that is either an internal node or a leaf in the attribute tree of the TAN classifier. Similar observations also holds for the attribute that constitutes the root of the tree.

Proposition 3.7 *Let $\mathcal{C}$ be a TAN classifier over $A \cup \{C\}$ and let $A_i \in A$ be an attribute that, in the graphical structure of $\mathcal{C}$ has the set of variables $p_{\mathcal{C}}(A_i) = \{A_j, C\}$, for some $A_j \in A \setminus \{A_i\}$, for its parents. Let $\mathcal{C}^-$ be a selective TAN classifier over $(A \setminus \{A_i\}) \cup \{C\}$. Then*

$$\mathit{MDL}(\mathcal{C} \mid D) < \mathit{MDL}(\mathcal{C}^- \mid D)$$

if and only if

$$H_{\hat{P}}(A_i \mid C, A_j) - H_{\hat{P}}(A_i) < -\frac{\log N}{2 \cdot N} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) - \sum_{A_k \in A \setminus \{A_i\}} \left(H_{\hat{P}}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{C}^-}(A_k))\right)$$

Proof. The proof follows the same line of reasoning as in Proposition 3.2. □

Thus, under certain conditions, the MDL function prefers the selective TAN classifier over the full TAN classifier, and, therefore, it removes an attribute under consideration. To interpret these conditions, from the selective classifier $\mathcal{C}^-$, we now construct another full classifier $\mathcal{C}'$ by adding the attribute A_i as a child of A_j and C. We observe that the new classifier $\mathcal{C}'$ is also a TAN classifier yet not necessarily of maximum log-likelihood. Since $\mathcal{C}'$ differs from $\mathcal{C}^-$ in just the set of parents of the attribute A_i, we have that

$$LL(\mathcal{C}^- \mid D) = LL(\mathcal{C}' \mid D) - N \cdot \big(H_{\hat{P}}(A_i) - H_{\hat{P}}(A_i \mid A_j, C)\big)$$

Based upon this observation, we find that the full classifier $\mathcal{C}$ is preferred over the selective classifier $\mathcal{C}^-$ if and only if

$$H_{\hat{P}}(A_i \mid A_j, C) - H_{\hat{P}}(A_i \mid A'_j) < -\frac{\log N}{2 \cdot N}(|\mathcal{C}| - |\mathcal{C}^-|) + \frac{LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D)}{N}$$

Whether or not the full classifier is preferred over the selective one, therefore, depends basically on the terms $H_{\hat{P}}(A_i \mid A_j, C) - H_{\hat{P}}(A_i \mid A'_j)$ and $\frac{1}{N} \cdot (LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D))$. For the latter term, we observe that, since $\mathcal{C}$ is a classifier of maximum log-likelihood given D, we have that $LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D) \geq 0$. Now, the full classifier $\mathcal{C}'$ in general may not be of maximum log-likelihood. Because it was constructed from a selective classifier of maximum log-likelihood, however, it is expected to have a relatively large log-likelihood given the data. Especially for larger values of N, therefore, the impact of the term $\frac{1}{N} \cdot (LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D))$ on the right-hand side of the inequality is expected to be small. For the left-hand side of the inequality, we have that $H_{\hat{P}}(A_i \mid A_j, C) - H_{\hat{P}}(A_i \mid A'_j) \leq 0$. The difference in entropy depends on the strength of the relationship of A_i with its parents in the classifier. Informally speaking, the weaker the relationship of A_i with its parents in the classifier, the more inclined the MDL-FS function is to remove A_i.

Building upon the same argument as before, therefore, we conclude that any attribute that is redundant for the class variable at level 1, should be removed. Unless the attribute is quite weakly related to its neighboring attributes in the classifier, according with the above proposition, however, the MDL function tends not to remove it. We conclude, as a consequence, that generally poor feature-selection results are obtained with the MDL function upon constructing TAN classifiers.

3.3.4 The MDL-FS score for selective TAN classifiers

In this section, we show how to learn selective TAN classifiers of maximum log-likelihood with the MDL-FS score by using a tree structured auxiliary network also of maximum log-likelihood. Learning a TAN classifier over a given set of variables with the MDL-FS function amounts to constructing both a classifier and an auxiliary network from the available data. Upon learning a selective TAN classifier with the MDL-FS score, moreover, learning the two networks is performed iteratively. In this section, we focus again on the use of tree-structured auxiliary networks with the MDL-FS function. In addition to the use of tree-structured networks of maximum log-likelihood as in the previous section, we also study the use of the attribute tree of the constructed TAN classifier for its associated auxiliary network. Note that using the attribute tree of a TAN classifier with the MDL-FS function serves to substantially reduce the computational effort involved in the learning task.

In the following, we investigate the ability of the MDL-FS function to identify and remove, from a TAN classifier, redundant attributes at different levels. We study the use of maximum log-likelihood tree-structured auxiliary networks for this purpose and establish the condition under which the MDL-FS function with such a network removes an attribute from a classifier. Although similar, the following property is not a direct consequence of Proposition 3.4; when deleting an attribute from a TAN

classifier, now, the selective classifier is also a TAN classifier of maximum log-likelihood which might be different from the selective classifier obtained from the full TAN by deleting the given attribute and its incident arcs. Similar observations hold also for the selective tree structure auxiliary network. The following observations pertains to an attribute that is either an internal node or a leaf in the attribute tree of the TAN classifier and in the auxiliary tree under consideration. Similar observations also hold for the attribute that constitutes the root of the tree.

Proposition 3.8 *Now, let $\mathcal{C}$ be a TAN classifier over $A \cup \{C\}$ and let $A_i \in A$ be an attribute that, in the graphical structure of $\mathcal{C}$ has the set of variables $p_{\mathcal{C}}(A_i) = \{A_j, C\}$, for some $A_j \in A \setminus \{A_i\}$, for its parents. Let $\mathcal{C}^-$ be a selective TAN classifier over $(A \setminus \{A_i\}) \cup \{C\}$. In addition, let $\mathcal{S}$ be a tree-structured Bayesian network over A and let $A'_j \in A$ be the parent of A_i in the graphical structure of $\mathcal{S}$ and let $\mathcal{S}^-$ be a tree-structured Bayesian network over $A \setminus \{A_i\}$. Then*

$$\textit{MDL-FS}(\mathcal{C}, \mathcal{S} \mid D) < \textit{MDL-FS}(\mathcal{C}^-, \mathcal{S}^- \mid D)$$

if and only if

$$H_{\hat{P}}(A_i \mid C, A_j) - H_{\hat{P}}(A_i \mid A'_j) < -\frac{\log N}{2 \cdot N} \cdot (|\mathcal{C}| - |\mathcal{C}^-|) -$$
$$\sum_{A_k \in A \setminus \{A_i\}} \big(H_{\hat{P}}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}}(A_k \mid p_{\mathcal{S}}(A_k)) -$$
$$H_{\hat{P}}(A_k \mid p_{\mathcal{C}^-}(A_k)) + H_{\hat{P}}(A_k \mid p_{\mathcal{S}^-}(A_k))\big)$$

Proof. The difference between the two log-likelihoods of the classifiers and of the auxiliary structures are

$$LL(\mathcal{C} \mid D) - LL(\mathcal{C}^- \mid D) = -N \cdot (H_{\hat{P}_D}(A_i \mid C, A_j) - H_{\hat{P}_D}(A_i))$$
$$-N \cdot \sum_{A_k \in A \setminus \{A_i\}} (H_{\hat{P}_D}(A_k \mid p_{\mathcal{C}}(A_k)) - H_{\hat{P}_D}(A_k \mid p_{\mathcal{C}^-}(A_k)))$$
$$LL(\mathcal{S} \mid D) - LL(\mathcal{S}^- \mid D) = -N \cdot (H_{\hat{P}_D}(A_i \mid A_j\prime) - H_{\hat{P}_D}(A_i))$$
$$-N \cdot \sum_{A_k \in A \setminus \{A_i\}} (H_{\hat{P}_D}(A_k \mid p_{\mathcal{S}}(A_k)) - H_{\hat{P}_D}(A_k \mid p_{\mathcal{S}^-}(A_k)))$$

Following the same line of reasoning as before, the proof is straightforward. □

Thus, under certain conditions, the MDL-FS function prefers the selective TAN classifier over the full TAN classifier, and, therefore, it removes an attribute under consideration. To interpret these conditions, from the selective classifier $\mathcal{C}^-$, we now construct another full classifier $\mathcal{C}'$ by adding the attribute A_i as a child of A_j and C. Similarly, from the selective auxiliary network $\mathcal{S}^-$, we now construct another full network $\mathcal{S}'$ by adding the attribute A_i as a child of A_j. Following the same line of reasoning as in Section 3.3.2 and 3.3.4, we find that the full classifier $\mathcal{C}$ is preferred over the selective classifier $\mathcal{C}^-$ if and only if

$$H_{\hat{P}}(A_i \mid A_j, C) - H_{\hat{P}}(A_i \mid A'_j) < -\frac{\log N}{2 \cdot N}(|\mathcal{C}| - |\mathcal{C}^-|)$$
$$+ \frac{LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D)}{N} - \frac{LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D)}{N}$$

Whether or not the full classifier is preferred over the selective one, therefore, depends basically on the terms $H_{\hat{P}}(A_i \mid A_j, C) - H_{\hat{P}}(A_i \mid A'_j)$ and $1/N \cdot (LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D)) - 1/N \cdot (LL(\mathcal{S} \mid D) -$

$LL(\mathcal{S}' \mid D)$). For the latter term, we observe that, since $\mathcal{C}$ is a classifier of maximum log-likelihood given D, we have that $LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D) \geq 0$. Now, the full classifier $\mathcal{C}'$ in general may not be of maximum log-likelihood. Because it was constructed from a selective classifier of maximum log-likelihood, however, it is expected to have a relatively large log-likelihood given the data. Similarly, since $\mathcal{S}$ is a tree of maximum log-likelihood given D, we have that $LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D) \geq 0$ and $\mathcal{S}'$ is expected to have a relatively large log-likelihood given the data. Especially for larger values of N, therefore, the impact of the term $1/N \cdot (LL(\mathcal{C} \mid D) - LL(\mathcal{C}' \mid D)) - 1/N \cdot (LL(\mathcal{S} \mid D) - LL(\mathcal{S}' \mid D))$ on the right-hand side of the inequality is expected to be small. For the left-hand side of the inequality, we have that $H_{\hat{P}}(A_i \mid A_j, C) - H_{\hat{P}}(A_i \mid A'_j) \leq 0$. The difference in entropy depends on the strength of the relationship of A_i with its parents in the classifier on the one hand and with its parent in the auxiliary network on the other hand. Informally speaking, the stronger the relationship of A_i with its parent in the auxiliary network and the weaker the relationship of A_i with its parents in the classifier, the more inclined the MDL-FS function is to remove A_i.

We illustrate the basic idea by means of an example.

Example 3.4 *Again, we consider the classification problem from Example 2.1. We note* $A = \{A_1, \ldots, A_8\}$. *Let's consider a full TAN classifier* $\mathcal{C}$ *over* $A \cup \{C\}$, *its associated tree structured auxiliary structure* $\mathcal{S}$, *and a selective TAN classifier* $\mathcal{C}^-$ *over* $(A \setminus \{A_i\}) \cup \{C\}$ *and its associated tree structure auxiliary network* $\mathcal{S}^-$ *as before. We now compare the MDL-FS score of* $\mathcal{C}$ *and* $\mathcal{S}$, *with the MDL-FS score of* $\mathcal{C}^-$ *and* $\mathcal{S}^-$.

Since the conditional mutual information of A_5 *and* A_7 *given* C *is maximal and both variables do not have strong relationships with other attributes, any full TAN classifier of maximum log-likelihood will include an edge between the two attributes. Similar observations hold for the auxiliary network. Since* $A_5 \equiv A_7$, A_7 *perfectly replaces* A_5 *in the classifier; we assume, without loss of generality, that* A_5 *does not have any children. From the equivalence of the two attributes, we now observe that* $H_{\hat{P}}(A_5 \mid A_7, C) = 0$. *We find that any TAN classifier* $\mathcal{C}$ *that contains both* A_5 *and* A_7 *has a lower MDL score than the selective classifiers that are constructed from* $\mathcal{C}$ *by removing* A_5 *or* A_7. *Recall that* A_5 *and* A_7 *are redundant for* C *at all levels. However, any TAN classifier* $\mathcal{C}$ *that contains both* A_5 *and* A_7 *has a higher MDL-FS score than the selective TAN classifiers that are constructed from* $\mathcal{C}$ *by removing* A_5 *or* A_7. *These two attributes will therefore be correctly removed. Similar observations hold for the attributes* A_3, A_7 *and* A_8 *which are redundant for* C *at level 1 and higher.*

The attributes A_1, A_2, A_4 *and* A_6 *again are identified as being irredundant at level* 1 *and therefore are correctly kept in the classifier by both scoring functions.*

With a tree-structured auxiliary network of maximum log-likelihood, therefore, the MDL-FS function serves to remove from TAN classifiers attributes that are redundant at level 1 *upon feature selection. A selective TAN classifier yielded for the example dataset by the MDL-FS function is shown in Figure 3.1 in the middle. We observe that the conditional log-likelihood of this classifier is proportional with* $CLL(\mathcal{C}, \mathcal{S} \mid D)/N = 0.75$. *This score is lower than the conditional entropy of the selective Naive Bayesian classifier obtained the MDL score* 0.84 *but higher than the conditional entropy of the class variable given the minimum set of irredundant attributes* $\{A_1, A_2, A_4\}$ *that is* 0. □

We observe that the MDL-FS function, when using a tree structured auxiliary structure, tends to eliminate attributes redundant at level 1 within the allowed noise modeled by the penalty term, $\log N/(2 \cdot N)(|\mathcal{C}| - |\mathcal{C}^-|)$, from a TAN classifier. Since we look upon the attribute's contribution at level 1 as an approximation of its contribution at level $|A| - 1$, then A_i is correctly removed from the classifier. To the attributes that are redundant for the class variable at a level higher than level 1, a similar analysis applies as the previous one. We note that, for identifying redundancy at a higher level, an auxiliary network of higher complexity is required.

The attributes that are redundant for the class variable at a level higher than level 1 tends to not be removed from the classifier. To study if such an attribute should indeed be removed, we now consider an attribute A_i that is redundant for the class variable C at level 0 yet irredundant at level 1. If it is redundant at the highest level $|A| - 1$, then the attribute should not contribute to the classification. In fact, it may bias the classification by its contribution to the class variable at level 1 if it is not removed. We note, however, that the MDL-FS function tends not to remove the attribute. If, on the other hand, the attribute A_i is irredundant at level $|A| - 1$, then it should in essence contribute to the classification at this level. We recall, however, that a TAN classifier can only express the information contributed by an attribute at levels 0 and 1. If we may look upon the attribute's contribution at levels 0 and 1 as an approximation of its contribution at level $|A| - 1$, then A_i should not be removed from the classifier. The MDL-FS function indeed tends to not remove it.

We argued before that learning a TAN classifier over a given set of attributes with the MDL-FS function can be computationally demanding if the function uses a maximum log-likelihood tree-structured auxiliary network, since basically two tree-structured networks need to be learned from the data. To reduce the computational effort involved in the learning task, instead of a separately learned maximum log-likelihood tree-structured network, the attribute tree of the TAN classifier can be used as an auxiliary network. We recall that this attribute tree is obtained from the TAN classifier by deleting the class variable and its incident arcs. When the attribute tree of the TAN classifier under study is used with the MDL-FS function, a property similar to the property described in the previous lemma holds. Since the differences in entropy will be smaller with the attribute tree than with the maximum log-likelihood tree-structured network, the MDL-FS function will the less inclined to prefer the selective classifier.

3.4 An MDL score for learning Bayesian network classifiers from data

In the previous section we have analyzed an MDL score for feature selection in the context of Bayesian network classifiers. In the following, we propose a slight modification of the MDL-FS score - it also includes a penalty term for the auxiliary structure - for learning the structure of Bayesian network classifiers from data. Such an MDL score is the interplay between the (approximation) of the conditional log-likelihood of the class variable given a set of attributes and a corresponding penalty term which captures the length of the encoding of the (approximation of the) conditional log-likelihood.

Let's now consider the MDL-FS score from Definition 3.1 to which we add a term for the penalty term of the auxiliary structure. Let $\mathcal{C}$ be a Bayesian network classifier over $A \cup \{C\}$ and let $\mathcal{S}$ be a Bayesian network over A as before. Then,

$$\textit{MDL-C}(\mathcal{C}, \mathcal{S} \mid D) = \frac{\log N}{2} \cdot (|\mathcal{C}| - |\mathcal{S}|) - \textit{CLL}(\mathcal{C}, \mathcal{S} \mid D)$$

where $\textit{CLL}(\mathcal{C}, \mathcal{S} \mid D) = \textit{LL}(\mathcal{C} \mid D) - \textit{LL}(\mathcal{S} \mid D)$; $|\mathcal{C}|$ and $|\mathcal{S}|$ the penalty terms of $\mathcal{C}$ and $\mathcal{S}$, respectively; $\textit{LL}(\mathcal{C} \mid D)$ and $\textit{LL}(\mathcal{S} \mid D)$ are the log-likelihood terms of $\mathcal{C}$ and $\mathcal{S}$, respectively.

The MDL-C score of a given Bayesian network classifier and the associated auxiliary structure serves to indicate the classifier's quality with respect to the data under study. The length of the description of the class variable given the data equals the length of the encoding of the observed conditional probability distribution $\hat{P}(C \mid A)$; since it does not factorise over the graphical structure of $\mathcal{C}$ we approximate it again with the conditional log-likelihood term $\textit{CLL}(\mathcal{C}, \mathcal{S} \mid D)$. In Section 3.1, we have showed that $\textit{CLL}(\mathcal{C}, \mathcal{S} \mid D)$, when $\mathcal{C}$ and $\mathcal{S}$ are complete Bayesian networks, captures the

length of the encoding of $\hat{P}(C \mid A)$ and therefore is related to the classifier's ability to explain the class variable given the data. The smaller the score, the better the classifier is. Although a fully connected network perfectly matches the data, it will generally show poor performance, as a result of overfitting. If $\mathcal{C}$ and $\mathcal{S}$ are simpler Bayesian networks than the complete networks, to approximate the conditional probability distribution with the conditional log-likelihood term, we maximize the log-likelihood of both the classifier and the auxiliary networks of a given family. For a discussion about the behavior of the conditional log-likelihood term we refer to Section 3.1.2.

The penalty term $\log N/2 \cdot (|\mathcal{C}| - |\mathcal{S}|)$ captures the length of the encoding of $CLL(\mathcal{C}, \mathcal{S} \mid D)$, unlike the penalty term of the MDL-FS score, $\frac{\log N}{2} \cdot |\mathcal{C}|$, that captures the length of the encoding of $LL(\mathcal{C} \mid D)$. It is therefore related to the complexity of the conditional log-likelihood: from the encoding of the Bayesian network classifier we subtract the encoding of the auxiliary network. Like the conditional log-likelihood term, the penalty term attains its maximum for a fully connected Bayesian network classifier and the empty auxiliary network. It attains its minimum for the empty classifier and a fully connected auxiliary network. Recall that, in a practical setting, we maximize the log-likelihood of both the classifier and the auxiliary networks of a given family. The penalty term now counterbalances the effect of the conditional log-likelihood term within the MDL-C function since it increases in value as the Bayesian network classifier is more densely connected as compared with the auxiliary network.

Let's now investigate the feature selection behavior of the MDL-C function. The MDL-C score is a sum over terms that compares the strength and the complexity of the relationships between an attribute and its parents in the classifier and of the relationships of the same attribute and its parents in the auxiliary network. Let $\mathcal{C}$ be a Bayesian network classifier over $A \cup \{C\}$ and let $\mathcal{S}$ be a Bayesian network over A. We can rewrite the MDL-C score as a sum of terms for the class variable and for each attribute A_i.

$$\begin{aligned} MDL\text{-}C(\mathcal{C}, \mathcal{S} \mid D) = N \cdot \left(H_{\hat{P}}(C) + \frac{\log N}{2 \cdot N} \cdot (|\Omega(C)| - 1) \right) \\ + N \cdot \sum_{A_i \in A} \left(H_{\hat{P}}(A_i \mid p_{\mathcal{C}}(A_i)) - H_{\hat{P}}(A_i \mid p_{\mathcal{S}}(A_i)) \right) \\ + N \cdot \sum_{A_i \in A} \frac{\log N}{2 \cdot N} \cdot (|\Omega(A_i)| - 1) \cdot (|\Omega(p_{\mathcal{C}}(A_i))| - |\Omega(p_{\mathcal{S}}(A_i))|) \end{aligned}$$

where, for each attribute $A_i \in A$, the sets $p_{\mathcal{C}}(A_i)$ and $p_{\mathcal{S}}(A_i)$ are the set of parents of A_i in the graphical structure of the networks $\mathcal{C}$ and $\mathcal{S}$, respectively. We observe the similarity between the above formula and the first formula from Section 3.2.2 which describes the MDL-FS score. The MDL-C score reduces redundant attributes for the class variable at the level dictated by the auxiliary structure within an allowed noise that, this time, models the difference between the complexity of the classifier and the complexity of the auxiliary network. Unlike for the MDL-FS score, for the MDL-C score the complexity of the auxiliary structure should be similar with the complexity of the Bayesian network classifier; otherwise, the penalty term would be negative and its purpose the punish too complex classifiers is lost.

Furthermore, the MDL-C function can be rewritten as the difference between two MDL scores: from the MDL score for the classifier $\mathcal{C}$ we subtract the MDL score of the auxiliary network $\mathcal{S}$. We have

$$MDL\text{-}C(\mathcal{C}, \mathcal{S} \mid D) = MDL(\mathcal{C} \mid D) - MDL(\mathcal{S} \mid D)$$

Then, a property similar with the one from Proposition 3.3 holds also for MDL-C. Let $\mathcal{C}$ and $\mathcal{C}'$ be two Bayesian network classifiers over $A \cup \{C\}$. Let $\mathcal{S}$ and $\mathcal{S}'$ be two Bayesian networks over A. If

$MDL(\mathcal{S} \mid D) = MDL(\mathcal{S}' \mid D)$, then

$$MDL(\mathcal{C} \mid D) - MDL(\mathcal{C}' \mid D) = MDL\text{-}C(\mathcal{C}, \mathcal{S} \mid D) - MDL\text{-}C(\mathcal{C}', \mathcal{S}' \mid D)$$

Informally speaking, for identical auxiliary structures, the MDL-C score behaves as with the MDL score, learning Bayesian network classifiers that maximize the log-likelihood of the joint probability distribution. This situation occurs only after a set of irredundant attributes for the class variable is selected.

For example, a greedy algorithm for learning Bayesian network classifiers from data first eliminates redundant attributes for the classification task and, in the same time, constructs good classifiers. Let's consider a forward selection algorithm that starts with the empty Bayesian network classifier and auxiliary structure. It iteratively adds single attributes, where in each iteration it computes (or approximates) a maximum log-likelihood Bayesian network classifier and auxiliary network of a given family over the selected set of attributes. The MDL-C function is used for selecting the attributes to be added as well as for a stopping criterion: the feature selection algorithm stops as soon as adding a single attribute cannot result in a classifier of higher score. The output of such an algorithm is a Bayesian network classifier of (approximative) maximum log-likelihood over an irredundant set of attributes for the classification.

We conclude that the MDL-C score is less suited for feature selection than the MDL-FS score since it is less selective, but the Bayesian network classifiers learned with MDL-C are better than the classifiers learned with MDL-FS since the penalty of MDL-C now encodes the complexity of the conditional log-likelihood term. Our theoretical findings are supported by the experimental results in Section 3.6.

3.5 Related work

In the previous sections we have proposed an MDL like function score for learning Bayesian network classifiers from data. In this section, we review the related work on practical algorithms for feature selection, and on learning Bayesian network classifiers from data using MDL-like functions.

After the crisp theoretical definitions of useful and not-useful attributes for the class variable given a set of attributes based on the associated conditional probability, for practical use, Tsamardinos and Aliferis [90] and Koller and Sahami [51] propose heuristics where they consider only the relationships between two attributes given the class variable at the time. Koller and Sahami [51] propose an approximative iterative algorithm that uses the cross entropy measure of two attributes given the class variable to find an approximative Markov Blanket. The cross entropy of A_i and A_j given C, using the conditional entropy is $H_P(C \mid A_i, A_j) - H_P(C \mid A_i)$, which is equivalent with the amount of irredundancy of C for A_i given A_j from Section 2.2. This algorithm iteratively deletes the attribute with the smallest cross-entropy for the class variable given one other attribute from the dataset until some stopping criteria is met (e.g. some predefined number of attributes are deleted or the cross-entropy of the remaining attributes is larger than a threshold γ). Using Example 2.1, Koller and Sahami's algorithm deletes, in a random order, the attributes A_5 and A_7 - they are redundant for C and for the other attributes - A_8 and A_3 - they are conditionally independent for C given A_2 - and A_1 and A_4 - they are redundant for C given A_2. Since this algorithm deletes attributes only by looking at its redundancy for C given another attribute, the algorithm fails to identify that A_1 and A_4 are important for the classification task. Recall that the MDL-FS score for both selective Naive Bayesian and TAN classifiers using a tree auxiliary structure of maximum log-likelihood correctly identifies the attributes A_1, A_4, A_2 and A_6 as useful for the classification task.

Since Tsamardinos and Aliferis's and Yu and Liu [93]'s heuristics also consider the interactions between attributes one at a time in an iterative algorithm similar with Koller and Sahami's algorithm, they have similar properties. Thus, they will also wrongly delete the attributes A_1 and A_4 from our example.

Fleuret [27] related the conditional probability definition to the notion of conditional mutual information. In his heuristic, he iteratively adds attributes that have high conditional mutual information scores with the class variable given each of the attributes that are already picked. Note that, in our example, the attributes A_2 and A_6 will be the first to be picked by the algorithm. However, the attributes A_1 and A_4 are again not picked because the conditional mutual information of A_2 and C given A_1, and A_4 respectively, are low.

In the sequel, one can use Pearson's correlation to study the paired correlation between two variables [39]. We now briefly review the concept of redundancy build upon Pearson's correlation coefficient used by Hall [41]. Informally speaking, he defines a subset of attributes S_c to be important for the classification task if each attribute from S_c is correlated to the class variable and not correlated to any other attribute from S_c. He measures the correlation between two variables using the difference between the entropy of a variable and the conditional entropy between the variable given the other variables. Upon applying Hall's concept of redundancy to our example, we find that the correlations between A_1 and C, between A_4 and C, and between A_1 and A_4 are considered. Since the relationship between the attributes A_1 and A_4 on the one hand and the class variable C on the other hand is not considered, the two attributes A_1 and A_4 would be deemed irrelevant for the classification task.

Our approach differs from previous work in the sense that we work with the conditional probability distribution directly and include the conditional log-likelihoods in an MDL-based function. As a consequence, the impact of these methods as compared with MDL-FS score for feature selection task is very different although they use similar definitions based on conditional entropy of the class variable given a set of attributes to denote useful and not useful set of attributes. The analysis is in favor for our method. Informally speaking, the previous algorithms will require that an attribute is irredundant for the class variable given *all* other attributes from the selected set. In our example, the previous algorithms will fail to identify the attributes A_4 and A_1 as important for the classification task because these attributes are redundant for the class variable given A_2 and thus their relevancy for the class variable at level 1 given each other is overlooked. In the previous sections we have shown that the MDL-FS score overcomes this drawback by: capturing first the strongest relationships between attributes and the class variable and evaluating a sum of terms that indicates the amount of irredundancy a set of attributes has for the class variable.

Kontkanen and co-authors [54] propose to perform feature subset selection using the supervised marginal log-likelihood (closely related with the conditional log-likelihood). When they evaluate a subset of features, they only consider the case when the attributes are independent of each other given the class variable. Their method is closely related with our method because it learns the joint probability distribution over the class variable and the attribute set. However, unlike our method, they do not use an auxiliary structure to express the joint probability distribution over the attribute set and, as a consequence, this method is too computationally expensive to consider the relationships between attributes given the class variable (for example to learn TANs). Jebara and Jaakkola [46] associate to each attribute a probability value with a maximum entropy discrimination framework. They use regression methods in discriminant functions (closely related with conditional log-likelihood) to perform classification or regression.

Furthermore, our MDL-C method discriminatingly learns the structure of Bayesian network classifiers [71] - (e.g. learns Bayesian network classifiers using the conditional probability distribution of the class variable given the attribute set). With this scope, Bilmes [8] uses a measure that prefers arcs between two attributes which have a high conditional mutual information given the class

variable but a low un-conditioned mutual information. Then, for the attributes A_i and A_j we have $I_P(A_i; A_j \mid C) - I_P(A_i; A_j)$ which we can equivalently rewrite in terms of (conditional) entropies as $-H_P(C) - H_P(C \mid A_i, A_j) + H_P(C \mid A_i) + H_P(C \mid A_j)$. Such an algorithm has a different behavior in learning the structure of the Bayesian network classifiers than an algorithm that uses the MDL-C score since it does not perform feature selection (e.g. the arcs are preferentially added between attributes of the entire set of attributes). Grossman and Domingos [37] also maximize the conditional log-likelihood using the parameters of maximum log-likelihood. For that, they use the regression algorithm of Zhou and Greiner [36]. Burge and Lane [12] learn also discriminative structures by using separate Bayesian network classifiers for each class value. They approximate the conditional log-likelihood for the two value class as the ratio between the log-likelihood of the classifier given one class variable and the classifier given the other class variable and the same classifier.

We note that there are several researchers that use the MDL score (or a MDL like score) for the feature selection task [52, 25, 83]. Kononenko [52] uses an MDL like score to assign the relevancy of an attribute for the class variable. Dy and Brodley [25] first cluster the dataset and then use an MDL like function to perform feature selection. Singh and Provan [83] first iteratively eliminate attributes if this improves the classification accuracy, and then construct a Bayesian network classifier using an MDL like score. Unlike in the previous work, we use an MDL like score for both the feature selection task and to construct a performant classifier.

3.6 Experimental Results

In the following, we study the feature-selection behavior of the MDL and MDL-FS functions in a practical setting by constructing selective Naive Bayesian and TAN classifiers from various different datasets using both functions. Then, we compare them with three other popular methods from feature selection literature: a wrapper method which uses the accuracy measure for training and testing the selective classifiers [50], Koller and Sahami's [51] and Hall's [41] filter methods.

For our experimental study, we use four datasets from the UCI Irvine repository eliminating any incomplete instances, and two artificially generated datasets. The characteristics of these UCI datasets are throughly analyzed in literature. For the *chess* dataset, Hall [41] obtains accurate selective Naive Bayesian classifiers over just 3 out of 36 attributes. From the *mushrooms* dataset simple logic rules can be extracted that contain only a small number of attributes (e.g. rules with only 2, 3 or 4 out of 22 attributes might have an accuracy between 98% to 100%). For the *splice* dataset, Domingos and Pazzani [19] point out that the most accurate classifiers were Naive Bayesian. The *oesoca dataset* was generated from a hand-crafted Bayesian network in the field of oesophageal cancer. The *artificial dataset* was generated by copying the 32 instances of the Example 2.1, 100 times, resulting in a dataset with 3200 instances. We used the method of Fayyad and Irani to discretize any continuous attributes in the various datasets [26].

In our study, we used the six datasets for learning several Naive Bayesian classifiers and TAN classifiers, with different algorithms and different scoring functions. In each experiment, we split each dataset randomly into a *training set* and a *test set* at a $2 : 1$ ratio; the training set was used to construct the classifier and the test set was used to establish the performance of the constructed classifier. We observe that in this case the training and test sets will be at different size. Thus, to fairly measure the conditional log-likelihood, we divide it by the size of the training set. We repeated each experiment 50 times, each time splitting the dataset anew in a training set and a test set to be able to compare the mean scores (to be able to say that one method is better than the other). We have performed Wilcox statistical tests, see Table 3.3, with the results obtained for each of the datasets for all methods presented in the experimental results for: (i) the number of selected attributes, (ii)

data	inst	attr	$l=0$	$l=1$	NB		TAN	
					% acc	*CLL*	*% acc*	*CLL*
chess	3196	36	2 ± 1	1 ± 1	88 ± 1	-3.11 ± 0.04	92 ± 1	0.89 ± 0.01
mush	5644	22	5 ± 0	14 ± 0	100 ± 0	-6.23 ± 0.03	100 ± 0	2.77 ± 0.02
splice	3000	60	0 ± 0	0 ± 0	96 ± 1	-0.20 ± 0.05	95 ± 1	3.43 ± 0.06
spam	4601	57	4 ± 0	4 ± 0	91 ± 1	-3.78 ± 0.04	92 ± 0	2.33 ± 0.03
oesoca	10000	25	0 ± 0	0 ± 0	71 ± 1	-1.63 ± 0.02	74 ± 0	1.46 ± 0.01
artif.	3200	8	0 ± 0	75 ± 0	81 ± 4	-1.64 ± 0.02	100 ± 0	0.84 ± 0.01

Table 3.1: The characteristics of the six datasets

the accuracies of the considered classifier and (iii) the conditional log-likelihood of the considered classifiers computed using the tree structured auxiliary network.

Prior to learning the classifiers, we established from the training set the numbers of redundant attributes at the levels 0 and 1. We report the averages and associated standard deviations obtained over all runs in percentage in Table 3.1. Recall that in Example 2.1, four attributes are redundant for the class variable at level 0 - they are A_1, A_4, A_5 and A_7 - and four attributes are redundant for the class variable at level 1 given one other variable - they are A_3, A_5, A_7 and A_8. However, when we split the artificial dataset in training and test set as described before, there is noise inherent to this process. We observe that most attributes are deemed irredundant for C except the copies A_5 and A_7 that are redundant for C at level 1 regardless of the splitting in training sets.

In our first experiment, we constructed from each dataset a full Naive Bayesian classifier and a full TAN classifier. The basic idea of this experiment was to establish baseline performances to compare the selective classifiers resulting from the other experiments against. Table 3.1 summarizes the results from the first experiment; it reports for each dataset the accuracy and the conditional log-likelihood divided by the size of the training set of constructed classifiers with the algorithms described in previous sections, averaged over 50 runs. We would expect the performance of the TAN classifier constructed from a specific dataset which has interactions between attributes, to be higher than that of the corresponding Naive Bayesian classifier, since a TAN classifier takes them into consideration while a Naive Bayesian classifier does not. For the splice dataset we find that the accuracy of the Naive Bayesian classifier slightly exceeds that of the TAN classifier. The slightly lower accuracy of the TAN classifier is readily explained from the random effect of splitting the dataset into a training set and a test set. For five of six databases, we found statistical significantly higher accuracy for TAN than for Naive Bayesian classifiers. Domingos and Pazzani [19] show that the improvement in accuracy classification by using a more complex Bayesian network classifier than the Naive Bayesian classifier is not necessary to be significant. Furthermore, for all databases, we found significantly higher conditional log-likelihoods for TAN than for Naive Bayesian classifiers. Recall that a TAN classifier has equal or higher log-likelihood than a Naive Bayesian classifier over the same attributes set; thus, it has equal or higher conditional log-likelihood with the same auxiliary network. We observe that, for mushrooms, splice and spambase datasets, the conditional log-likelihoods of the full TAN classifiers are positive but the ones of the Naive Bayesian classifiers are negative. Then, the attributes are strongly connected to each other given the classifier since the conditional log-likelihoods of TANs are higher than of NBs. But the attributes are weakly connected between them in the absence of the class variable - since a low log-likelihood for the auxiliary structure dominates the log-likelihood of the classifier.

For the second experiment, Table 3.2 reports the percentages of selected attributes and the classi-

fication accuracies of the selective Naive Bayesian and TAN classifiers constructed with the forward-selection using the MDL, MDL-FS and MDL-C functions and three popular feature-selection specific methods: the accuracy method for evaluation a specific classifier [58, 50] and Koller and Sahami [51]'s and Hall [41]'s algorithms described in the previous section. In Table 3.3, we present some statistical tests we performed to compare the experimental results: the number of significant wins-losses (we have performed Wilcox tests as described above), the averages and the rank of the MDL-FS score over the other methods used to construct (selective) Naive Bayesian classifiers. According with the definition of the wrapper and filter approach [50, 90], the MDL, MDL-FS and MDL-C algorithms are wrappers when the performance measure is the conditional log-likelihood, and they becomes filters when the performance measure is the classification accuracy. Whereas, if the accuracy measure is used instead of the MDL score, the resulting algorithm is a wrapper when the algorithm evaluates selective NB and TAN classifiers over the test set using the accuracy measure, and a filter when the algorithm evaluates the performance of the specific classifiers with the conditional log-likelihood.

We observe that, upon constructing a selective Naive Bayesian classifier, with the MDL-FS function, more selective classifiers were obtained than with the MDL function. From Section 3.3, we recall that the MDL function serves to remove any attribute that is redundant for the class variable at level 0; it tends not to remove any other attribute unless it is very weakly related with the class variable. We observe that indeed the 4 attributes redundant at level 0 were eliminated from the classifier for the artificial dataset. The more selective behavior was expected of the MDL-FS function as it removes, not just attributes that are redundant at level 0, but also many attributes that are redundant for the class variable at level 1. We observe that indeed, for the artificial dataset, in addition to the four attributes redundant at level 0, attributes redundant at level 1 for the class variable - these are A_3 and A_8 - were removed.

In the sequel, upon constructing a TAN classifier with the MDL function, hardly any attributes were removed. From Section 3.3, we recall that the MDL function tends to remove redundant attributes only if they are very weakly related with the other attributes. From the feature-selection results obtained, we thus have that, apparently, most redundant attributes show a relatively strong relationship with one or more other attributes. For the artificial dataset, the redundant attributes at level 0 or 1, because they are strongly connected with other attributes, were indeed not eliminated. With the MDL-FS function, more selective TAN classifiers were obtained. For the artificial dataset, redundant attributes at level 1 were indeed eliminated; the resulting TAN classifier is presented in Figure 3.1 in the middle. We note that, although with the MDL-FS function far more selective classifiers were yielded, the accuracies obtained are approximately the same as with the MDL function. For the conditional log-likelihood measure function for all datasets except the splice dataset and the artificial dataset, the conditional log-likelihoods for the MDL-FS function are slightly lower than for the MDL function. From Section 3.1.2 we recall that the conditional log-likelihood score increases when the complexity of the classifier increases. We note that the feature-selection behavior of the two functions is relatively robust as the standard deviation of the average accuracies is quite small. Higher standard deviations of the conditional log-likelihoods were obtained for the MDL scoring than for the MDL-FS function, showing that the latest function models better the conditional log-likelihood than the earlier one.

The classifiers learned with the MDL-C scoring functions are less selective than the classifiers learned with the MDL-FS function because the penalty term of MDL-C is smaller than the penalty of MDL-FS and thus MDL-C is less selective. In the sequel, the classifier learned with MDL-C is more selective than the classifiers learned with the MDL function because of the use of the conditional log-likelihood. Note that for TAN classifiers, where the complexity of the classifier is higher than the complexity of the tree structured auxiliary structure, with the MDL-C function we have higher

conditional log-likelihood than with the MDL-FS function. As in Section 3.4, for TAN classifiers, we thus have that MDL-C is less selective than MDL-FS but the classifiers learned with MDL-C are more performant than the ones learned with the MDL-FS function.

The accuracy method has the highest accuracies for both Naive Bayesian and TAN classifiers since it evaluates the accuracy itself in constructing selective classifiers. However, this algorithm is computationally very expensive since we have to compute each step the accuracy over the test set; the amount of evaluation also increases with the complexity of the classifiers we construct. The conditional log-likelihoods of this function are slightly worse than the conditional log-likelihoods for our MDL-FS function. However, we observe that the standard deviations are considerable higher for this function than for the MDL-FS score because various classifiers can have good accuracies but their conditional log-likelihood can be rather large. For example, for the artificial dataset, all TAN classifiers which contains the attributes A_1, A_2 and A_4 and an arc between A_1 and A_4 have the accuracy 1 (e.g. full TAN classifiers and the selective TAN obtained with the MDL-FS function) whereas their conditional log-likelihood can vary. For the artificial dataset, the greedy algorithm might fail in learning the complex relationship between attributes (e.g. the xor between A_1 and A_4). However, for the same dataset, the TAN classifier obtained using the backward elimination is the one illustrated in Figure 3.1 on the right; in Section 3.2, we show that we need a more complex auxiliary network than the tree structured one to learn this classifier with the MDL-FS score.

Koller and Sahami's approximative method depends heavily on the threshold γ over which an attribute is considered useful for the classification task. The higher the threshold γ the more selective is this method, but the performance of the selective Naive Bayesian classifier can be diminished; the lower the threshold, the less selective the resulting classifier is. In this paper, we present results with $\gamma = 0.05$; for all datasets the selection is very strong and the accuracies are comparable with the full Naive Bayesian classifiers. Hall's and Koller and Sahami's algorithms, on average, for Naive Bayesian classifiers, have a comparable performance regarding the number of selected attributes and the classification accuracy with the algorithms using the MDL-FS function. However, the conditional log-likelihood, on average, even though it is positive, is considerable smaller than the conditional log-likelihood obtained using the MDL-FS score.When we compare these algorithms with the other algorithms for constructing TAN classifiers, we find they are very selective, with slightly worse accuracies, but with much worse conditional log-likelihoods. For the artificial dataset, both algorithms select on average only two attributes A_2 and A_6, as when the MDL-FS score for Naive Bayesian classifiers is used. As we have showed in the previous section, neither of these two methods, however, are able to identify the XOR relationship between A_1 and A_4.

We observe that the MDL-FS has the best conditional log-likelihood score. It is the second most selective methods (on average only Hall's method reduces more attributes). The accuracies of the MDL-FS method are comparable with the accuracies of the other algorithms. Let's now compare the MDL-FS score function with the algorithms for constructing (selective) Naive Bayesian classifiers. We observe that the more selective algorithms than the MDL-FS for TAN's are: the algorithm which uses the classification accuracy for evaluation, and the ones which we presented previously for constructing Naive Bayesian classifiers Hall's, Koller and Sahami's and MDL-FS for NB algorithms.

		NB			TAN		
alg	*data*	*% sel*	*% acc*	*CLL*	*% sel*	*% acc*	*CLL*
MDL-FS	chess	13 ± 2	94 ± 0	0.41 ± 0.01	42 ± 4	93 ± 1	0.86 ± 0.02
	mush	5 ± 0	98 ± 0	0.86 ± 0.01	51 ± 4	100 ± 0	2.56 ± 0.05
	splice	22 ± 1	96 ± 1	1.59 ± 0.03	14 ± 1	94 ± 1	1.75 ± 0.05
	spam	22 ± 1	93 ± 0	0.88 ± 0.02	59 ± 4	93 ± 0	2.01 ± 0.08
	oesoca	26 ± 2	69 ± 1	0.97 ± 0.01	47 ± 5	73 ± 1	1.24 ± 0.05
	artif	25 ± 0	87 ± 1	0.47 ± 0.01	50 ± 0	100 ± 0	0.76 ± 0.01
MDL-C	chess	20 ± 2	94 ± 0	0.41 ± 0.01	49 ± 6	93 ± 2	0.88 ± 0.02
	mush	9 ± 0	99 ± 0	0.86 ± 0.01	60 ± 3	100 ± 0	2.68 ± 0.05
	splice	46 ± 2	95 ± 1	1.59 ± 0.03	23 ± 2	94 ± 1	2.12 ± 0.07
	spam	29 ± 2	93 ± 1	0.88 ± 0.02	73 ± 2	92 ± 1	2.26 ± 0.04
	oesoca	26 ± 2	68 ± 1	0.98 ± 0.01	52 ± 5	74 ± 1	1.29 ± 0.04
	artif	28 ± 1	87 ± 1	0.47 ± 0.01	50 ± 0	100 ± 0	0.75 ± 0.01
MDL	chess	58 ± 3	88 ± 1	-1.50 ± 0.22	97 ± 1	92 ± 1	0.90 ± 0.02
	mush	92 ± 3	100 ± 0	-6.08 ± 0.14	94 ± 2	100 ± 2	2.77 ± 0.03
	splice	47 ± 2	96 ± 0	1.25 ± 0.05	34 ± 1	96 ± 0	2.47 ± 0.04
	spamb	96 ± 0	90 ± 1	-3.74 ± 0.06	95 ± 1	92 ± 1	2.34 ± 0.04
	oesoca	72 ± 2	72 ± 1	-0.31 ± 0.14	98 ± 2	74 ± 1	1.45 ± 0.02
	artif.	50 ± 0	81 ± 1	-0.02 ± 0.01	100 ± 0	100 ± 1	0.84 ± 0.01
Acc	chess	14 ± 1	95 ± 1	0.40 ± 0.01	14 ± 0	95 ± 0	0.68 ± 0.01
	mush	14 ± 0	100 ± 0	0.23 ± 0.01	14 ± 0	100 ± 0	0.98 ± 0.01
	splice	19 ± 4	95 ± 1	1.40 ± 0.06	19 ± 4	95 ± 1	1.76 ± 0.15
	spam	25 ± 5	93 ± 1	0.30 ± 0.20	27 ± 4	93 ± 1	1.14 ± 0.10
	oesoca	43 ± 9	72 ± 1	0.37 ± 0.25	64 ± 6	75 ± 1	1.32 ± 0.03
	artif.	12.5 ± 0	88 ± 1	0.29 ± 0.01	12.5 ± 0	87 ± 1	0.30 ± 0.01
K & S	chess	11 ± 0	90 ± 1	0.36 ± 0.01	11 ± 0	90 ± 1	0.63 ± 0.01
	mush	37 ± 1	100 ± 0	-2.41 ± 0.07	37 ± 1	100 ± 0	1.86 ± 0.05
	splice	29 ± 1	96 ± 1	1.46 ± 0.05	29 ± 1	96 ± 1	2.30 ± 0.04
	spam	18 ± 1	91 ± 1	-0.11 ± 0.07	18 ± 1	91 ± 1	1.30 ± 0.05
	oesoca	24 ± 1	70 ± 1	0.83 ± 0.03	24 ± 1	71 ± 1	1.08 ± 0.02
	artif.	26 ± 0	87 ± 2	0.43 ± 0.03	26 ± 0	88 ± 1	0.49 ± 0.03
Hall	chess	8 ± 0	91 ± 1	0.38 ± 0.08	8 ± 0	91 ± 1	0.61 ± 0.01
	mush	9 ± 1	100 ± 0	0.52 ± 0.01	9 ± 1	100 ± 0	0.95 ± 0
	splice	10 ± 1	94 ± 1	1.25 ± 0.02	10 ± 1	94 ± 1	1.40 ± 0.02
	spamb	25 ± 2	80 ± 1	-1.13 ± 0.16	25 ± 2	83 ± 1	0.44 ± 0.03
	oesoca	24 ± 1	68 ± 1	0.90 ± 0.01	24 ± 1	68 ± 1	1.00 ± 0.01
	artif.	25 ± 0	87 ± 1	0.46 ± 0.01	25 ± 0	88 ± 1	0.46 ± 0.01

Table 3.2: The feature-selection results obtained for Naive Bayesian and TAN classifiers, with the forward-selection (FS) algorithms using the MDL, MDL-C and MDL-FS functions that use a maximum log-likelihood tree auxiliary network and the accuracy evaluation wrapper method and Koller and Sahamis and Halls filter algorithms.

		NB			TAN		
	alg	*% sel*	*% acc*	*CLL*	*% sel*	*% acc*	*CLL*
Average	full BNC	100	87	−2.77	100	92	1.95
	MDL-FS	19	90	0.86	42	92	1.53
	MDL-C	26	90	0.87	51	92	1.66
	MDL	69	88	−1.73	83	91	1.79
	Acc	21	91	0.49	27	93	1.03
	K&S	24	88	0.09		92	1.28
	Hall	17	86	0.40		89	0.81
Nr. sign.	full BNC	6-0	2-2	6-0	6-0	0-0	0-5
wins MDL-FS	MDL-C	6-0	0-1	0-0	5-0	0-0	0-4
from	MDL	6-0	2-2	6-0	6-0	6-0	0-6
	Acc	2-2	0-3	6-0	2-4	1-2	4-1
	K&S	2-1	2-2	6-0	1-5	4-1	5-1
	Hall	2-3	3-2	6-0	0-6	4-0	6-0
Rank	full BNC	7.0	3.34	7.0	7.0	1.67	1.0
	MDL-FS	2.0	2.67	1.17	3.5	2.0	4.17
	MDL-C	3.33	2.5	1.0	4.5	2.0	3.33
	MDL	5.83	3.83	5.83	6.0	1.67	1.33
	Acc	3.17	1.17	4.0	3.33	1.5	5.5
	K&S	2.5	4.17	4.83	2.33	4.17	5.0
	Hall	1.67	4.0	4.33	1.33	5.0	6.83

Table 3.3: Wilcox significance tests for the MDL-FS function when compared with the MDL and accuracy scores, K & S and Hall's methods for Naive Bayesian and TAN classifiers.

Chapter 4

Conclusion

In the first part of this thesis, we have studied the feature-selection behavior of the MDL-FS function, an MDL kind of function, for learning Bayesian network classifiers from data. We define the concept of redundant and irredundant attributes for the class variable given sets of attributes; based on the cardinality of these attributes, we have different levels of redundancy and irredundancy. Based on the observation that the poor feature-selection behavior of the MDL function is due to the use of the joint probability distribution over a classifier's variables, we have analysed an MDL-based function that captures the conditional distribution instead of the joint probability from the standard MDL. Since computing conditional log-likelihood is generally acknowledged to be hard, we associate to each Bayesian network classifier an auxiliary network to model also the distribution over the variables set. We have argued, both theoretically and experimentally, that the MDL-FS function is better tailored to the task of feature selection for more complicated Bayesian network classifiers than the Naive Bayesian ones than the MDL score: with the MDL-FS function, classifiers are yielded that have a performance comparable to the ones found with the MDL function, yet include fewer attributes. We also have showed that a slight modification of the MDL-FS function results in a score suited to learn Bayesian network classifiers from data. We perform some experiments that compare our method with popular methods from feature selection literature; in some cases, with the MDL-FS score, we have obtained better or/and more selective classifiers.

Part II

Evolutionary MCMC

Chapter 5

Preliminaries

The Markov chain Monte Carlo (MCMC) framework is used for sampling from complicated target distributions (e.g. multi-variable distributions with non-linear correlations between dimensions) that cannot be sampled with simpler, distribution specific, methods. MCMC algorithms are applied in many fields, and their use in Machine Learning has recently been advocated in Andrieu et al. [3]. There are common points in the stochastic process of evolutionary computation (EC) and MCMC: both are *Markov chains* with fixed transition probabilities between states. Furthermore, from all the standard MCMC techniques, the Metropolis-Hastings (MH) algorithms have the most in common with EC: both MH and EC algorithms have a selection like step to propagate good individuals to the next generation. There are however also many differences that arise from the different scope of these algorithms: EC is used for optimization and MCMC is used for sampling. Furthermore, MCMC uses a single chain which runs for a long time, whereas EC algorithms use a population of individuals that interact. Motivated by the common points of these two algorithms, and to improve the efficiency of standard MCMC, there are some variants that work with a population of MCMCs. In the Evolutionary MCMC (EMCMC) framework of Drugan and Thierens [23, 22], population-based MCMCs exchange information between the individual states, such that at population level, they are still MCMCs. In this part of the thesis, we propose a general *EMCMC* framework where we theoretically and experimentally study recombination operators, acceptance rules, and their interaction.

Ideally, an MCMC algorithm proposes individual states (or individuals) directly from the target distribution. Such an algorithm would mix – or converge to the target distribution – very fast since all states would be proposed in proportion with their target probability. Unfortunately, in practice one cannot sample directly from the target distribution. A standard MCMC typically generates individuals with a distribution (e.g. the normal distribution) that does not have any knowledge of the target distribution. We study proposal distributions that try to adapt to the target distribution by using recombination operators. Due to the sampling process the population contains more often individual states with high probability in the target distribution; the recombination operators exploit the structural information present in the parent states to generate new states which also have a high probability in the target distribution. If the exploration bias of the recombination operators coincides with the structural relationships present in the target distribution then the proposed states will more likely be accepted by the acceptance rule, thus increasing the performance of the EMCMC sampling algorithm. In this perspective, the recombination operators adapt the proposal distribution towards the target distribution. We call EMCMCs that use recombination to adapt the proposal distribution *recombinative EMCMCs.*

In this part of the thesis, we design recombination operators for efficient multi-dimensional EM-

CMCs. On one hand, we want to "economically" generate new individuals; on the other hand, we want to generate "good" individuals such that the EMCMC algorithm is performing well. We show how our framework extends the recombination operators described in the literature; here, recombination exploits new types of commonalities and relationships - such as correlation - between the dimensions of the target distribution. We identify and discuss important constraints that are reflected in the sort of recombination operators allowed in order to preserve the Markov chain property at population level.

We investigate the properties of recombination operators to further integrate them into EMCMCs that sample from the desired target distribution. The mutation and recombination operators usually have no information about how fit the individuals in the current and proposed population are. Then, as in standard MCMCs, we need an acceptance rule to ensure correct sampling from the target distribution. We investigate various acceptance rules and their interactions with various mutation and recombination operators. We show where and how the use of recombination can improve the performance of MCMCs.

5.1 Background: MCMC framework

MCMC is a general framework to generate samples $X^{(t)}$ from a probability distribution $P(\cdot)$ while exploring its search space $\Omega(X)$ using a *Markov chain*. MCMC does not sample directly from $P(\cdot)$, but only requires that the density $P(X) > 0$ can be evaluated within a multiplicative constant $P(X) = P'(X)/Z$, where Z is a normalization constant and $P'(\cdot)$ is the unnormalized target distribution. A Markov chain is a discrete-time stochastic process $\{X^{(0)}, X^{(1)}, \ldots\}$ with the property that the state $X^{(t)}$ given all previous values $\{X^{(0)}, X^{(1)}, \ldots, X^{(t-1)}\}$ only depends on $X^{(t-1)}$: $P(X^{(t)} \mid X^{(0)}, X^{(1)}, \ldots, X^{(t-1)}) = P(X^{(t)} \mid X^{(t-1)})$. We call $P(\cdot \mid \cdot)$ the *transition matrix* (or distribution) of the Markov chain. A *homogeneous* MCMC has a *stationary* - this is, independent of time t - *transition matrix* (or distribution) with the following properties: (i) all the entries are non-negative, and (ii) the sum of the entries in a row is 1. We consider an MCMC homogeneous unless specified otherwise.

In infinite time, an MCMC converges to the probability distribution $P(\cdot)$, thus it samples with higher probability from more likely states of $P(\cdot)$. A MCMC which has an *irreducible* and *aperiodic* stationary transition matrix, and the probability 1 to pass any state infinitely often (or all states are *recurrent*), converges to a unique *stationary distribution* [89]. A MCMC chain is irreducible if, and only if, every state of the MCMC chain can be reached from every other state in several steps. A MCMC is aperiodic if, and only if, there exist no cycles to be trapped into. A sufficient, but not necessary, condition to ensure that $P(\cdot)$ is the stationary distribution is that the irreducible and aperiodic MCMC satisfies the *detailed balance condition* [3]. A MCMC satisfies the detailed balance condition if, and only if, the probability to move from X to Y multiplied by the probability to be in X is equal to the probability to move from Y to X multiplied by the probability to be in Y: $P(Y \mid X) \cdot P(X) = P(X \mid Y) \cdot P(Y)$.

5.1.1 Metropolis-Hastings algorithms

Many MCMC algorithms are Metropolis-Hastings (MH) algorithms [66, 45]. Since we cannot sample directly from $P(\cdot)$, MH algorithms consider a simpler distribution $S(\cdot \mid \cdot)$, called the *proposal distribution* for sampling the next state of an MCMC chain. $S(Y \mid X^{(t)})$ generates the candidate

state Y from the current state $X^{(t)}$, and the new state Y is accepted with probability:

$$A(Y \mid X^{(t)}) = \min(1, \frac{P'(Y) \cdot S(X^{(t)} \mid Y)}{P'(X^{(t)}) \cdot S(Y \mid X^{(t)})})$$

If the candidate state is accepted, the next state becomes $X^{(t+1)} = Y$. Otherwise, $X^{(t+1)} = X^{(t)}$. For continuous search spaces, the transition probability for arriving in Y when the current state is $X^{(t)}$ is

$$T(Y \mid X^{(t)}) = S(Y \mid X^{(t)}) \cdot A(Y \mid X^{(t)}) + \delta_{X^{(t)}}(Y) \cdot r(X^{(t)})$$

where $\delta_{X^{(t)}}(Y)$ is the Dirac function, which is 1 only if $X^{(t)} = Y$, and 0 otherwise, and $r(X^{(t)})$ is the term associated with rejection $r(X^{(t)}) = \int_{\Omega(X)} S(Y \mid X^{(t)}) \cdot (1 - A(Y \mid X^{(t)}))dY$.

For finite search spaces, the transition probability for arriving in Y when the current state is $X^{(t)}$ is

$$T(Y \mid X^{(t)}) = \begin{cases} S(Y \mid X^{(t)}) \cdot A(Y \mid X^{(t)}) & \text{if } X^{(t)} \neq Y \\ 1 - \sum_{Y', Y' \neq X^{(t)}} S(Y' \mid X^{(t)}) \cdot A(Y' \mid X^{(t)}) & \text{otherwise} \end{cases}$$

A MH algorithm is aperiodic, since the chain can remain in the same state with a probability greater than 0, and by construction it satisfies the detailed balance condition, $P(X^{(t)}) \cdot T(Y \mid X^{(t)}) = P(Y) \cdot T(X^{(t)} \mid Y)$. If, in addition, the chain is irreducible, then it converges to the stationary distribution $P(\cdot)$. The rate of convergence depends on the relationship between the proposal distribution and the target distribution: the closer the (marginalization of the) proposal distribution is to the stationary distribution, the faster the chain converges. A popular Metropolis-Hastings algorithm is the *Metropolis algorithm*, where the proposal distribution is *symmetrical* $S(Y \mid X^{(t)}) = S(X^{(t)} \mid Y)$ and the acceptance rule becomes $A(Y \mid X^{(t)}) = \min(1, \frac{P'(Y)}{P'(X^{(t)})})$. Note that, to find these acceptance probabilities, we do not have to compute the proposal probabilities. Thus, the Metropolis algorithm is computationally more efficient than a general MH algorithm were we have to compute these proposal probabilities.

5.1.2 Mutation

A popular irreducible proposal distribution often used within MH algorithms is defined by a *mutation operator*. We present two mutation operators: uniform mutation for discrete search spaces and normal mutation for real-coded search spaces. We generically denote the proposal distributions resulting from mutation operators with S_m. In real space, we consider a state (or an individual) as a vector of real values with length ℓ, $X = (X[1], \ldots, X[\ell])$. Similarly, we consider a state in the discrete space as a string of ℓ characters, $X = (X[1], X[2], \ldots, X[\ell])$ Each position h in an individual X is instantiated with an *allele* $X[h] \in \Omega(X[\cdot])$, where $\Omega(X[\cdot])$ is the multi-set of all possible values of $X[\cdot]$. The h-th position in X is called the *locus* of $X[h]$, where $1 \leq h \leq \ell$.

Uniform mutation

The *uniform mutation operator* [62, 65, 85] randomly changes every value of each variable of the current state with a non-zero probability, called mutation rate. We denote with $\Delta(Y, X)$ some distance metric between the individuals Y and X.

For example, in the binary space, assuming a mutation rate of $p_m > 0$, the probability to generate a candidate state Y from $X^{(t)}$ is:

$$S_{m,u}(Y \mid X^{(t)}) = (1 - p_m)^{\ell - \Delta(Y, X^{(t)})} \cdot {p_m}^{\Delta(Y, X^{(t)})}$$

where $\Delta(Y, X^{(t)})$ is now the *hamming distance*.

The bigger the uniform mutation rate, the bigger the jump in the search space of the child state from the parent state. We denote with $S_{m,u}$ the *uniform mutation proposal distribution*. When the context is not ambiguous, we simple refer to it as mutation.

We now show that normal mutation is linearly scalable with the number of dimensions when generating a child.

Theorem 5.1 *The uniform mutation operator defines an irreducible, symmetric and stationary proposal distribution. The time complexity of generating one individual with $S_{m,u}$ is linear with the number of dimensions ℓ, $\mathcal{O}(\ell)$.*

Proof. $S_{m,u}$ is irreducible because from any string there is a non-zero probability to go to any other string, $S_{m,u}(Y \mid X^{(t)}) > 0$. $S_{m,u}$ is also stationary: it is invariant in time, and the sum of the elements on one row is 1, $\sum_{Y \in \Omega(X)} S_{m,u}(Y \mid X^{(t)}) = 1$. $S_{m,u}$ is symmetrical: the probability to generate an individual from another one depends on the number of positions in which the two individuals coincide.

The computational time to generate an individual in a ℓ dimensional space is ℓ times larger than to generate a child in 1 dimension because we have to multiply ℓ probabilities to generate an allele. We have $S_{m,u}(Y \mid X^{(t)}) = \prod_{h=1}^{\ell} S_{m,u}(Y[h] \mid X^{(t)}[h])$. The complexity is constant with the number of parents since there is only one of them. □

The *uniform mutation transition matrix*, $T_{m,u}$, proposes candidate individuals with $S_{m,u}$ and accepts them with A.

Corollary 5.1 *The uniform mutation transition matrix, $T_{m,u}$, defines an irreducible MH algorithm which converges to its stationary distribution.*

Normal mutation

This mutation changes every value of each variable of the current state with a non-zero probability adding a random normal variable sampled from a normal distribution, $\mathcal{N}(0, \sigma_m)$, with mean 0 and the standard deviation σ_m. The probability to generate a candidate state Y from $X^{(t)}$ is:

$$S_{m,n}(Y \mid X^{(t)}) = \prod_{h=1}^{\ell} \frac{1}{\sigma_m \sqrt{2 \cdot \pi}} \cdot \exp - \frac{(Y[h] - X^{(t)}[h])^2}{2 \cdot \sigma_m^2}$$

where $Y[h]$ and $X^{(t)}[h]$ are the values of Y and $X^{(t)}$, respectively, in position h. The larger the standard deviation, σ_m, the bigger the jump in the search space of the child state from the parent state. Note that with $S_{m,n}$, each variable is generated independently of the other variables of the same individual.

Theorem 5.2 *The normal distributed mutation operator defines an irreducible, symmetric and stationary proposal distribution, $S_{m,n}$. The time complexity of generating one individual with $S_{m,n}$ is linear with the number of dimensions ℓ, $\mathcal{O}(\ell)$.*

Proof. $S_{m,u}$ is irreducible because from any vector there is a non-zero probability to go to any other vector, $S_{m,n}(Y \mid X^{(t)}) > 0$. $S_{m,n}$ is also stationary: it is invariant in time, and $\int_{Y \in \Omega(X)} S_{m,n}(Y \mid X^{(t)}) = 1$. $S_{m,n}$ is symmetrical: the probability to generate an individual from another one depends on the distance between the two individuals.

The proof for the time complexity is similar to the proof from Theorem 5.1. □

The *normal mutation transition matrix*, $T_{m,n}$, proposes candidate individuals with $S_{m,n}$ and accepts them with A.

Corollary 5.2 *The normal mutation transition matrix, $T_{m,n}$, defines an irreducible MH algorithm that converges to the stationary distribution.*

5.1.3 Mixtures and cycles

Following simple mathematical rules, one can combine MCMC algorithms with the same target distribution in *mixtures* and *cycles* in another MCMC with that target distribution [89]. A *mixture* is a sum of transition distributions where at each step one distribution is selected according to some constant positive probability. For example, let us consider T_1 and T_2 two transition matrices with the target distribution $P(\cdot)$. Then, $p_1 \cdot T_1(\cdot \mid \cdot) \oplus (1 - p_1) \cdot T_2(\cdot \mid \cdot)$ is a mixture with the same target distribution, $P(\cdot)$, where $p_1 \in (0, 1)$ and $\oplus$ shows that, in one step, $T_1(\cdot \mid \cdot)$ or $T_2(\cdot \mid \cdot)$ are applied and not both at once. Mixtures have some remarkable properties. In a mixture of MH algorithms, if one of the algorithms is irreducible, then the mixture is irreducible and converges to the target distribution. Since the irreducible transition matrix goes in several steps from any to any individual, so does the mixture.

A *cycle* is the product of transition distributions where in each step one distribution is used in turn, and when the last one is used, the cycle is restarted. For example, $T_1 \times T_2$ is a cycle, with T_1 and T_2 as before. For cycles, there are no general rules to establish the irreducibility and detailed balance of an MCMC. They have to be checked for each cycle. If one of the transition distributions in a cycle is an irreducible MH transition matrix over a small set (for example, bounded in real-vector space $\mathbb{R}^l$) then the cycle converges to the target distribution.

5.1.4 Simulated annealing (SA)

SA [49, 31] is a minor modification of a single chain MH algorithm used for optimization. Instead of sampling from the distribution $P(\cdot)$, SA samples at step t from $P'^{(t)}(\cdot) = P'(\cdot)^{1/Temp^{(t)}}$, where $Temp^{(t)}$ decreases according to a cooling schedule to 0. With the temperature close to ∞, the chain accepts almost any candidate state according to the MH acceptance rule A, whereas, when the temperature is close to 0, the chain rejects almost all states that have lower unnormalized probability than the current one. Note that, for a constant temperature $Temp^{(t)}$, SA is an MCMC which converges to the distribution $P^{(t)}(\cdot)$. This distribution, $P^{(t)}(\cdot)$ is similar with the original distribution $P(\cdot)$, in the sense that if one state is more probable that another one in the original distribution, it is also more probable in the new distribution. If in addition $Temp^{(t)} < 1$, $P^{(t)}(\cdot)$ is concentrated in the maximum. However, every time the SA chain is cooled, the transition matrix is changed and the detailed balance is not satisfied. Yet, in infinite time, SA converges to the optimum and, more generally, if $Temp^{(t)}$ decreases to 1, SA converges to the stationary distribution $P(\cdot)$. SA is a *non-homogeneous MCMC* which converges to a given stationary distribution. The time to convergence depends on the cooling schedule. In practice, a fast cooling schedule is preferred to a slower one, increasing the risk of poor performance.

5.2 EMCMC framework

When we refer to the performance of an MCMC, we refer to how well an MCMC is mixing or how "fast" it converges to the target distribution. We say that an MCMC is mixing "well" if it rapidly traverses the search space and, in the same time, accurately samples the target distribution. Note that the mixing concept in (E)MCMC is not related to the mixing of building blocks in the EC literature.

A single MCMC is usually running for a long time until it converges to the stationary distribution $P(\cdot)$. There exist variations on the standard MCMC algorithm to speed up this mixing process. We are particularly interested in techniques that use multiple interacting chains in parallel as opposed to a single chain. We generically denote these techniques the EMCMC framework.

An EMCMC allows interactions between the individual chains under the assumption that individuals in the current population exchange information that "helps" the EMCMC to sample the desired distribution. Note that, in EMCMCs, the population is a multi-set of individual states rather than a collection of MCMCs: the current individual states depend on several states from the previous population. Now the sample at time t is the population $X^{(t)} = (x_1^{(t)}, \ldots, x_N^{(t)})$ of N states (or individuals) $x_\cdot^{(t)}$.

Definition 5.1 *An evolutionary Markov chain Monte Carlo (EMCMC) algorithm is a population MCMC that exchanges information between individual states such that, at population level, the EMCMC is an MCMC.*

Similarly to an MCMC, the main goal of an EMCMC is to sample from a given distribution, $P(\cdot)$. Ideally, an MCMC algorithm would propose individuals directly from the target distribution. Then, there would be no bottlenecks because the individual states are disproportionately proposed with their probability. Unfortunately, in practice, this is hardly the case since we do not know where in the search space there are the fit individuals. A standard MCMC, for example, generates individuals with some mutation proposal distribution (e.g. the uniform mutation proposal distribution $S_{m,u}$) that does not have any knowledge of the sampled distribution.

A method to speed up the mixing is to propose individuals using proposal distributions that are "close" to the target distribution. We use recombination operators that exploit the common structure of the parents. In the next section, we give general rules for designing proposal distributions (e.g. recombination) for efficient EMCMCs. In Chapter 6, we present various recombination operators and their relevant properties for EMCMCs. We propose several recombination operators for discrete spaces in Section 6.1. In Section 6.2, we use geometrical transformations - more specifically linear transformations like translation and rotation - of the parents into children to formally express our recombinations in real-coded spaces. We show that our framework extends the existing recombination operators; now, recombination exploits new types of commonalities and relationships between dimensions (e.g. correlation) in the target distribution. Furthermore, these recombinations can express more complex expressions between dimensions than linear ones because we can transform any function of two or more individuals.

We call EMCMCs that use recombination to exchange information between individuals *recombinative EMCMCs*. Section 6.3 shows how mutation can be combined with recombination into irreducible proposal distributions.

The recombination operators usually have no information about how fit the individuals in the current and proposed population are. Then, like for the standard MCMCs, we need acceptance rules to sample from the target distribution. In Chapter 7, we investigate various acceptance rules and their interactions with various proposal distributions. In Section 7.1, we investigate the properties of the recombinative EMCMC resulting from combining recombination with the standard acceptance rule.

Although the detailed balance condition does not hold for this recombinative EMCMC, we prove that, in certain conditions, it still can be used to sample from the target distribution $P(\cdot)$.

Detailed balance is a sufficient, but not a necessary condition, for an irreducible aperiodic EMCMC to converge to a desired target distribution $P(\cdot)$. By definition, MH algorithms are aperiodic and have detailed balance. Most EMCMCs are irreducible MH algorithms - by use of mutation - and apply recombination in the proposal distribution. In Section 7.2, we show that to obtain detailed balance, the individuals that interact through recombination need also to interact in the acceptance rule.

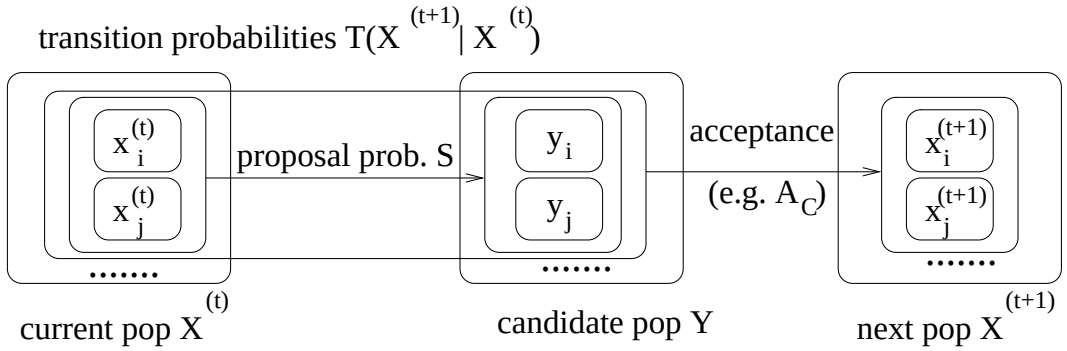

Figure 5.1: MH algorithms in the EMCMC framework: from two or more individuals (e.g. $x_i^{(t)}$ and $x_j^{(t)}$) we generate children (e.g. y_i and y_j) using a proposal distribution S that may exchange information between individuals and accept or reject them using an acceptance rule (e.g. the coupled acceptance rule A_C). The transition probabilities are considered at population level (e.g. $T(X^{(t+1)} \mid X^{(t)})$).

We generically call this the coupled acceptance rule. However, such an acceptance rule has negative effect over the performance of an EMCMC. If some children are fit individuals but the others are not, this acceptance rule can reject "good" individuals whereas the standard MH acceptance will always accept them. Figure 5.1 outlines the MH algorithms in the EMCMC framework.

The acceptance rules directly derived from the EC's elitist selection rules results in elitist acceptance rules that accept the most fit parents and children. In Section 7.3, we show that in general, such EMCMCs do not have detailed balance; they have a target distribution which is disproportionately peaked in the high regions of the initial target distribution. We control the performance of these EMCMCs by controlling the height of these peaks with a temperature schedule attached to the individuals from the population such that the higher the fitness the lower the attached temperature. Then, the fit individuals remain longer in the population whereas with less fit ones we explore the search space.

In Chapter 8, we analytically test the discussed (E)MCMCs on several problems. The obtained results show that recombination improves the mixing of EMCMC especially when the standard MH acceptance rule is used with recombination. Chapter 9 concludes the second part of this thesis.

5.3 Recombination for EMCMCs - general remarks

In the previous section we have outlined the EMCMC framework. In the following, we study how to design recombination operators for efficient - computationally inexpensive and well performing - multi-dimensional EMCMCs. We know that the "closer" the proposal distribution to the target distribution, the better an MH algorithm is [3]. Mutation and recombination operators generate special cases of proposal distributions.

Definition 5.2 *A* recombination proposal distribution *is the distribution defined by the recombination probabilities to generate one or more children from two or more parents. A* mutation proposal distribution *is the distribution defined by the mutation probabilities to generate one child from one parent.*

Normal mutation, for example, generates a child from one parent with low computational cost, but does not make any assumptions about the target distribution. Recombination generates one or more children from two or more parents and make the assumption that "good" individuals are more often present in the population than "bad" ones and thus information present in two or more individuals might contain useful information about other "good" individuals. Therefore, recombination adapts the proposal probabilities from the current population. With recombination, we can design a proposal distribution that exploits the particularities of the target distribution and thus it can be more efficient than mutation. For example, in the next section, we propose recombinations that exploit the correlations between dimensions. Generically, we denote recombination proposal distributions with S_r.

In the next paragraphs, we define rules for recombination operators that generate proposal distributions for population-based MH algorithms. We are interested in irreducibility and symmetry - properties that are common for all proposal distributions - but also in the properties of the interactions between individuals (e.g. how they interact through recombination) and the interactions between the loci of individuals (e.g. correlations).

5.3.1 Restrictions on the proposal distributions for EMCMCs

Because we intend to further integrate the recombinative proposal distributions in the MH framework, there are several constraints in designing them. The probability to propose a new population from the existing one should be constant in time to generate an MCMC's transition matrix. We can only use the information from the current population to propose the next population because otherwise we do not have a Markov chain any more. Nevertheless, we should be able to compute non-symmetrical proposal distributions and we restrict the acceptance rules of these recombination operators with ones similar to the standard MH acceptance rule to have detailed balance. Note that the MH acceptance rule is computationally more expensive than the Metropolis acceptance rule since we need to compute the non-symmetrical probability to generate these children from their parents and the inverse corresponding probability to generate these parents from their children.

Since we investigate MH algorithms that converge to the target distribution, we are interested in search spaces that allow irreducible proposal distributions such that each state can be generated from other states infinitely often. To visit a state infinitely often with a real valued MH algorithm, we bound the search space. In the existing EMCMCs, the search spaces are also closed: the candidate individuals generated should belong also to the search space. The requirements and assumptions we add to a search space allow for proposal distributions with certain properties. However, in this way, we restrict the types of the search spaces we could use.

5.3.2 Symmetry vs bias

It is important to establish if a proposal distribution is symmetrical or not: for the non-symmetrical distributions, we have to compute the proposal probabilities, whereas for symmetrical ones we can use the Metropolis algorithm. With a symmetrical proposal distribution, the probability to generate candidate individual(s) from parent(s) is equal with the reverse probability. If, however, a proposal distribution is not symmetrical, it should compensate the extra cost in computing the proposal probabilities by, for example, being "closer" to the target distribution.

In the sequel, we think that the proposal distribution should not bias the sampling unless it contains information (it is "closer") about the target distribution. In the absence of acceptance - that is all the states are accepted - the transition distribution of an MH algorithm coincides with the proposal distribution. If the target distribution of the algorithm where all individuals that are proposed are

accepted is equal in all states - the distribution is uniform - we say that the proposal distribution does not bias the sampling. Otherwise, we say that the proposal distribution does bias the sampling. If the target distribution in the absence of acceptance is "closer" to the desired target distribution of an MH algorithm - in the presence of acceptance - than to the uniform distribution we say that the proposal distribution biases the sampling towards the target distribution.

A proposal distribution that depends on the target distribution is not symmetrical unless the target distribution is uniformly distributed. In this case, to have detailed balance, we need to compute each step the proposal probabilities to go from the current states to the proposed ones and vice-versa. Such an algorithm is computationally more expensive than generating states with a symmetrical, but non-biased, proposal distribution that can be used with the Metropolis acceptance rule. Thus, for a non-symmetrical distribution, we want that the proposal distribution biases the sampling towards the target distribution.

5.3.3 Scalable vs performant operators

When designing recombination operators, on one hand we want proposal distributions that exploit the properties of the target distribution (e.g. the relationships between dimensions). On the other hand we want computational inexpensive proposal distributions that allow to economically generate individuals (e.g. scalable distributions). In Chapter 6, we propose recombinations that exploit certain types of common structure of the search space and with a computational complexity comparable with mutation. In this case, we say that the recombination proposal distributions are efficient.

Definition 5.3 *We call an operator* linearly scalable in multi-dimension search spaces *if the computational time to generate a child linearly increases with the number of dimensions. We call a recombination operator* linearly scalable with the number of parents *if the computational time to generate a child linearly increases with the number of parents.*

In the next chapter we propose linearly scalable recombinations with the number of dimensions, ℓ, and with the number of parents, p. The complexity of most of the proposed symmetrical recombinations that generates one child from p parents is $\mathcal{O}(\ell \cdot p)$; these operators generate p children in $\mathcal{O}(\ell \cdot p^2)$. For some symmetrical recombinations that generate p children from p parents, the complexity time per child, in some cases, is $\mathcal{O}(\ell)$, like for mutation; to generate p children, they spend $\mathcal{O}(\ell \cdot p)$. Note that the time complexity to generate p children with normal mutation is $\mathcal{O}(\ell \cdot p)$. For our proposed non-symmetrical recombinations the computational effort is non-linear.

We argue that, if a target distribution has certain types of correlation between dimensions, a proposal distribution that exploits these correlations is better than a proposal distribution that does not.

Definition 5.4 *A proposal distribution has* no correlations *between loci of an individual if the generation of an allele in one locus is independent of generation of the alleles in the other loci. A proposal distribution has* correlations *between loci of an individual, if the generation of an allele on a locus depends on the alleles of the other loci.*

Note that the normal mutation $S_{m,n}$ does not exploit the correlations between dimensions.

Proposition 5.1 *A proposal distribution with no correlation between dimensions is linearly scalable with the number of dimensions.*

Proof. Each allele on a position is generated independently of the alleles of the other dimensions. □

5.3.4 Family vs population recombinations

Given the number of chains that interact, we distinguish between *family* and *population* recombinations. Recombining few states (e.g. two or three states) is an example of the first approach, while in the latter all chains from the population exchange information. For family recombination, each generation, the population is grouped in families such that all chains of an EMCMC, eventually, interact though recombination. The recombination proposal distribution, S_r is the distribution defined by the recombination probabilities at population level. At individual (and at family level), the proposal probabilities of recombination might not be stationary since they depend on the family members with which they are grouped. At population level, the probability to generate with recombination one population from another one is stationary.

Note that the larger the family, the more parents are used and the more computationally expensive a recombination is. Therefore, a large family should be justified by the complicated relationships between individuals that recombination models. In this thesis, we focus on small size population recombinations that are composed only of a family.

5.3.5 Irreducible vs reducible proposal distributions.

In the existing binary coded EMCMCs [23] and in some of the real-coded EMCMCs, recombination is respectful. With *respectful* recombination, Radcliffe [73], the common parts of vectors are protected against disruption: if at a certain locus the parents have identical alleles, then the children have the same alleles for that locus. All respectful recombinations are reducible, according with Drugan and Thierens [23], but non-respectful recombinations can be both irreducible or reducible. With respectful recombination, if in one dimension all parents have the same allele, there is no diversity in information that children can exploit. Then, we generate a new allele similarly with mutation. In Section 6.3, we show how to combine mutation - that is irreducible - with recombination to obtain irreducible proposal distributions.

Chapter 6

Recombination

In the previous chapter we have discussed the EMCMC framework in general terms. In the chapter we investigate the key properties of recombination operators for EMCMCs, more specifically we will study their symmetry, bias, irreducibility, and scalability.

First, we study these properties of recombination operators for discrete sampling spaces; in Section 6.2 we focus on recombination operators in real-coded sampling spaces. In Section 6.3 we show how to combine recombination with mutation to obtain irreducible proposal distributions.

6.1 Discrete recombination as proposal distribution

From a computational point of view, it is extremely important to establish if a recombination operator is symmetrical or not: for the non-symmetrical recombinations, we have to compute the proposal probabilities, whereas for symmetrical operators we can use the Metropolis algorithm which allows us to propose new individual states without calculating the proposal probabilities. In the next paragraphs, we propose two generic rules to construct symmetric recombinations that we further refine for binary sampling spaces. We also exemplify on non-symmetric recombinations for which the two rules do not hold.

For our examples on both symmetrical and non-symmetrical recombinations, we restrict our attention to *respectful* recombinations [73].

Focus: *Respectful recombination is the interaction between two or more parents in a search space that generate individuals in the same search space such that if at a certain locus the parents have identical alleles, then the children have the same alleles for that locus.*

Thus, with respectful recombination the common parts of strings are protected against disruption. In this thesis when we talk about recombination we always mean respectful recombination.

Theorem 6.1 *The respectful recombination operators are* reducible.

Proof. From a population of individuals with identical alleles on a locus, the probability to go to some population with another allele on that locus is 0. □

In Table 6.1 we present the operators composed from mutation and/or recombination, their irreducibility, their symmetry, and their number of parents compared with the number of children, their correlation and if they are linearly scalable or not.

type op	op	irred	symmetry	par/ch	corr	lin scal
mut	$S_{m,u}$	irred	symm	1 / 1	no	yes
recomb	S_{unif}	red	symm	2 / 2	no	yes
	S_{dif}	red	symm	3 / 1	no	yes
	S_{mask}	red	non-symm	2 / 1	no	yes
	S_{tree}	red	non-symm	N / 1	yes	yes
mixture	$S_{m\oplus r}$	irred	symm	2 / 2		
cycle	$S_{m\times unif}$	irred	symm	2 / 2	no	no
sums	S_{m+mask}	irred	non-symm	2 / 1	no	yes
	S_{m+tree}	irred	non-symm	N / 1	yes	no

Table 6.1: Properties of several discrete space mutation/recombination operators: if they are irreducible or not, symmetrical or not, and how many children are generated from how many parents, their correlation and if they are linearly scalable or not.

6.1.1 Symmetrical recombinations

In EMCMCs, the symmetry is obtained by preserving the distance between the parents and their children. For example, the distance between p children is equal with the distance between the p parents that generate the children, or the distance between a parent and its child is constant as compared with the distance between two other individuals in the population.

p parents generating p children

When the distance between the generated children is the same with the distance between their parents, the recombination operator is symmetrical.

Proposition 6.1 *Consider p parents uniform randomly chosen without replacement from the current population, $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$. The recombination operator where p candidate individuals, $\{y_i, \ldots, y_{i+p-1}\}$, are generated from these parents such that the distance between the parents is equal with the distance between their children*

$$\Delta(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}) = \Delta(y_i, \ldots, y_{i+p-1})$$

is symmetrical.

Proof. The parents and the children are interchangeable; they have the same distance in between them. Thus, this recombination is symmetrical. □

Note that if the number of children is different of p, in general, the symmetry condition does not hold. We discuss such examples in the next section.

The swapping recombinations, often used in EMCMCs and in standard GAs, are particular cases of the above proposition where the distance between individuals are kept constant by swapping alleles.

Proposition 6.2 *The recombination proposal distributions which swap parts of individuals in between chains using a uniform distribution are symmetrical, respectful and stationary.*

Proof. Since there are equal probabilities to swap alleles (parts) in between parents and in between children, this recombination is symmetrical and the distance between them remains equal. If the

parents have the same allele on a locus, so do the children since the swapping does not change the values of alleles. □

We have recombinations which exchange non-common alleles, e.g. uniform crossover, or parts of individuals, e.g. 1 and 2 point crossover [62, 65]. These recombinations usually have only two parents.

In binary space, an example of swapping recombination is *parameterized uniform crossover*, S_{unif}, which generates two candidate individuals by swapping alleles between two parents with a uniform probability, p_x. Thus, it is impossible to generate children that have other common alleles than their parents. Where the two parents differ, an allele is swapped with the probability p_x and is not swapped with the probability $1 - p_x$. The probability to generate y_i and y_{i+1} is

$$S_{unif}(y_i, y_{i+1} \mid x_i^{(t)}, x_{i+1}^{(t)}) = {p_x}^{\Delta(y_i, x_i^{(t)})} \cdot (1 - p_x)^{\Delta(x_i^{(t)}, x_{i+1}^{(t)}) - \Delta(y_i, x_i^{(t)})}$$

where $\ell - \Delta(y_i, x_i^{(t)}) - \ell + \Delta(x_i^{(t)}, x_{i+1}^{(t)})$ is the number of common alleles for y_i and $x_i^{(t)}$ but not common for $x_i^{(t)}$ and $x_{i+1}^{(t)}$. It is interesting to observe that the time complexity to generate two children from two parents with S_{unif}, like for uniform mutation, is linear with the dimensionality, $\mathcal{O}(\ell)$. Note that this operator has no correlations between dimensions.

For $p_x = 0.5$, the operator is called uniform crossover and has been applied for strings of bits [62] and for strings of real numbers [33].

Three parents generating one child

In the following, we introduce a general condition to design symmetrical recombinations using three parents which generate one child.

Proposition 6.3 *Consider three parents uniform randomly chosen without replacement from the current population, $\{x_i^{(t)}, x_{i+1}^{(t)}, x_{i+2}^{(t)}\}$. The recombination operator where a candidate individual, y_i, is generated from the three parents such that the total distance between parents is equal with the total distance between the candidate individual and $\{x_{i+1}^{(t)}, x_{i+2}^{(t)}\}$,*

$$\Delta(x_i^{(t)}, x_{i+1}^{(t)}) + \Delta(x_i^{(t)}, x_{i+2}^{(t)}) + \Delta(x_{i+1}^{(t)}, x_{i+2}^{(t)}) = \Delta(y_i, x_{i+1}^{(t)}) + \Delta(y_i, x_{i+2}^{(t)}) + \Delta(x_{i+1}^{(t)}, x_{i+2}^{(t)})$$

is symmetrical.

Proof. The parent $x_i^{(t)}$ and the child y_i are interchangeable; they have the same total distance with the other two parents. Thus, this recombination is symmetrical. □

As an example in the binary space, we propose the *total difference crossover*, S_{dif}. A similar type of recombination has been used in real coded EAs [84, 87]. The new individual, y_i has the same alleles like $x_i^{(t)}$ on the positions where the two other parents coincide. On the other positions, we flip the alleles of $x_i^{(t)}$ with the probability p_x. The probability to generate y_i is

$$S_{dif}(y_i \mid x_i^{(t)}, x_{i+1}^{(t)}, x_{i+2}^{(t)}) = {p_x}^{\Delta(y_i, x_i^{(t)})} \cdot (1 - p_x)^{\Delta(x_{i+1}^{(t)}, x_{i+2}^{(t)}) - \Delta(y_i, x_i^{(t)})}$$

Proposition 6.4 *S_{dif} is symmetric, respectful and stationary. The time complexity of S_{dif}, like for S_{unif}, is linear with the dimensionality, $\mathcal{O}(\ell)$. S_{dif} has no correlations between dimensions.*

Proof. The proof is immediate. □

The xor crossover [85] is a special case of S_{dif} where the probability to flip a bit is 1 for $x_i^{(t)}$'s bits where $x_{i+1}^{(t)}$ and $x_{i+2}^{(t)}$ disagree.

The main difference between the two symmetrical types of recombination is that one preserves the sum of distances between the three parents when generating a child and the other preserves the distance between two parents when generating two children.

6.1.2 Non-symmetrical recombinations

We show two types of non-symmetric recombinations – *masked recombination* and *population frequencies recombination* [61] – that are closely related with the symmetric ones, but which do not preserve the distance between parents and children.

Masked recombination

Masked recombination is a swapping recombination where only one child is generated. Then, the distance between parents might not be the same with the distance between the child and one of the parents. Thus, masked recombination is not symmetrical.

Inspired by the snooker recombination from real coded EMCMCs [33], we propose the *masked recombination* for binary strings, S_{mask}. A child y_i is generated from a parent, $x_i^{(t)}$, and a mask, $x_{i+1}^{(t)}$. The common alleles of $x_i^{(t)}$ and $x_{i+1}^{(t)}$ are passed to y_i, but the non-common alleles are flipped in $x_i^{(t)}$ with the rate p_x. If the candidate individual does not coincide in positions where $x_i^{(t)}$ and $x_{i+1}^{(t)}$ coincide, the proposal distribution is 0; otherwise the probability to generate y_i is

$$p_x{}^{\Delta(y_i,x_i^{(t)})} \cdot (1-p_x)^{\Delta(x_i^{(t)},x_{i+1}^{(t)})-\Delta(y_i,x_i^{(t)})}$$

This crossover and the parameterized uniform crossover has the same probabilities to generate one child. But, since S_{mask} generates only one child, in general, the distance is not preserved and the symmetry condition does not hold. Thus, we have to compute the probabilities to generate a candidate individual with S_{mask} in the acceptance rule of the MH algorithm.

Proposition 6.5 *S_{mask} is reducible and stationary. Consider that from a parent $x_i^{(t)}$ and a mask $x_{i+1}^{(t)}$ we generate a child, y_i with S_{mask}. Then, S_{mask} is non-symmetrical. The time complexity to generate a child with S_{mask} is linear with the string size ℓ, $\mathcal{O}(\ell)$. S_{mask} has no correlations between dimensions.*

Proof. Let's consider that $x_i^{(t)} \neq y_i$ because bits are flipped on some positions. In those positions, the mask $x_{i+1}^{(t)}$ and the child y_i has the same values, whereas $x_i^{(t)}$ and $x_{i+1}^{(t)}$ do not. Then, it is impossible to generate $x_i^{(t)}$ from y_i and $x_{i+1}^{(t)}$. The rest of the properties follow directly. □

Recombination using probabilistic models

This recombination builds a probabilistic model of the parents to generate the children. It is analogous to the operator that generates individuals for the estimation distribution (EDA) algorithms applied in Evolutionary Computation for solving optimization problems [70].

We propose the *tree frequencies probabilistic recombination*, S_{tree}, closely related with the probabilistic model of Baluja [4]. Unlike the previous recombination operators where an allele is generated only given the alleles on the same position, S_{tree} considers the dependencies between two positions in the population using the Chow and Liu [16] algorithm. Thus S_{tree} biases the exploration according to these non-linear correlations between dimensions.

In the following, we describe the algorithm we use to generate individuals with S_{tree}. This algorithm constructs from the population of current individuals a tree with maximum entropy using a mutual information function. The entropy describes the level of uncertainty in a statistical variable. Here, the frequencies of the alleles in a position define a statistical variable for that position. Mutual information captures the extent to which two statistical variables are dependent. This algorithm keeps adding dependencies between variables based upon their mutual information under the constraint of building a tree (e.g. there are no cyclic path between variables). The higher the mutual information is, the sooner the algorithm tries to add the dependency in the tree.

A root for this tree is chosen at random from the set of positions. The allele for the root position is chosen based on its frequencies in the current population. We iteratively generate the other alleles based on their dependency with an allele - called parent - which was already instantiated in the tree. If h is the root of the tree, then the allele $y_i[h]$ is generated using the distribution $N(y_i[h])/N$, where $N(y_i[h])$ is the number of alleles $y_i[h]$ in the current population. Otherwise, if h has the parent h_1 in the tree, then the allele $y_i[h]$ is generated with the probability

$$\frac{N(y_i[h], y_i[h_1])}{N(y_i[h_1])}$$

where $y_i[h_1]$ is the allele already generated in position h_1, and $N(y_i[h], y_i[h_1])$ is the number of individuals in the current population that have allele $y_i[h]$ on position h and allele $y_i[h_1]$ in position h_1.

We observe that S_{tree} is the most expensive recombinative proposal distribution we have investigated for EMCMC. Unlike the other discrete space recombinations, S_{tree} exploits some relationships between dimensions: it computes the dependencies between two positions in order to construct the tree of maximum entropy and to assign a value to an allele given its parent. Then, the generation of an allele on a position also depends on the alleles on another position.

Proposition 6.6 *S_{tree} is respectful, non-symmetrical, stationary and biases the exploration according to the non-linear correlations between dimensions. Assuming an acceptance rule that accepts everything, all individuals will become the same after several steps. The computational complexity to generate a child with S_{tree} is $\mathcal{O}(\ell^2 \cdot N)$, where l is the dimensionality and N the size of the population.*

Proof. When an individual is generated with S_{tree} and replaces a parent, some allele frequencies can increase at the cost of the others. Then, the probability to generate the current population from the next one is smaller than vice-versa, while the probability that the same alleles are generated at the next step increases. The computational complexity of this operator is given by building the maximum log-likelihood tree. Chow and Liu [16] show that this is $\mathcal{O}(\ell^2 \cdot N)$. □

S_{tree} is a generalization of Laskey and Myers [61]'s recombination proposal distribution; when generating an allele, they consider only the frequencies of the alleles on the same position and not also on the other positions as S_{tree} does. Therefore, their recombination, unlike S_{tree}, does not exploit the relationships between dimensions.

It is interesting to observe that these non-symmetrical recombination distributions cannot be simply used with the Metropolis-Hastings algorithm because the probability to generate the candidate population from the current one can be 0 even when the reverse probability is non-zero. Then, the probability to accept the candidate population is 0. In the next section, we show how to use non-symmetrical proposal distributions by adding mutation.

Given the number of chains which interact, we distinguish between *family* and *population* recombinations. Recombining few chains (e.g. two or three chains) is an example of the first approach,

while in the latter all chains from the population exchange information. The above recombination proposal distributions are all, except for S_{tree}, family recombinations since they recombine at most three individuals; S_{tree} is a population recombination since it constructs the tree of maximum log-likelihood given all the individuals in the current population.

In case of family recombination, we assume that each generation the population is uniformly random grouped in disjunct families such that each individual is belonging to exactly one family. All the chains from an EMCMC, eventually, interact in population recombinations. We call *recombination proposal distribution* the distribution defined by the recombination probabilities at population level. We denote it with S_r. At individual and at family level, the proposal probabilities of recombination are not stationary since they depend on the family members with which they are grouped. At population level, the probability to generate with recombination one population from another one is stationary.

For the above family recombinations (e.g. S_{unif}, S_{dif} and S_{mask}), the time complexity at population level is linear with the number of individuals in the population: each generation, each individual is randomly paired in exactly one family. The complexity of these recombination proposal distributions at population level therefore is $\mathcal{O}(\ell \cdot N)$. Note that, at population level, the complexity of the mutation proposal distribution depends also linearly on the number of individuals in the population $\mathcal{O}(\ell \cdot N)$. Then, at population level, S_{tree} is still the most expensive proposal distribution with $\mathcal{O}(\ell^2 \cdot N)$.

6.2 Real-coded recombination as proposal distribution

In the previous section we have proposed and investigated recombination proposal distributions for discrete space EMCMCs. In the following, we propose a general framework for real-coded sampling spaces where recombination operators generate children using linear transformations (e.g. rotation, translation). Consider the parent solutions that we intend to recombine as the points of a geometrical figure. Then, recombination itself can be considered as a geometrical transformation of the parents into their children, where the set of children are, at their turn, points of another geometrical figure. Using this framework, we propose new recombination operators that exploit new commonalities and relationships between the dimensions of the target distribution to adapt the proposal probabilities from the current population. For instance, with translation we assume that the set of parents suggest a direction of sampling where "good" individuals are. With rotation, we assume that we can obtain "good" individuals when rotating parents around a point given by the set of parents. We show that with these transformations we can express any proposal distribution for a real-coded EMCMC.

In Table 6.2 we present the operators we have studied in real-coded space, their irreducibility, their symmetry, their number of parents compared with the number of children and whether they exploit the correlations between dimensions.

6.2.1 Linear transformations

Let us consider that a child has a *main parent* to which a value is added that depends on a certain number of individuals in the current population. To generate the child, the main parent is transformed with a linear transformation of a function of p parents, $e(\cdot) : \Omega(x)^p \rightarrow \Omega(x)$. Recall that every linear transformation corresponds to a unique matrix (and vice versa).

We consider $\Omega(x)$ an *affine space*: for any individual $y_i \in \Omega(x)$ there is a unique function $e(\cdot)$ such that from $x_i^{(t)} \in \Omega(x)$ we generate y_i, $y_i = x_i^{(t)} + e(\cdot)$. Thus, a closed affine space $\Omega(x)$ allows for irreducible proposal distributions. The addition "$+$" and scalar multiplication "$\cdot$" are operations

type op	op	irred	symm	par/ch	corr	lin scal	param
normal mut	$S_{m,n}$	yes	yes	1 / 1	no	yes	σ_m
simplex	S_{rot}	no	yes	p / $1, \ldots, p$	yes	yes	β_i
	S_{dif}	no	yes	p / 1	no	yes	$\{\gamma_i, \ldots, \gamma_k\}$
	S_{pctx}	no	yes	p / p	some	yes	
	S_{scl}	no	yes	p / 1,p	no	yes	
	S_{ref}	no	yes	p / 1	no	yes	
	S_{up}	no	no		some	no	
	S_{wgh}	no	no	p / 1	some	no	
cyc & mix	$S_{m\times r\oplus r\times m}$	yes	yes	p / p	no/some		
& sum	$S_{m\times r\times m}, S_{r\times m\times r}$	yes	yes	p / p	no/some		

Table 6.2: Several mutation and recombination operators and their combination

associated with the affine space. The parents are commutative and associative give the addition "$+$" and the scalar multiplication "$\cdot$": the order of the parents is not important for the operators. An example of such affine space is the real vector space.

We denote with $\Gamma \in \Omega^{\ell\times\ell}(x[\cdot])$ a square matrix, $\ell \times \ell$, with statistical variables as elements that represent a linear transformation.

Focus. *A* proposal distribution *on the closed affine search space $\Omega(x)$ is the distribution defined by the probabilities to generate one or more individuals, y_i, by adding to the corresponding main parent, x_i, a linear transformation of a function $e(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)})$ that maps any, one or more parents to a single state in $\Omega(x)$ and a matrix Γ with $\ell \times \ell$ stochastic variables. Then*

$$y_i = x_i^{(t)} + \Gamma \times e(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}) \tag{6.1}$$

where $e(\cdot) : \Omega(x)^p \longrightarrow \Omega(x)$ is a function resulting in a vector and p is the number of parents.

Note that $e(\cdot)$ can be any function of the parents and it is specific for a proposal distribution. In the next section, we give various specific examples of the function $e(\cdot)$. It is interesting to observe that although Equation 6.1 represents a linear transformation, the correlations between the dimensions can also be non-linear because $e(\cdot)$ can be any function of the parents.

In the following, we show that all proposal distributions on an affine space can be expressed with Equation 6.1.

Theorem 6.2 *If $\Omega(x)$ is a closed affine space, then any irreducible proposal distribution for (population-based) MH algorithms can be expressed with Equation 6.1.*

Proof. Equation 6.1 can generate all individuals in $\Omega(x)$ from all other individuals by choosing different functions $e(\cdot)$ and different matrices Γ with stochastic variables as elements. □
Theorem 6.2 shows that we can obtain any proposal distribution by choosing, in Equation 6.1, various sets of parents, functions $e(\cdot)$ and matrices Γ. In particular, from Definition 5.2 it is clear how to obtain any mutation and recombination proposal distribution.

Corollary 6.1 *A mutation proposal distribution generates an individual y_i from $x_i^{(t)}$ such that*

$$y_i = x_i^{(t)} + \Gamma \times e(x_i^{(t)})$$

A recombination proposal distribution generates an individual y_i from two or more parents, $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$, such that

$$y_i = x_i^{(t)} + \Gamma \times e(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)})$$

For example, for the normally distributed mutation, Γ is a diagonal matrix with normally distributed variables from $\mathcal{N}(0, \sigma_m)$, and the function $e(x_i^{(t)}) = [1, \ldots, 1]$ is the identity vector. In the following we show that, with translation, we can generate any individual from a set of parents. Therefore, all the discussed operators can be formulated as translation. However, some linear transformations (e.g. rotation) have very complicated formulas as translations, so more specific expressions will be used.

6.2.2 Translation recombination

We call *translation*, the generation of a child y_i from a main parent $x_i^{(t)}$ with a *translation vector* $\overrightarrow{r}_i = \Gamma \times e(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)})$ in Equation 6.1, where Γ is a matrix with statistical variables as elements and $e(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)})$ a vector as before. To design efficient translation recombinations, we use a basic algebraic property that any real-valued vector can be uniquely projected on an orthogonal basis and a projection to an orthogonal basis determines a unique vector. As a consequence, any translation direction of $\overrightarrow{r}_i$ is a sum of ℓ perpendicular vectors, where ℓ is the dimension of the search space. We now have

$$\overrightarrow{r}_i = \gamma_0 \cdot \overrightarrow{a}_0 + \ldots + \gamma_{\ell-1} \cdot \overrightarrow{a}_{\ell-1} \tag{6.2}$$

where $(\overrightarrow{a}_0, \ldots, \overrightarrow{a}_{\ell-1})$ is an orthogonal basis and $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ statistical variables, for example, normally distributed that we use to vary the length of the corresponding vectors to obtain any length for the vector $\overrightarrow{r}_i$.

Definition 6.1 Translation recombination *is a recombination that generates one or more children* $\{y_i, \ldots, y_{i+p-1}\}$ *by translating their parents* $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$ *with a translation vector that depends on two or more parents using Equation 6.2.*

In the following proposition, we show that the translation vector from Equation 6.2 generates translation recombination proposal distributions; they are irreducible if $(\overrightarrow{a}_0, \ldots, \overrightarrow{a}_{\ell-1})$ is an orthogonal basis and $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ are statistical variables sampled from a distribution that covers the search space, $\Omega(x[\cdot])$.

Proposition 6.7 *Consider the orthogonal basis* $(\overrightarrow{a}_0, \ldots, \overrightarrow{a}_{\ell-1})$ *and a translation recombination that generates a child* y_i *from a main parent* $x_i^{(t)}$ *using the vector from Equation 6.2. This translation recombination is irreducible when the involved statistical variables,* $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ *are sampled from a distribution that covers the search space* $\Omega(x[\cdot])$*. It is symmetrical if* $\overrightarrow{r}_i$ *is independent of the main parent, and the statistical variables* $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ *are symmetrically distributed around* 0*. If the statistical variables* $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ *are independently generated, this proposal distribution is biased according to the linear correlations between dimensions.*

Proof. When $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ are sampled from a distribution that covers the search space, the component vectors (e.g. $\gamma_0 \cdot \overrightarrow{a}_0$) can have any length and we can translate the current individual in any direction and with any value in the search space. If $\overrightarrow{r}_i$ is independent of the main parent then $x_i^{(t)} = y_i + \overrightarrow{-r}_i$, and the probability to generate y_i from $x_i^{(t)}$ is equal to the probability to generate $x_i^{(t)}$ from y_i.

Consider that the statistical variables $\{\gamma_0, \ldots, \gamma_{\ell-1}\}$ are independently generated. Then, for such a proposal distribution, the correlation between dimensions are given by the relationships between the perpendicular vectors with different lenghts $\{\overrightarrow{a}_0, \ldots, \overrightarrow{a}_{\ell-1}\}$. But to compute a vector given other perpendicular vector(s), we have to solve linear equations; therefore, the resulting proposal distribution is biased according to the linear relationships between dimensions. □

Note that all the proposal distributions in real-coded EMCMCs can be considered translations: a child is generated by translating the main parent in the search space. Mutation, for example, can be considered the translation of the current individual $x_i^{(t)}$ with the unity vector $e(\cdot)$ in a direction and at a distance given by a diagonal matrix Γ. For the normally distributed mutation, Γ has normally distributed elements. However, some linear transformations, for example rotation, can be more efficiently expressed with specific equations rather than with Equation 6.2. Later, we show that with translation and rotation operators we exploit different types of relationships between dimensions.

Related work

In the previous section we have specified a general framework for defining recombination operators in proposal distributions for real-coded EMCMCs. In the literature only a few specific recombinative proposal distributions for real-coded EMCMCs can be found. We show that they are respectful, symmetrical or non-symmetrical, and that they do not bias the search according to the linear correlations between dimensions. In Section 6.2.5, we will propose translation recombinations that, unlike the proposal distributions discussed here do bias the search in correspondence with the linear correlations between dimensions. These proposal distributions can be symmetrical and non-symmetrical translations; some are symmetrical because they are constructed to preserve the distances between or/and the volume of the individuals in a family. We will also propose efficient non-symmetrical translation recombinations that bias the search toward the target distribution.

1. *Difference recombination.* It generates one individual by translating the main parent with a vector whose direction and size is given by the difference between two other individuals in the current population. Then, $\overrightarrow{r}_i = \gamma_0 \cdot \overrightarrow{a}_0$, where $\overrightarrow{a}_0 = x_j^{(t)} - x_k^{(t)}$ and $i \neq j \neq k$ and γ_0 is a constant. If, in addition, $x_j^{(t)}$ and $x_k^{(t)}$ are randomly chosen from the population, following Proposition 6.7, this recombination is reducible, symmetrical, and does not bias the resulting proposal distribution according to the linear correlations. Therefore, it does not exploit correlations between dimensions. Furthermore, this recombination is respectful: if the three parents have the same value on a position, so does the child.

   Difference recombination is linearly scalable with the number of dimensions and, because the number of parents is constant $p = 3$, the computational complexity is constant with the number of parents. This translation is commonly used in Differential Evolution algorithms [84], but also in some EMCMCs [86, 87, 23]. In the next section, we extend this operator to more than three parents.

2. *Snooker recombination.* If the translation vector $\overrightarrow{r}_i$ depends on all the parents - thus also on the main parent - the resulting proposal distribution is non-symmetrical in which case we need to compute the proposal probabilities for the Metropolis Hasting acceptance rule. In the existing EMCMCs, see Liang and Wong [63], Gilks and Roberts [34], Strens et al. [86], these recombinations depend on two parents and samples on the direction specified by them. Then, $\overrightarrow{r}_i = \gamma_0 \cdot \overrightarrow{a}_0$, where $\overrightarrow{a}_0 = x_j^{(t)} - x_i^{(t)}$. This recombination is, in general, non-symmetrical, does not exploit correlations between dimensions and is respectful.

   According to Proposition 6.7, this recombination is not biased and reducible. It is also respectful: if the two parents have the same value on a position, so does the child.

3. *Reflection recombination.* A special case of translation is reflection. Here the main parent is reflected through the geometrical shape given by the other parents, called the polygonal face.

Note that, if the reflection of the main parent exists, the distance between the main parent and a parent of the polygonal face is equal to the distance between the child and the same parent from the face. Such a recombination operator is symmetrical although the translation vector depends on all the parents in the family. Reflection for EMCMC was proposed by Strens et. al [86] for $p = \ell + 1$ parents. It should be noted that reflection is an adaption of the non-symmetrical simplex recombination used in optimization by Nelder and Mead [68]. Here the parent to be reflected is the one with the lowest value from all the parents, whereas for reflection in EMCMC it is chosen at random from the population. In the next section we will define a non-symmetrical recombination for real-coded EMCMCs similar to the simplex method.

Section 6.2.4 proposes translations that are symmetrical because they are constructed to preserve the distances between – and/or the volume of – the individuals in a family. We also propose efficient non-symmetrical translation recombinations that bias the search toward the target distribution.

6.2.3 Rotation recombination

In the following, we introduce the rotation recombination proposal distribution. Rotation is a move of a state around another state in the search space commonly used in 2D and 3D image processing. Using Equation 6.1, we call *rotation*, the generation of y_i from $x_i^{(t)}$ using the stochastic rotation matrix Γ and a state M around which an individual is rotated. In 2D, the rotation matrix is $\Gamma = \left[\begin{smallmatrix} cos(\alpha_i) & sin(\alpha_i) \\ -sin(\alpha_i) & cos(\alpha_i) \end{smallmatrix}\right]$, where α_i is a statistical variable representing the angle with which a state is rotated. Then, we have $e(x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}) = (x_i^{(t)} - M)$, where M can be any function of the parents. However, for more than two dimensions, Γ has a rather complicated expression and involves the multiplication of $\binom{l}{2}$ matrices of size $l \times l$. To efficiently rotate a state around another one in a multi-dimensional search space, we transform the real-valued space into *polar coordinates*; in these coordinates, the rotation of an individual around a state is equivalent with the translation of the corresponding angles in polar coordinates.

To compute the transformations into polar coordinates and vice-versa, we use the method described in Bauwens et al. [6]. Using the polar coordinates, we now describe the algorithm for rotating the state $x_i^{(t)}$ around a state M. We first translate the coordinate axes in M because we need to rotate around M. In polar coordinates, we have $(\rho(x_i^{(t)} - M), \alpha_i^{(t)})$, where

$$\rho(x_i^{(t)} - M) = sign(x_i^{(t)}[1] - M[1]) \cdot \sqrt{(x_i^{(t)} - M)' \cdot (x_i^{(t)} - M)}$$

$$\alpha_i[h] = \arcsin \frac{x_i^{(t)}[\ell - h + 1] - M[l - h + 1]}{\rho(x_i^{(t)} - M) \cdot \prod_{g=1}^{h-1} \cos(\alpha_i[g])}$$

where $1 \leq h \leq \ell - 1$, and $sign(\cdot)$ is the sign function and, by convention, $\prod_{g=1}^{0} cos(\alpha_i[g]) = 1$. We now rotate clockwise around ℓ axes with the angles $\beta = (\beta[1], \ldots, \beta[\ell - 1]) \in [-\pi/2, \pi/2]^{\ell-1}$ to obtain the candidate individual y_i. Thus, in polar coordinates, the vector $y_i - M$ has the same length but different angles than $x_i^{(t)} - M$; in short, $(\rho(x_i^{(t)} - M), \alpha_i + \beta)$. We transform these polar coordinates back into original coordinates $y_i - M$ and we obtain y_i:

$$y_i[1] = M[1] + \rho(x_i^{(t)} - M) \cdot \prod_{h=1}^{\ell-1} \cos(\alpha_i[h] + \beta[h])$$

$$y_i[h] = M[h] + \rho(x_i^{(t)} - M) \cdot \sin(\alpha_i[\ell - h + 1] + \beta[\ell - h + 1]) \cdot \prod_{g=1}^{\ell-h} \cos(\alpha_i[g] + \beta[g])$$

Note that these transformations to the polar coordinates only exist if $\cos(\alpha[g]) \neq 0$ and $\cos(\alpha[g] + \beta[g]) \neq 0$, for all $1 \leq g \leq \ell - 1$.

We now formally define the rotation recombination M.

Definition 6.2 Rotation *generates a child y_i by rotating a parent $x_i^{(t)}$ around a state $M \in \Omega(x)$ with an angle $\beta = (\beta[1], \ldots, \beta[\ell - 1]) \in [-\pi/2, \pi/2]^{\ell-1}$ composed of statistical variables such that*

$$y_i = M + (\rho(x_i^{(t)} - M), \alpha_i + \beta) \tag{6.3}$$

where $x_i^{(t)} = M + (\rho(x_i^{(t)} - M), \alpha_i)$ is the representation in polar coordinates of $x_i^{(t)}$. Rotation recombination *is a recombination that generates one or more children $\{y_i, \ldots, y_{i+p-1}\}$ by rotating their parents $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$ around a state M with Equation 6.3.*

A direct consequence of the previous definition is that, for rotation recombination, M and/or β depend on at least one other parent in addition to the main one.

Proposition 6.8 *Rotation recombination is reducible, linearly scalable with the number of dimensions and biases the exploration according to non-linear correlations between the dimensions of the sampling space.*

Proof. Consider the equations that describe the transformation in polar coordinates of a state. To multiply a vector with a transpose vector we need $\mathcal{O}(\ell)$ multiplications. Then, the computational effort to compute $(x_i^{(t)} - M)' \cdot (x_i^{(t)} - M)$ is linear $\mathcal{O}(\ell)$. In the sequel, the product of cosinesses $\prod_{g=1}^{h-1} \cos(\alpha_i[g])$ can be stored; when generating an allele h, we can use the previous product of cosinesses and multiply it with $\cos(\alpha_i[h-1])$. This is also linear with the number of dimensions. Similarly, we find that transforming a state back from polar coordinates is also linear. Translating a state into polar coordinates is linear because, in polar coordinates, each allele is independently updated. We conclude that rotation is linearly scalable. It is reducible because not all individuals from a real space can be generated by rotating a state around another one. Since M and/or β depend on at least one other parent than the main one, rotation recombination biases the exploration according to non-linear correlations between dimensions. All alleles – except the first one – non-linearly depend on the values of β and M and therefore on values of alleles in the other dimensions. □

Note that the correlations between dimensions exploited with the rotation operator are linear in polar coordinates. In the following section we propose a symmetrical rotation around the mean.

6.2.4 Simplex geometrical recombination operators

In the previous section, we have introduced the affine transformation framework for geometrical recombination proposal distributions. Although our proposed operators cover the general case where an arbitrary number of parents are used to generate one or any number of children, we will further focus our study on recombination operators that are simplex geometrical shapes. Simplex geometrical shapes are the shapes in a ℓ dimensional space consisting of $\ell + 1$ points and all their interconnecting line segments, etc. We are interested in non-degenerate geometrical shapes: there exists a unique ℓ dimensional space that passes through such geometrical shape with $\ell + 1$ vertexes. For example, two points define a unique line that passes through them, three points define a unique 2D plane, and four points a unique 3D space. We call these recombinations *simplex geometrical recombinations*.

Note that these simplex geometrical recombinations are different from simplex recombination in Evolutionary Computation where the lowest fitted individual is reflected. Here, simplex geometrical recombination is any linear transformation of simplex geometrical shapes.

6.2.5 Symmetric non-biased recombinations

We propose several symmetrical, simplex geometrical recombinations. Some of these recombinations (e.g. rotation, parent centric translation), like normally distributed mutation, generate individuals in the neighborhood of the main parent. But, with recombination, the size and the direction of the jump from the (main) parent depends on the parent set. Reflection, on the other hand, reflects the main parent through the polygonal face defined by the other ℓ parents. As a consequence, we consider that these recombinations adapt the corresponding proposal distribution from the current set of parents based on their proximity and correlations. Thus, if the parents are close to each other, the children are in the neighborhood of their parents, and if the parents have certain correlations between dimensions, so will their children. Note that the p-parents, $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$, are randomly selected from the current population without replacement.

Rotation recombination around the mean

In the following, we propose a rotation recombination where one, some or all parents, $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$, are rotated around the parent's mean, $m = \sum_{j=i}^{i+p-1} x_j^{(t)}/p$, with some angle, not necessarily the same for all parents. We sample the rotation's angles β_i with a normal distribution with mean 0 and the standard deviation equal with the distance between $x_i^{(t)}$ and the parent's mean m, which is $\|x_i^{(t)} - m\|$. Values around 0, both positive and negative, are generated more frequently; as a result we generate more often small rotations, where children are situated in the vicinity of their parents. Note that the rotation proposal distribution adapts the size and the direction of the jump of the child from its main parent. Thus, the size of the jump is given by the distance $\|x_i^{(t)} - m\|$ and the jump is on the spheroid with center in m and the diameter $2 \cdot \|x_i^{(t)} - m\|$. With rotation, we assume that good individuals are in the vicinity of the main parent and on the same spheroid with all the parents. We denote this rotation with S_{rot}.

Proposition 6.9 *S_{rot} is respectful and biases the exploration according to non-linear correlations between dimensions. When we generate a child with S_{rot}, the complexity time is $\mathcal{O}(\ell \cdot p)$, where p is the number of parents and ℓ the number of dimensions. When we generate p children with S_{rot}, the time complexity is also $\mathcal{O}(\ell \cdot p)$. When S_{rot} has $p = \ell + 1$ parents that defines a non-degenerate simplex geometrical shape, S_{rot} is symmetrical.*

Proof. To prove the symmetry of S_{rot}, we use the property that $p = \ell + 1$ children that define a non-degenerate simplex geometrical shape are equally distanced from their mean and determine a unique ℓ-dimensional spheroid with the center in the mean. As a consequence, when rotating one, some or all individuals around the parent's mean, the resulting individual is also on this spheroid. From Proposition 6.8 it follows directly that S_{rot} is respectful, that it exploits non-linear correlations, and that it is linearly scalable with the number of dimensions. When S_{rot} generates one child, the computational complexity is given by the computation of the parent's mean m. It is $\mathcal{O}(\ell \cdot p)$ because the values of m are independently generated on each dimension $m[h] = \sum_{j=i}^{i+p-1} x_j^{(t)}[h]/p$, where $1 \leq h \leq l$. Note that the rotation of a state around another state is only $\mathcal{O}(\ell)$. When p children are generated, the computational effort is $\mathcal{O}(\ell \cdot p)$, or only $\mathcal{O}(\ell)$ per generated child, which is similar to the computational effort for mutation. □

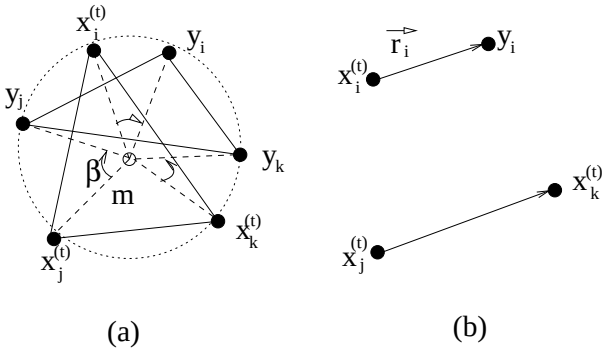

Figure 6.1: (a) Rotation of $\{x_i^{(t)}, x_j^{(t)}, x_k^{(t)}\}$ around the parent's mean m. (b) Differential recombination from $x^{(t)}$ to y_i.

In Figure 6.1 (a), we show an example in 2D space where three children are obtained by rotating the parents with different angles.

Difference recombination

Consider the recombination that generates a child y_i by translating the main parent $x_i^{(t)}$ with a vector $\overrightarrow{r}_i = \sum_{j=i+2}^{i+p-1} \gamma_j \cdot (x_{i+1}^{(t)} - x_j^{(t)})$ where $\gamma_j \in \Omega(x[\cdot])$ can be any statistical variable, for example, sampled from a normal distribution. In Equation 6.2, we have that

$$y_i[h] = x_i^{(t)}[h] + \sum_{j=i+2}^{i+p-1} \gamma_j \cdot (x_{i+1}^{(t)}[h] - x_j^{(t)}[h])$$

Note that $\overrightarrow{r}_i$ is parallel and of the same size with the sum of vectors in-between ℓ parents. We denote this operator with S_{dif}. Again, S_{dif} adapts the size and the direction of the jump of the child from its main parent. The size and the direction of the jump is given by the sum of distances $x_{i+1}^{(t)} - x_j^{(t)}$. With S_{dif}, we assume that good individuals are in the vicinity of the main parent and on a vector parallel with the vectors between the other parents in the family.

Proposition 6.10 *S_{dif} is respectful, symmetrical, and has no correlations between dimensions. The computational effort to generate a child with S_{dif} is $\mathcal{O}(\ell \cdot p)$, with ℓ the number of dimensions and p the number of parents.*

Proof. S_{dif} is linearly scalable with the number of parents since $\overrightarrow{r}_i$ is a sum of ℓ terms and linearly scalable with the number of dimensions since the generation of one allele is independent of the alleles from the other dimensions. From Proposition 6.7, the symmetry and respectfulness directly follows. Since each allele is generated separately, S_{dif} does not exploit the correlations between dimensions. □

For $p = 3$, we obtain the standard differential recombination used by real-coded EMCMCs [87, 86] and by some binary coded EMCMCs [23]. Since the standard differential recombination always uses three parents, it is linearly scalable with the number of dimensions and the computational effort is constant with the number of parents, $\mathcal{O}(\ell)$. In Figure 6.1 (b), we give an example of translation of the parent $x_i^{(t)}$ in a direction parallel with the vector $x_j^{(t)} - x_k^{(t)}$ in 2D space.

Parent centric translation recombination

In the following, we integrate the *parent centric recombination* from real-coded GAs by Deb et al. [18] into real-coded EMCMCs. Consider the translation vector from $x_i^{(t)}$ to the parent's mean, $m = \sum_{j=i}^{i+p-1} x_j^{(t)}/p$, is $\overrightarrow{a}_0 = m - x_i^{(t)}$ and the translation vectors $\overrightarrow{e}_j$, where $i < j < i + p - 1$, that are perpendicular to the vector $\overrightarrow{a}_0$; the direction and length of $\overrightarrow{e}_j$ is given by the vector that starts in $x_i^{(t)}$, where $x_j^{(t)} \neq x_i^{(t)}$, and it is perpendicular to $\overrightarrow{a}_0$. We have that

$$\overrightarrow{r}_i = \gamma_i \cdot \overrightarrow{a}_0 + \gamma_{i+1} \cdot \overrightarrow{e}_{i+1} + \ldots + \gamma_{i+p-2} \cdot \overrightarrow{e}_{i+p-2} \tag{6.4}$$

where we sample the stochastic variables $\{\gamma_i, \ldots, \gamma_{i+p-2}\}$ from normal distributions with mean 0 and non-zero fixed variance; therefore children are situated in the vicinity of their parents.

Note that $(\overrightarrow{a}_0, \overrightarrow{e}_{i+1}, \ldots, \overrightarrow{e}_{i+p-2})$ is not necessarily an orthogonal basis because the vectors $\{\overrightarrow{e}_{i+1}, \ldots, \overrightarrow{e}_{i+p-2}\}$ are not necessarily perpendicular to each other. When the parents $\{x_i^{(t)}, \ldots, x_{i+p-2}^{(t)}\}$ form a non-degenerate simplex, the vectors $\overrightarrow{e}_j$, with $i < j < i + p - 2$, are of the same length and define a non-degenerate $(\ell - 1)$-dimensional simplex shape with center in $x_i^{(t)}$. The mean m is the center of the spheroid on which all the parents are situated, the direction $\overrightarrow{a}_0$ passes through the center of the $(\ell - 1)$-dimensional simplex defined by the parents other than $x_i^{(t)}$. Then the perpendicular from the other parents to $\overrightarrow{a}_0$ are vectors of the same length. It is interesting to note that the simplex defined by $\{\overrightarrow{a}_0, \overrightarrow{e}_{i+1}, \ldots, \overrightarrow{e}_{i+p-2}\}$ is a scaled version of the simplex $\{x_{i+1}^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$.

Because p directions are enough to generate a non-degenerate simplex, we draw perpendiculars on $\overrightarrow{a}_0$ from $p - 2$ parents, $\{x_{i+1}^{(t)}, \ldots, x_{i+p-2}^{(t)}\}$, and not from $x_{i+p-1}^{(t)}$.

This translation proposal distribution adapts the size and the direction of the jump of the child from its main parent. Thus, the size and the direction of the jump is given by the sum of vectors from Equation 6.4. Note that the bigger the distances between the parents the bigger the translation in the search space, because $\overrightarrow{e}_i$ depends on the distance between the parent's mean and the main parent.

When we generate only one individual with a parent centric translation recombination, the resulting proposal distribution is non-symmetrical; the distances between the main and the other parents can be different than the distances between the generated child and the parents that are not the main parent. To obtain symmetrical recombinations, we translate all p parents with the same vector $\overrightarrow{r}_i$. We first choose at random a parent, let us say $x_i^{(t)}$, from which we compute the vector $\overrightarrow{r}_i$. Given $\overrightarrow{r}_i$, we determine the direction, sign and length of the other translation vectors. We denote these symmetrical parent centric translation recombination with S_{pctx}. With S_{pctx}, we assume that good individuals are in the vicinity of their parents and on a direction from the mean to the main parent and the directions perpendicular to it.

Proposition 6.11 *S_{pctx} is respectful, symmetrical and biases the exploration according to the linear correlations between the first and the other dimensions. The complexity time to generate p children with S_{pctx} is $\mathcal{O}(\ell \cdot p)$, with ℓ the number of dimensions and p the number of parents.*

Proof. Note that the values of m are independently generated on each dimension $m[h] = \sum_{j=i}^{i+p-1} x_j^{(t)}[h]/p$, where $1 \geq h \geq l$. S_{pctx} exploits linear correlations between the first and the rest of the dimensions: $\overrightarrow{e}_j$ depends on the parent $x_j^{(t)}$ and its direction is perpendicular to $\overrightarrow{a}_0$ which is determined by the parent $x_i^{(t)}$ and the parent's mean m. Since there is no relationship between two vectors perpendicular to $\overrightarrow{a}_0$,there are no correlations exploited between the other dimensions. S_{pctx} is respectful because, in the dimensions where all the parents are equal, all the component vectors, $\{\overrightarrow{a}_0, \overrightarrow{e}_{i+1}, \ldots, \overrightarrow{e}_{i+p-2}\}$, are 0. Symmetry results directly from the above discussion.

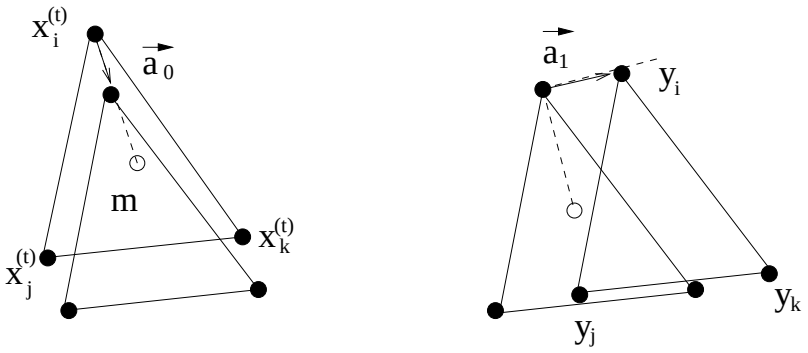

Figure 6.2: A translation of $\{x_i^{(t)}, x_j^{(t)}, x_k^{(t)}\}$ to $\{y_i, y_j, y_k\}$ with S_{pctx}.

The computational effort to compute S_{pctx} is mainly determined by computing the slopes of the perpendicular vectors $\overrightarrow{e}_j$ which is linear with the number of dimensions and number of individuals. Then, to generate p children we need $\mathcal{O}(\ell \cdot p)$ time, or only $\mathcal{O}(\ell)$ per generated child, which is similar to the computational effort for mutation. □

In Figure 6.2, we given an example of S_{pctx} translation of three parents $\{x_i^{(t)}, x_j^{(t)}, x_k^{(t)}\}$ in a two dimensional space. We have two vectors for translation: $\overrightarrow{a}_0$ and $\overrightarrow{e}_{i+1}$. Note that, in this case, the vectors $(\overrightarrow{a}_0, \overrightarrow{a}_1 \equiv \overrightarrow{e}_{i+1})$ form an orthogonal basis.

Scaling recombination

Scaling recombination generates children by modifying the distances between parents. Consider the scaling recombination operator where a candidate individual y_i is generated by translating $x_i^{(t)}$ with a unity vector , $\overrightarrow{r}_i$, with the direction given by the difference between the parents' mean and the main parent, $m - x_i^{(t)}$, and the statistical variable γ_i that is sampled from a normal distribution with mean 0 and fixed non-zero standard deviation. We denote this recombination with S_{scl}. For symmetry, with S_{scl}, we can either generate only one child or we can generate p children. In the last case, we first generate one child y_i by translating the main parent with a vector $\overrightarrow{r}_i$ from $x_i^{(t)}$ to m. Then we generate the other $p-1$ children, $\{y_{i+1}, \ldots, y_{i+p-1}\}$; for that, we translate each parent $x_j^{(t)}$ with a vector $\overrightarrow{r}_j$ from $x_j^{(t)}$ to m, where $\overrightarrow{r}_j$ has the same lenght as $\overrightarrow{r}_i$ and $i+1 \leq j \leq i+p-1$.

Like rotation and the other translations, S_{scl} generates children in the vicinity of their parents. A special scaling operator is the *snooker recombination* proposed by Strens et al. [86] that uses only two parents and $\gamma_i \in \{-1, 1\}$.

Unlike the previous recombinations, the scaling recombination specifies a proposal distribution where only the direction of the jump, and not its size, is adapted from the population. This restriction is the price for its symmetry; because all the parents are involved in translation, the distances between the main parent and the rest of the parents are most likely not the same as the distances between the child and the rest of the parents. A translation that generates one child and also the translation vector's length depends on all the parents, is non-symmetrical; we give such an example in Section 6.2.5. With S_{scl}, we assume again that good children are in the vicinity of their main parent in a direction from the mean to the main parent.

Proposition 6.12 *S_{scl} is symmetrical, respectful, and has no correlations between the dimensions.*

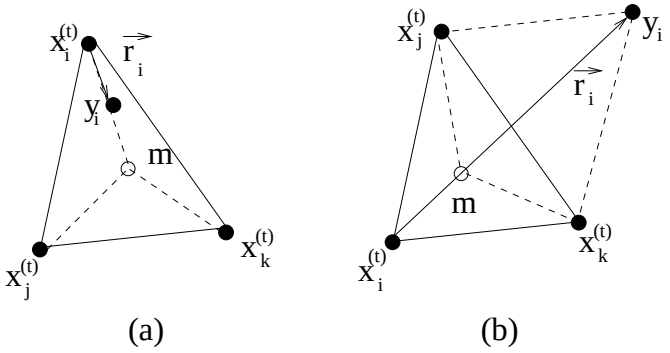

Figure 6.3: (a) Scaling of $x_i^{(t)}$ to y_i with S_{scl}. (b) Reflection of $x_i^{(t)}$ to y_i with S_{ref}.

When S_{scl} generates one or p children, the complexity time is $\mathcal{O}(\ell \cdot p)$, with ℓ the number of dimensions and p the number of parents.

Proof. When we translate y_i on the direction $m - x_i^{(t)}$, the mean of y_i and the rest of the parents, $\{y_i\} \cup \{x_{i+1}^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$, is on the same direction. Since the translation vector's length is sampled from a normal distribution with fixed variance, the recombination is symmetrical. S_{scl} is a particular case of S_{pctx}. Therefore, Proposition 6.11 shows that S_{scl} is respectful, and does not exploit the correlations between dimensions. The computational complexity can also be derived from the correspondence between S_{scl} and S_{pctx}. □

In Figure 6.3 (a), we give an example in 2D space of translation of the parent $x_i^{(t)}$ with the scaling recombination in y_i on the direction $x_i^{(t)} - m$.

Reflection recombination

Consider a translation recombination that translates the main parent through the opposite face of the simplex such that the volume of the simplex is preserved. Then, the main parent is reflected on the face of the $(\ell - 1)$-dimensional simplex that does not contain the main parent. We call this reflection recombination and we denote this recombination with S_{ref}. The reflection recombination was introduced by Strens et al. [86]. In the following, we analyze the properties of and integrate this operator in the simplex geometrical recombinations framework.

The reflected individual is situated on the direction from the main parent $x_i^{(t)}$ to the parent's mean m; on this direction, the distance from $x_i^{(t)}$ to the polygonal face defined by the rest of the parents $\{x_{i+1}^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$ is equal with the distance between the generated individual y_i and $\{x_{i+1}^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$, but on the other side of the polygonal face. Unlike the other recombinations, this does not generate individuals in the vicinity of their parents. This time, we assume that good individuals are situated opposite to the main parent given the geometrical figure defined by $\{x_{i+1}^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$.

Proposition 6.13 *S_{ref} is symmetrical, respectful, and has no correlations between the dimensions of the sampling space. The time complexity to generate a child with S_{ref} is $\mathcal{O}(\ell \cdot p)$, with ℓ the number of dimensions and p the number of parents.*

Proof. S_{ref} is symmetrical since the distance from the main parent to the polygonal face given by the parents other than the main parent is equal to the distance between the child and the same face. The proof of S_{ref}'s scalability becomes obvious if we consider the procedure of Press et al. [72] to build a reflection: for each locus h, we need to compute the value $y_i[h] = 2/l \cdot \sum_{j=i}^{i+p-1} x_j^{(t)}[h] - (2/\ell + 1) \cdot x_i^{(t)}[h]$. The computational effort of S_{ref} is thus $\mathcal{O}(\ell \cdot p)$. □

In Figure 6.3 (b), we give an example in 2D space of reflection of the parent $x_i^{(t)}$ in y_i through the line $x_j^{(t)} - x_k^{(t)}$.

6.2.6 Non-symmetric biased simplex geometrical recombinations

The following recombinations are non-symmetrical but bias the proposal distribution towards the target distribution.

Uphill simplex geometrical recombinations

We integrate the downhill simplex recombination by Nelder and Mead [68], from optimization into the EMCMC framework to obtain efficient recombinations that are biased toward the target distribution. Consider one of the symmetrical simplex recombinations that generates only one child. When using our recombination operators, we assume that we generate better individuals by exploiting the correlation and proximity of the parents. To bias the sampling, we order the parents in increasing fitness value; let us assume that $P'(x_i^{(t)}) \geq \ldots \geq P'(x_{i+p-1}^{(t)})$. Using this order each parent gets a probability to be selected as the main parent: the higher the fitness the more probable it is chosen. For example, $x_i^{(t)}$ is chosen as the main parent with the probability $1/2$, $x_{i+1}^{(t)}$ with probability $1/2^2$, and $x_{i+p-1}^{(t)}$ is chosen with probability $1/2^p$. We assume that in the neighborhood of highly probable states are other highly likely states; we also assume a proposal distribution that generates a child in the neighborhood of the main parent. If the main parent is the most probable from all the parents, then its child has a good chance of also being the most probable state compared to the parents other than the parent. Then, the probability to generate a child from its parents is equal with the reverse probability. In this case, the proposal distribution is very efficient because from good individuals it proposes other good individuals that are accepted with high probability. We call these recombinations *uphill simplex geometrical recombinations* and we generically denote them with S_{up}.

Proposition 6.14 *S_{up} is non-symmetrical and biases the distribution towards the target distribution.*

Proof. Follows directly from the above discussion. □

To calculate the computational time complexity of S_{up}, we have to consider the cost of sorting the candidate main parents. The fastest sorting algorithm (e.g. heap sort) has time complexity $\mathcal{O}(p \cdot \log p)$, where p is the number of parents. The computational effort of the uphill recombinations depends also on the involved simplex recombinations and if the sorting is the most expensive in generating a child or not. For example, with S_{rot}, when a single individual is generated, the computational effort of the uphill recombination is given by rotation, $\mathcal{O}(\ell \cdot p + p \cdot \log p)$.

Uphill parent centric translation recombination

In the following, we propose a non-symmetrical parent centric translation recombination which is "closer" to the target distribution. This time, the translation vectors are computed relative to the center of the mass of the parents $w = (\sum_{j=i}^{i+p-1} P'(x_j^{(t)}) \cdot x_j^{(t)})/(\sum_{j=i}^{i+p-1} P'(x_j^{(t)}))$. We now have $\overrightarrow{a}_0 = w - x_i^{(t)}$. Like for S_{pctx}, we also sample in the perpendicular directions, $\overrightarrow{e}_j$, from $x_j^{(t)}$ to

$\overrightarrow{a}_0$, with $i < j < p-1$. Note that w is closer to high probable individuals than to less probable ones. This translation is generating small jumps for probable individuals and large jumps for less probable individuals. Then, the probability to remain around good individuals is larger than around other states, a property that makes this distribution resemble the target distribution. We denote this operator with S_{wgh}. The probability to generate an individual y_i from its parents $\{x_i^{(t)}, \ldots, x_{i+p-2}^{(t)}\}$ with S_{wgh} is

$$\frac{1}{\sigma_i \cdot \sqrt{2 \cdot \pi}} \cdot \exp\left(-\frac{\|\gamma_i \cdot \overrightarrow{a}_0\|^2}{2 \cdot \sigma_i^2}\right) \cdot \prod_{j=i+1}^{i+p-2} \frac{1}{\sigma_j \cdot \sqrt{2 \cdot \pi}} \cdot \exp\left(-\frac{\|\gamma_j \cdot \overrightarrow{e}_j\|^2}{2 \cdot \sigma_j^2}\right)$$

where $\{\sigma_i, \ldots, \sigma_{i+p-2}\}$ are the standard deviations for the directions $\{\overrightarrow{a}_0, \overrightarrow{e}_{i+1}, \ldots, \overrightarrow{e}_{i+p-2}\}$, respectively, and are set apriori. We observe that even when the distances between parents would be preserved, the translation vector $\overrightarrow{r}_i$ depends on the values of the parents $\{x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$ that can be different for $\{y_i\} \cup \{x_{i+1}^{(t)}, \ldots, x_{i+p-1}^{(t)}\}$. Thus, this recombination is not symmetrical even when we would generate p children simultaneously as with S_{pctx}.

Proposition 6.15 *S_{wgh} is non-symmetrical, but is biased towards the target distribution, and is not linearly scalable.*

Proof. The non-symmetry of S_{wgh} results from the above discussion. The probability to remain near the "good" state is larger with S_{wgh} than for the other states. That makes S_{wgh} to resemble more the target distribution than the uniform distribution. S_{wgh} is not linearly scalable because $\{\overrightarrow{a}_0, \overrightarrow{e}_{i+1}, \ldots, \overrightarrow{e}_{i+p-2}\}$ is not necessarily an orthogonal basis. Then, we cannot uniquely compute the probability to generate the main parent from its child and the other parents. Therefore, S_{wgh} is not linearly scalable. □

6.3 Irreducible recombinative proposal distributions

In the previous sections, we have proposed and analyzed several recombination operators. In this section we study the properties of the various combinations of proposal distributions. Our goal is to obtain operators that are irreducible. We further want that the complex proposal distributions inherit from the component distributions some desirable properties like symmetry. In the following, to obtain irreducible proposal distributions, we combine recombination with mutation - which is irreducible.

6.3.1 Mixtures, cycles, sums

In EMCMCs, like for transition distributions, the proposal distributions are combined following some simple mathematical rules in *mixtures*, a probabilistic sum of distribution where only one distribution is used at a time, and *cycles*, a product of proposal distributions. We propose the *sum of proposal distributions* that is an addition of proposal distributions that are used in the same time.

Definition 6.3 *A* mixture of proposal distributions *is a probabilistic sum of proposal distributions where at each step one distribution is selected according to some constant positive probability. A* cycle of proposal distributions *is the product of proposal distributions where in each step each distribution is used in turn, and when the last distribution is used, the cycle is restarted. A* sum of proposal distributions *is the proposal distributions where each step the effect of all distributions are summed.*

Note that sums, like cycles, use all distributions at once, but, unlike cycles that apply them sequentially - the output of one distribution is the input for another distribution in the cycle - the sum generates individuals with each distribution in parallel and sums the effect of proposed individuals. Like mixtures, sums are adding the effect of one distribution to another one, but unlike mixtures, each step, *all* the distributions are added to generate the current individuals rather than using a distribution at a time.

Mixture

Mixtures of proposal distributions have similar properties as mixtures of transition distributions and they are used by most of the recombinative EMCMCs [23, 63].

Theorem 6.3 *In a mixture of proposal distributions, if one distribution is irreducible, then the mixture is irreducible. If, in a mixture, all the component distributions are symmetrical, then the mixture is symmetrical. The mixture $S_{m\oplus r}$ is biased toward the target distribution if the recombination S_r is. The mixture $S_{m\oplus r}$ biases the exploration with the same type of correlation between dimensions as the recombination S_r does.*

Proof. If one component distribution is irreducible, than we can go from one state to any other state with the mixture. Symmetry results from commutativity of addition and symmetry of components. In the absence of acceptance, the target distribution of generating states with mutation is uniformly distributed, since all the states are generated with the same probability. If the recombination is biased towards the target distribution, so does a mixture of mutation with recombination. The normal distribution does not exploit the correlation between dimensions. If the recombination operator exploits some correlation, the same correlations are exploited with a mixture of mutation with recombination. □

For example, the following mixture

$$S_{m\oplus r} = p_m \cdot S_m \oplus (1 - p_m) \cdot S_r$$

is irreducible when $p_m > 0$, and symmetrical when the recombination is symmetrical. Note that $S_{m\oplus r}$ is equivalent with recombination if $p_m = 0$; then $S_{m\oplus r} = S_r$ is reducible if the recombination is reducible. The mixture $S_{m\oplus wgh}$ is biased toward the target distribution because the recombination S_{wgh} is biased. The mixture $S_{m\oplus rot}$ is exploiting some non-linear correlation between dimensions since S_{rot} does.

Cycles

Unlike for mixtures, for cycles, there are no general rules for irreducibility or symmetry. They have to be checked for each cycle. In general, since the product of two matrices usually does not commute, the cycle of two proposal distributions are non-symmetrical, although the proposal distributions in themselves are symmetrical. Cycles of mutation and recombination proposal distributions are common for the standard GAs. For example

$$S_{m\times r} = S_r \times S_m \quad ; \quad S_{r\times m} = S_m \times S_r$$

Proposition 6.16 *If the recombination S_r is symmetrical, we have*

$$S_{m\times r}(Y \mid X^{(t)}) = S_{r\times m}(X^{(t)} \mid Y)$$

Proof. When S_r is symmetric, the proposition holds since

$$S_{m\times r}(Y \mid X^{(t)}) = \sum_{Y'\in\Omega(X)} S_m(Y' \mid X^{(t)}) \cdot S_r(Y \mid Y') =$$

$$\sum_{Y'\in\Omega(X)} S_r(Y' \mid Y) \cdot S_m(X^{(t)} \mid Y') = S_{r\times m}(X^{(t)} \mid Y)$$

□

In the following proposition, we show the properties of some cycles in the discrete space.

Proposition 6.17 *Consider the uniform mutation and one of the recombinations for discrete spaces S_{unif}, S_{dif}, S_{mask} and S_{tree}. $S_{m\times r}$ and $S_{r\times m}$ are irreducible, stationary and bias the exploration with the same type of correlation as the recombination does. $S_{m\times r}$ and $S_{r\times m}$, are symmetrical for any recombination that swaps alleles [65]. $S_{m\times dif}$ and $S_{dif\times m}$ are non-symmetrical.*

Proof. $S_{r\times m}$ and $S_{m\times r}$ are symmetrical for recombinations that swap alleles because mutation generates the alleles which differ in the two populations and recombination swaps them or vice-versa.

By means of an example, we proof that $S_{dif\times m}$ is non-symmetrical. Consider the current population of bits $X^{(t)} = \{0,1,0\}$ and the candidate population $Y = \{1,1,1\}$, the mutation rate of $1/3$, and, for simplicity, the xor operator. We compute the probability to generate Y from $X^{(t)}$ with uniform mutation and then with xor recombination and the inverse probability to generate $X^{(t)}$ from Y.

Let's consider all possible parent choices for xor. With the xor recombination, given the distance $\Delta(0,1)$ between the first two bits, we generate 1 from the third bit of the current population 0; the intermediate population is now $Y' = \{0,1,1\}$. The distance between the second and the third bit is also $\Delta(1,0)$, and thus the intermediate population is again $Y' = \{1,1,0\}$. Since the distance between first and second bits of the current population is $\Delta(0,0)$, we generate 1 from 1 and the intermediate population is $Y' = \{0,1,0\}$. When we mutate the intermediate populations, we have $S_m(1,1,1 \mid 0,1,0) = (1/3)^2 \cdot 2/3$ and $S_m(1,1,1 \mid 1,1,0) = (2/3)^2 \cdot 1/3$. Computing in a similar manner the inverse probability, for all possible intermediate populations, we have $S_{dif\times m}(Y \mid X^{(t)}) = 1/3 \cdot ((1/3)^2 \cdot 2/3 + 2 \cdot (2/3)^2 \cdot 1/3) = 10/81$.

To generate $X^{(t)}$ from Y with $S_{dif\times m}$, we mutate Y to $Y' = \{0,1,1\}$ and then swap with the xor operator the last bit given the difference between the first two bits resulting in $X^{(t)}$. Similarly, we mutate Y to $Y' = \{1,1,0\}$ and swap the first bit of Y' or we mutate into $Y' = Y$ and do not swap the middle bit with xor since the difference between the first and the last bit is 0. We then have $S_{dif\times m}(X^{(t)} \mid Y) = 1/3 \cdot (2 \cdot (2/3)^2 \cdot 1/3 + 1/3 \cdot (2/3)^2) = 3/81$.

We conclude that $S_{dif\times m}$ is not symmetrical since $S_{dif\times m}(X^{(t)} \mid Y) \neq S_{dif\times m}(Y \mid X^{(t)})$. □

Parallel Recombinative Simulated Annealing (PRSA) [65] uses recombination that swaps alleles followed by mutation.

In the following proposition, we show the properties of some cycles in the real-coded space.

Proposition 6.18 *Consider the cycles where mutation is combined with one of the symmetric simplex geometrical recombinations. These cycles are irreducible and bias the exploration with the same type of correlation as the recombination does. When combining S_{rot}, S_{pctx}, S_{scl} and S_{ref} with mutation, the resulting cycles are not symmetric.*

Proof. $S_{m\times rot}$, $S_{m\times pctx}$, $S_{m\times scl}$ and $S_{m\times ref}$ are not symmetric since the distances, directions and volumes of the parents are not kept constant with mutation. The rest of the properties follow directly. □

Note that it is impractical to use a non-symmetrical cycle: to compute its probabilities, we have to sum over all possible intermediate populations.

Sums

The properties of sums are similar with the ones of mixtures for symmetry, but, like for cycles, there are no general rules for irreducibility; a sum of any two irreducible distributions (e.g. a distribution and the same distribution negated) is not necessarily another irreducible distribution.

Theorem 6.4 *A sum of symmetrical proposal distributions is symmetrical.*

Proof. The sum over real-valued distributions (matrices) is commutative. □

Recall from Section 6.1.2 that we cannot use the masked recombination, S_{mask}, with an MH algorithm: when a non-common bit for the parent and the mask is flipped, there is a 0 probability to flip that bit back using the generated child and the same mask. As a consequence, the MH algorithm would be reducible. In the next paragraph, we propose an irreducible proposal distribution that sums S_{mask} and uniform recombination. Consider a parent $x_i^{(t)}$ and a mask $x_{i+1}^{(t)}$ chosen at random from the population. Like for S_{mask}, for the non-common values of the two parents, $x_i^{(t)}$ is flipped with the probability p_x to generate the child y_i. Unlike for S_{mask}, for the common parts of these parents, $x_i^{(t)}$ is flipped with the low probability $1/\ell$ to generate the child y_i. We generate from the mask $x_{i+1}^{(t)}$ a second child y_{i+1} with the uniform mutation with the mutation rate p_m. We denote this proposal distribution with S_{m+mask} where

$$S_{m+mask}(y_i, y_{i+1} \mid x_i^{(t)}, x_{i+1}^{(t)}) = S_{mask}(y_i \mid x_i^{(t)}, x_{i+1}^{(t)}) + S_m(y_{i+1} \mid x_{i+1}^{(t)})$$

In the next proposition we show that S_{m+mask}, unlike S_{mask}, can be used with an MH algorithm. Furthermore, although it is a cycle, its computational time is similar with the one of uniform mutation.

Proposition 6.19 *S_{m+mask} is irreducible, has no correlations between dimensions and has positive non-zero values, $S_{m+mask}(\cdot \mid \cdot) > 0$. S_{m+mask} is symmetrical if $p_m = 1/2$ or $p_x = 1/\ell$. If $p_m \neq 1/2$ and $p_x \neq 1/\ell$ then S_{m+mask} is non-symmetrical. The time complexity to generate two children from two parents with S_{m+mask} is linear with the string size ℓ, $\mathcal{O}(\ell)$.*

Proof. S_{m+mask} is irreducible, since is has $S_{m+mask}(\cdot \mid \cdot) > 0$. If $p_x = 1/\ell$, the S_{m+mask} is equivalent with the mutation operator, since all alleles in the parents can be flipped with the probability $1/\ell$. Then S_{m+mask} is symmetric. For $p_m = 1/2$, we uniform randomly generate the child y_{i+1} from the mask $x_{i+1}^{(t)}$, and the uniform randomly probability to generate the mask from y_{i+1}. Then, S_{m+mask} is symmetric since the common and uncommon parts of the parents and the children are randomly generated.

By means of an example, we show that S_{m+mask} is non-symmetrical for other values of p_m and p_x. Consider $x_i^{(t)} = x_{i+1}^{(t)} = 0$ and $\{y_i, y_{i+1}\} = \{1, 0\}$. When $y_i = 1$ and $y_{i+1} = 0$, the probability to generate y_i is $1/\ell$, and the probability to generate y_{i+1} is $1 - p_m$. The inverse probability is $S_{m+mask}(x_i^{(t)}, x_{i+1}^{(t)} \mid y_i, y_{i+1}) = (1 - p_x) \cdot (1 - p_m)$. When $y_i = 0$ and $y_{i+1} = 1$, the probability to generate y_i is $1 - 1/\ell$ and the probability to generate y_{i+1} is p_m. The reverse probability is now $p_x \cdot p_m$. Then $S_{m+mask}(y_i, y_{i+1} \mid x_i^{(t)}, x_{i+1}^{(t)}) = (1-p_m)/\ell + (1-1/\ell) \cdot p_m$ and $S_{m+mask}(x_i^{(t)}, x_{i+1}^{(t)} \mid y_i, y_{i+1}) = (1-p_x) \cdot (1-p_m) + p_x \cdot p_m$. We now have that if $p_x \neq 1/\ell$ and $p_m \neq 1/2$, then S_{m+mask} is non-symmetrical. □

Similarly, we combine the tree frequencies probabilistic recombination, S_{tree}, with the uniform mutation in a sum in order to use it with the MH algorithm. We first construct the maximum entropy tree. We choose at random a position, h, which we consider the root, we propose an allele $y_i[h]$ with the probability $(N(y_i[h]) + 1)/(N + |\Omega(x[\cdot])|)$. Iteratively, we propose an allele $y_i[h]$ with the probability

$$(N(y_i[h], y_i[h1]) + 1)/(N(y_i[h1])) + |\Omega(x[\cdot])|)$$

where the allele on the position $h1$, $y_i[h1]$, is already instantiated. We denote this operator with $S_{m+tree} = (S_{tree} + 1/N)/(1 + \Omega(x[\cdot])/N)$. Like S_{tree} and unlike the other proposal distribution, S_{m+tree} exploits some relationships between different dimensions.

Proposition 6.20 *S_{m+tree} is irreducible, non-symmetrical and biases the exploration according to non-linear correlations between the dimensions. The time complexity to generate an individual with S_{m+tree} is $\mathcal{O}(\ell^2 \cdot N)$, where ℓ is the string size and N the population size.*

Proof. The proof is immediate. □

In the real-space, let us consider the sum S_{m+dif} between a uniform mutation and the differential recombination S_{dif}. An individual y_i is generated with S_{m+dif} from the main parent $x_i^{(t)}$, and the parents set $\{x_j^{(t)}, x_k^{(t)}\}$ such that $y_i = x_i^{(t)} + \gamma_i \cdot (x_j^{(t)} - x_k^{(t)}) + \gamma_m$. We consider that ter Braak [87]'s differential proposal distribution is a sum, on each locus, between differential recombination and uniform mutation - that is a random variable sampled from a uniform distribution. S_{m+dif} is irreducible, symmetrical, but does not exploit the correlations between dimensions.

Proposition 6.21 *Consider the sum between the normal mutation and one of the simplex geometrical recombinations*

$$S_{m+r} = S_m + S_r \quad ; \quad S_{r+m} = S_r + S_m$$

These sums are symmetrical, when the recombination is symmetrical, irreducible, biased when one of the recombinations is biased, and exploits the correlations between dimensions when one of the recombinations does.

Proof. These sums are irreducible because we can arrive from any population to any other population. The rest of the properties follow directly. □

6.3.2 Restricted position sums

In the following, we propose a particular case of sums that generate with each component proposal distribution different (parts of) individuals. These sums are computationally efficient and linearly scalable, even when the component distributions are non-symmetrical.

Definition 6.4 *A* sum on exclusive positions in a population *is a sum that generates with each proposal distribution different positions in a population.*

The proposal distribution that generates exclusive positions in a population does not affect the positions that they do not generate. Thus, to compute the probabilities in this sum, we have to compute only the probabilities of the component proposal distributions. For example, with mutation we generate an individual y_k in the population and the rest of p children we generate with recombination

$$S_{m+r}(y_i, \ldots, y_{i+p-1} \mid x_i^{(t)}, \ldots, x_{i+p-1}^{(t)}) =$$

$$S_m(y_i \mid x_i^{(t)}) + S_r(y_{i+1}, \ldots, y_{i+p-1} \mid x_i^{(t)}, \ldots, x_{i+p-1}^{(t)})$$

Under certain conditions, these exclusive position sums have similar properties as the mixture of distributions.

Proposition 6.22 *Consider a sum on exclusive positions where the position to be generated with each proposal distribution are picked at random. If each proposal distribution is irreducible, then this sum is irreducible. This sum is symmetrical if the component distributions are symmetrical.*

Proof. Since the irreducible proposal distribution goes in several steps from any to any individual, so does this sum. The rest of the properties follows directly. □

Restricted position sums for MH algorithms

Since, in an EMCMC, the unnormalized proposal distribution, $P'_i(\cdot)$, can be evaluated at individual level (e.g. for an individual $x_i^{(t)}$), but, usually, cannot be evaluated at allele level (e.g. given an allele $x_i^{(t)}[h]$), we propose sums on exclusive individuals of transition distributions.

Definition 6.5 *A* sum on exclusive individuals in a population *is a combination of transition distributions such that each candidate individual is proposed and accepted with exactly one transition probability.*

The transition distribution that generates and accepts exclusive individuals in a population, does not affect the positions that they do not generate. Like mixtures of transition distributions, this sum inherits its properties from the component transition distributions.

Theorem 6.5 *If each transition distribution is irreducible, then the sum on exclusive individuals is irreducible. This sum has detailed balance if the component distributions have detailed balance.*

Proof. This sum is irreducible since from a population we can arrive into any other population with a non-zero probability. Since we sum over the component distributions, when the components have detailed balance, the sum has detailed balance. □

6.3.3 Combination of mixtures, cycles and sums

Based on mixtures, cycles, and sums one can create various other ones. For example, the components of a mixture can be cycles, or we can use a cycle in a cycle.

$$S_{m\times r\oplus r\times m} = p_m \cdot S_{m\times r} \oplus (1-p_m)\cdot S_{r\times m} \;\;;\;\; S_{m\times r\times m} = S_{r\times m} \times S_m \;\;;\;\; S_{r\times m\times r} = S_{m\times r} \times S_r$$

Proposition 6.23 *$S_{m\times r\oplus r\times m}$ is irreducible if $p_m > 0$. It is also symmetrical if recombination is symmetrical and $p_m = 0.5$. $S_{m\times r\times m}$ and $S_{r\times m\times r}$ are irreducible and symmetric if recombination is symmetric.*

Proof. From Proposition 6.18, if recombination is symmetrical then $S_{m\times r}(Y \mid X^{(t)}) = S_{r\times m}(X^{(t)} \mid Y)$. If $p_m = 0.5$,

$$S_{m\times r\oplus r\times m}(Y \mid X^{(t)}) = 0.5 \cdot S_{m\times r}(Y \mid X^{(t)}) \oplus 0.5 \cdot S_{r\times m}(Y \mid X^{(t)}) = S_{m\times r\oplus r\times m}(X^{(t)} \mid Y)$$

We also have

$$S_{m\times r\times m}(Y \mid X^{(t)}) = \sum_{Y',Y''\in\Omega(X)} S_m(Y' \mid X^{(t)}) \cdot S_r(Y'' \mid Y') \cdot S_m(Y \mid Y'') =$$

$$\sum_{Y',Y''\in\Omega(X)} S_m(X^{(t)} \mid Y') \cdot S_r(Y' \mid Y'') \cdot S_m(Y'' \mid Y) = S_{m\times r\times m}(X^{(t)} \mid Y)$$

The other properties follow directly. □

Chapter 7

Acceptance rules

In the previous chapter we have shown how to combine recombinative proposal distributions with mutation to obtain irreducible recombinative proposal distributions. In this chapter, we study the interaction between recombination and various MH acceptance rules and discuss how to obtain EMCMCs with specific properties, such as detailed balance.

First, we investigate the properties of the EMCMC algorithm resulting from a recombinative proposal distribution and the standard MH acceptance rule. In Section 7.2, we show that to obtain detailed balance we need an MH acceptance rule where all children created by a recombination operator are either all accepted or all rejected. In Section 7.3, we study some MH acceptance rules for optimization.

7.1 The standard MH acceptance rule in (recombinative) EMCMCs

In the following, we investigate the properties of non-recombinative and recombinative EMCMCs that use the standard MH acceptance rule.

7.1.1 Multiple independent chains (MICs)

In an attempt to improve the mixing behavior of MCMCs one could make use of multiple chains that run independently (MICs). The chains are started at different initial states and their output is observed at the same time. It is hoped that this way a more reliable sampling of the target distribution $P(\cdot)$ is obtained. It is important to note that no information exchange between the chains takes place.

Recommendations in the literature are conflicting regarding the efficiency of multiple independent chains. Consider that the number of individuals generated with a long MCMC and with a MIC are the same. Consider also that a MIC and an MCMC start their chains from randomly generated states since we do not have any a priori information about the search space. Then, there is no guarantee that these initial states belong to relevant parts of the search space, therefore there is also no guarantee that a MIC will perform better than a well mixing, single MCMC chain.

Yet there are at least theoretical advantages of multiple independent chains MCMC for establishing its convergence to $P(\cdot)$ [33]. For example, let's consider a large dimensional distribution where a MCMC takes a long time to find a relevant region of the search space and to escape from it to search

for other relevant regions. Then, the time necessary for a long MCMC can be larger than just starting multiple MCMCs spread over the search space sampling in different regions.

Since the chains do not interact, MIC is at population level an MCMC with transition probabilities equal to the product of component chains transition probabilities, or

$$T(X^{(t+1)} \mid X^{(t)}) = \prod_{i=1}^{N} T(x_i^{(t+1)} \mid x_i^{(t)})$$

where $X^{(t+1)}$ is the next population. If the MCMCs have detailed balance, are irreducible and aperiodic, then MIC inherits these properties and it converges, at population level, to the product of their target distributions, $\prod_{i=1}^{N} P_i(\cdot)$, where $P_i(\cdot)$ is the target distribution of the i-th chain.

7.1.2 One parent against one child in recombinative EMCMCs

In general, an MH algorithm that accepts with the standard MH acceptance rule the individuals generated with some recombinative proposal distribution does not have detailed balance. Unfortunately, if the detailed balance condition does not hold, there is no standard method to determine the target distribution. In the following, we show that, under certain conditions, the marginal distribution over the samples of a chain is $P(\cdot)$.

Theorem 7.1 *Consider the MH algorithm that proposes p children from p parents using an irreducible proposal distribution S, and each child competes against one of the parents, randomly chosen without replacement from the parents set.*

This algorithm has detailed balance if and only if the probability to generate the children from the parents is equal with the probability to generate some children and some parents from the rest of the parents and children. Then, the algorithm converges to the product of distributions $\prod_{i=1}^{p} P(\cdot)$.

If the proposal distribution is independent of the target distribution then the individual states from a chain in the population converge to the target distribution $P(\cdot)$.

Proof. We split our prove in three parts. First we prove that, in general, this MH algorithm does not have detailed balance. Then, we prove that the target distribution of the samples from a chain in a population is $R(\cdot)$. Last, we show that, under certain conditions, $R(\cdot) = P(\cdot)$.

For ease of exposure and without loss of generality, let's consider populations of two individuals and two children $\{y_1, y_2\}$ that are generated with some irreducible and symmetrical proposal distribution S from two parents $\{x_1, x_2\}$. Then $S(y_1, y_2 \mid x_1, x_2) = S(x_1, x_2 \mid y_1, y_2)$. Let's assume that the child y_1 competes against the parent x_1 in the standard Metropolis acceptance rule $A(y_1 \mid x_1) = \min\{1, P'(y_1)/P'(x_1)\}$, and the child y_2 competes against the parent x_2 in $A(y_2 \mid x_2) = \min\{1, P'(y_2)/P'(x_2)\}$. We denote with $T_{1.1}$ the resulting transition matrix.

1. Proof: *this MH algorithm does not have detailed balance.* If both parents are different from their children $x_1 \neq y_1$ and $x_2 \neq y_2$, the transition probability is

$$T_{1.1}(y_1, y_2 \mid x_1, x_2) = S(y_1, y_2 \mid x_1, x_2) \cdot A(y_1 \mid x_1) \cdot A(y_2 \mid x_2)$$

If one parent and one child are equal, let's say $x_1 = y_1$, the transition probability $T_{1.1}(x_1, y_2 \mid x_1, x_2)$ is equal with the transition probability where both children $\{x_1, y_2\}$ are accepted plus the sum where one less fit individual y' is proposed and rejected, where $P'(y') < P'(x_1)$. Then,

$$T_{1.1}(x_1, y_2 \mid x_1, x_2) = S(x_1, y_2 \mid x_1, x_2) \cdot A(x_1 \mid x_1) \cdot A(y_2 \mid x_2)$$
$$+ \sum_{y' \mid P'(y') < P'(x_1)} S(y', y_2 \mid x_1, x_2) \cdot [1 - A(y' \mid x_1)] \cdot A(y_2 \mid x_2)$$

If both parents and children are equal, $\{y_1, y_2\} = \{x_1, x_2\}$, the transition probability is

$$T_{1.1}(x_1, x_2 \mid x_1, x_2) = 1 - \sum_{\{y_1,y_2\} \neq \{x_1,x_2\}} T_{1.1}(y_1, y_2 \mid x_1, x_2)$$

From the last equality we have that $T_{1.1}$ is stationary. Since $T_{1.1}$ is a stationary Metropolis algorithm, that is irreducible and aperiodic when the proposal distribution is irreducible, the algorithm converges to a stationary distribution. In the following, we show that $T_{1.1}$ does not have detailed balance unless $S(x_1, x_2 \mid y_1, y_2) = S(x_1, y_2 \mid y_1, x_2)$, for any x_1, x_2, y_1, y_2. We denote with $R(\cdot, \cdot)$ the target distribution for this algorithm.

If $x_1 \neq y_1$ and $x_2 \neq y_2$ and both children are accepted or rejected, the detailed balance condition holds for any proposal distribution S. It is straightforward to show that

$$R(x_1, x_2) \cdot S(y_1, y_2 \mid x_1, x_2) \cdot A(y_1 \mid x_1) \cdot A(y_2 \mid x_2) =$$

$$R(y_1, y_2) \cdot S(x_1, x_2 \mid y_1, y_2) \cdot A(x_1 \mid y_1) \cdot A(x_2 \mid y_2)$$

if $R(x_1, x_2) = P(x_1) \cdot P(x_2)$ for any x_1, x_2.

If one child is rejected, for example $x_1 = y_1$, and if $R(x_1, x_2) = P(x_1) \cdot P(x_2)$ for any x_1, x_2, we have detailed balance for the first terms of $T(x_1, y_2 \mid x_1, x_2)$, and the inverse transition probability, $T(x_1, x_2 \mid x_1, y_2)$ since

$$R(x_1, x_2) \cdot S(x_1, y_2 \mid x_1, x_2) \cdot A(x_1 \mid x_1) \cdot A(y_2 \mid x_2) =$$

$$R(x_1, y_2) \cdot S(x_1, x_2 \mid x_1, y_2) \cdot A(x_1 \mid x_1) \cdot A(x_2 \mid y_2)$$

We further group the terms with the same y' from the direct, $T(x_1, y_2 \mid x_1, x_2)$, and the inverse, $T(x_1, x_2 \mid x_1, y_2)$, transition probabilities. The detailed balance condition holds if, for all of them, we have that

$$R(x_1, x_2) \cdot S(y', y_2 \mid x_1, x_2) \cdot [1 - A(y' \mid x_1)] \cdot A(y_2 \mid x_2) =$$

$$R(x_1, y_2) \cdot S(y', x_2 \mid x_1, y_2) \cdot [1 - A(y' \mid x_1)] \cdot A(x_2 \mid y_2)$$

This is true when $S(y', y_2 \mid x_1, x_2) = S(y', x_2 \mid x_1, y_2)$ for any states $\{y', y_2, x_1, x_2\}$ and, again, $R(\cdot, \cdot) = P(\cdot) \cdot P(\cdot)$.

We conclude that, for this algorithm, the detailed balance condition holds if and only if $S(x_1, x_2 \mid y_1, y_2) = S(y_1, x_2 \mid x_1, y_2)$, for any choice of x_1, x_2, y_1, y_2. In the sequel, when the detailed balance holds, the MH algorithm converges to the distribution $P(\cdot) \cdot P(\cdot)$.

2. Proof: *the samples from the first chain converge to some distribution* $R(\cdot)$.

The marginal transition probability to generate x_1 from y_1 when summing over the variables of the second chain is

$$T_{1.1}(y_1 \mid x_1) = \sum_{x_2, y_2} R(x_2) \cdot T_{1.1}(y_1, y_2 \mid x_1, x_2)$$

We can rewrite the marginal probability $T_{1.1}(y_1 \mid x_1)$ as the sum over x_2 and y_2 of transition probabilities where both children are accepted $T_{1.1}(y_1, y_2 \mid x_1, x_2)$ plus the sum of transition probabilities where y_2 is rejected. Given the definition of $T_{1.1}(y_1, y_2 \mid x_1, x_2)$, and the fact that we sum over all x_2, y_2, we have that

$$T_{1.1}(y_1 \,|\, x_1) = \sum_{x_2, y_2} R(x_2) \cdot S(y_1, y_2 \,|\, x_1, x_2) \cdot A(y_1 \,|\, x_1) \cdot A(y_2 \,|\, x_2)$$

$$+ \sum_{x_2,y_2} R(x_2) \cdot S(y_1, y_2 \mid x_1, x_2) \cdot A(y_1 \,|\, x_1) \cdot [1 - A(y_2 \,|\, x_2)]$$

We group the terms and simplify the above equation to

$$T_{1.1}(y_1 \,|\, x_1) = A(y_1 \,|\, x_1) \cdot (\sum_{x_2,y_2} R(x_2) \cdot S(y_1, y_2 \,|\, x_1, x_2))$$

Let's denote

$$S'(y_1 \mid x_1) = \sum_{x_2,y_2} R(x_2) \cdot S(y_1, y_2 \,|\, x_1, x_2)$$

$S'(y_1 \mid x_1)$ is a number that depends on y_1 and x_1 but is independent of x_2 and y_2. We further rewrite the marginal transition probability

$$T_{1.1}(y_1 \,|\, x_1) = A(y_1 \,|\, x_1) \cdot S'(y_1 \,|\, x_1)$$

We have that $T_{1.1}(y_1 \mid x_1)$ is stationary since $T_{1.1}$ is stationary; then, $\sum_{y_1} T_{1.1}(y_1 \mid x_1) = \sum_{x_2} R(x_2) \sum_{y_1,y_2} T_{1.1}(y_1, y_2 \,|\, x_1, x_2) = 1$.

We conclude that the marginal target distribution of the first chain is $R(\cdot)$.

3. Proof: *$R(\cdot) = P(\cdot)$ if P is independent of S.*

Let's consider, without loss of generality, that $P(x_1) > P(y_1)$. Then, $A(y_1 \mid x_1) = P(y_1)/P(x_1)$ and $A(x_1 \mid y_1) = 1$. The detailed balance condition for the marginal transition distribution is

$$\frac{P(x_1)}{P(y_1)} = \frac{R(x_1)}{R(y_1)} \cdot \frac{S'(y_1 \mid x_1)}{S'(x_1 \mid y_1)}$$

We show that $S'(y_1 \mid x_1) = S'(x_1 \mid y_1)$ if P is independent of S.

Let's denote $S(y_1 \mid x_1, x_2) = \sum_{y_2} S(y_1, y_2 \mid x_1, x_2)$. The expected value of the above distribution $S(y_1 \mid x_1, \cdot)$ is $S(y_1 \mid x_1)$ and the expected value of $P(\cdot)$ is some number μ. The covariance between P and $S(y_1 \mid x_1)$ is

$$\sum_{x_2} (P(x_2) - \mu) \cdot (S(y_1 \mid x_1, x_2) - S(y_1 \mid x_1)) = (\sum_{x_2} P(x_2) \cdot S(y_1 \mid x_1, x_2)) - \mu \cdot S(y_1 \mid x_1)$$

We know that the covariance between the $P(\cdot)$ and $S(y_1 \mid x_1)$ is equal with the covariance between $P(\cdot)$ and $S(x_1 \mid y_1)$, since the two distributions $P(\cdot)$ and $S(\cdot \mid \cdot)$ are independent of each other. Since S is symmetrical, we have that $S(x_1 \mid y_1) = S(y_1 \mid x_1)$. We further find that $\sum_{x_2} P(x_2) \cdot S(y_1 \mid x_1, x_2) = \sum_{x_2} P(x_2) \cdot S(x_1 \mid y_1, x_2)$, and, thus, $S'(y_1 \mid x_1) = S'(x_1 \mid y_1)$. This concludes our proof. □

According to the above theorem, the detailed balance condition holds for uniform mutation distribution S_m and symmetrical recombination distributions that generate one child (e.g. S_{dif}), but does not hold for other symmetrical recombinations (e.g. S_{unif} and S_{m+mask}). For any four individuals, $S_{unif}(y_1, y_2 \mid x_1, x_2) \neq S_{unif}(y_1, x_2 \mid x_1, y_2)$.

Consider a non-symmetrical proposal distribution S that generates two children y_1 and y_2 from two parents $x_1^{(t)}$ and $x_2^{(t)}$. To use non-symmetrical proposal distributions, we need to weigh the fraction $P'(y_1)/P'(x_1^{(t)})$ with a correction term, $S_{x_2^{(t)}}(x_1^{(t)} \mid y_1, y_2)/S_{y_2}(y_1 \mid x_1^{(t)}, x_2^{(t)})$, that compensates the non-symmetry of the proposal distribution as in the standard acceptance rule. We denote

alg	transition	DB	dist/optim
MCMC [45, 66]	$T_{m,u} = S_{m,u} \cdot A$, $T_{m,n} = S_{m,n} \cdot A$	yes	$P(\cdot)$
MIC	$T_{m,u}$, $T_{m,n}$	yes	$\prod_{i=1}^{N} P(\cdot)$
rEMCMC	$T_{1.1} = S_{m,u\times unif} \cdot A$	no	$P(\cdot)$
aEMCMC	$0.5 \cdot T_{m,n} \oplus 0.5 \cdot (0.25 \cdot S_{rot} \oplus 0.25 \cdot S_{ref}$ $\oplus 0.25 \cdot S_{diff} \oplus 0.25 \cdot S_{scl}) \cdot A$	no	$\prod_{i=1}^{N} P(\cdot)$
mut + A_C	$T_C = S_m \cdot A_C$	yes	$\prod_{i=1}^{N} P(\cdot)$
nested	$T_{nEMCMC} = T_{1.1} \cdot A_C$	yes	$\prod_{i=1}^{N} P(\cdot)$
rEMCMC + A_C	$T_{m,u\times unif} = S_{m,u\times unif} \cdot A_C$	yes	$\prod_{i=1}^{N} P(\cdot)$
PCTX	$0.5 \cdot S_m \cdot A \oplus 0.5 \cdot S_{pctx} \cdot A_C$		
naEMCMC	$0.5 \cdot T_{m,n} \oplus 0.5 \cdot (0.25 \cdot S_{up,rot} \oplus 0.25 \cdot S_{up,ref}$ $\oplus 0.25 \cdot S_{up,diff} \oplus 0.25 \cdot S_{up,scl}) \cdot A_C$	yes	$\prod_{i=1}^{N} P(\cdot)$
PT [32]	$(T_S = S_S \cdot A_C) \times T_m$	yes	$\prod_{i=1}^{N} P^{\frac{1}{Temp_i}}(\cdot)$
EMC [62]	$T_S \times ((1 - p_r) \cdot T_m \oplus p_r \cdot (T_{unif} = S_{unif} \cdot A_C))$	yes	$\prod_{i=1}^{N} P^{\frac{1}{Temp_i}}(\cdot)$
rEMC [63]	$(S_S \cdot A) \times (p_m \cdot T_{m,u} \oplus p_r \cdot T_{unif}$ $\oplus (1 - p_m - p_r) \cdot T_s)$	yes	$\prod_{i=1}^{N} P^{\frac{1}{Temp_i}}(\cdot)$
DeMCMC [85]	$(1 - p_r - p_{xor}) \cdot T_m \oplus p_r \cdot T_{unif} \oplus p_{xor} \cdot (S_{xor} \cdot A)$	yes	$\prod_{i=1}^{N} P(\cdot)$
popMCMC [61]	$T_{freq} = S_{freq} \cdot A_P$	yes	$\prod_{i=1}^{N} P(\cdot)$
DE-MC [87]	$(S_{dif} + S_{m,u}) \cdot A$	yes	$\prod_{i=1}^{N} P(\cdot)$
DS-MCMC [86]	$(p_m \cdot S_m \oplus (1 - p_m) \cdot S_{scl}) \cdot A$	yes	$\prod_{i=1}^{N} P(\cdot)$
SA [49]	$T_m = S_m \cdot A$	no	optim
PRSA [65]	$T_{m,u} \times T_{unif}$	no	optim
ECA	$T_{ECA} = S_{m,u} \times S_{unif} \cdot ECA$	no	$R(\cdot)$

Table 7.1: The (E)MCMC algorithms presented in this thesis: their transition matrices, if they have detailed balance (DB) or not, and their target distribution.

with $S_{y_2}(y_1 \mid x_1^{(t)}, x_2^{(t)})$ the probability to generate y_1 from $x_1^{(t)}$ and $x_2^{(t)}$ in the context of y_2. Now, such an MH acceptance rule is

$$A(y_1 \mid x_1^{(t)}) = \min\{1, \frac{P'(y_1)}{P'(x_1^{(t)})} \cdot \frac{S_{x_2^{(t)}}(x_1^{(t)} \mid y_1, y_2)}{S_{y_2}(y_1 \mid x_1^{(t)}, x_2^{(t)})}\}$$

In the next section, we show that this MH algorithm is more efficient than an MH algorithm with detailed balance. Furthermore, in our experiments, we observe that this MH algorithm is very efficient when we use recombination to propose candidate individuals. All the proposal distributions presented in Section 6.1 and the symmetrical distributions from Section 6.2 are independent of the target distribution $P(\cdot)$ since they do not consider the probability of the component states when they generate candidate individuals. In Chapter 8 we will investigate experimentally how recombination influences the sampling performance of (E)MCMCs when the proposed individuals are accepted/rejected with the standard acceptance rule.

7.2 Recombinative EMCMCs with detailed balance: all children accepted or all rejected

In the previous section we have investigated the (recombinative) EMCMC that uses the standard MH acceptance rule. Now, we show that to obtain EMCMCs with detailed balance, in general, we need to use an MH acceptance rule where all children that are proposed with recombination should be accepted or all rejected. Most EMCMCs have detailed balance because, for irreducible, aperiodic MCMCs, it is a sufficient, but not necessary, condition to converge to the target distribution. In Table 7.1, we show the algorithms presented in this paper, their transition distributions, whether they possess detailed balance or not, and the target distribution from which they sample.

7.2.1 All children accepted or all rejected

A common characteristic for the acceptance rules with detailed balance is that children are all accepted or rejected. For example, with the *coupled acceptance rule* A_C, proposed by Liang and Wong [62] two parents, $\{x_i^{(t)}, x_j^{(t)}\}$ and their two children are considered for acceptance; both children are accepted or rejected:

$$A_C(y_i, y_j \mid x_i^{(t)}, x_j^{(t)}) = \min\{1, \frac{P_i'(y_i) \cdot P_j'(y_j)}{P_i'(x_i^{(t)}) \cdot P_j'(x_j^{(t)})} \cdot \frac{S(x_i^{(t)}, x_j^{(t)} \mid y_i, y_j)}{S(y_i, y_j \mid x_i^{(t)}, x_j^{(t)})}\}$$

where $P_i'(\cdot)$ is the unnormalized target distribution of the i-th chain.

Theorem 7.2 *An MH algorithm where all children generated by parents are accepted or all rejected has detailed balance.*

Proof. Without loss of generality, we assume that from 2 parents, $X = \{x_1, x_2\}$, we generate 2 children $Y = \{y_1, y_2\}$. For simplicity, we assume that the proposal distribution is symmetrical. We show that if the target distribution of the two chains, $P(\cdot, \cdot)$, is the product of target distributions for each chain, $P_1(\cdot) \cdot P_2(\cdot)$, the MH algorithm that accepts/rejects individuals with the coupled acceptance rule, A_C, has detailed balance.

Let's consider that $X \neq Y$ and $P_1'(y_1)/P_1'(x_1) \cdot P_2'(y_2)/P_2'(x_2) < 1$. By rewriting A_C, the transition probability to go from X to Y is

$$T_C(y_1, y_2 \mid x_1, x_2) = S(y_1, y_2 \mid x_1, x_2) \cdot \frac{P_1'(y_1) \cdot P_2'(y_2)}{P_1'(x_1) \cdot P_2'(x_2)}$$

Furthermore, the inverse transition probability is $T_C(x_1, x_2 \mid y_1, y_2) = S(x_1, x_2 \mid y_1, y_2) \cdot 1$. These transition probabilities have detailed balance since

$$P(x_1, x_2) \cdot S(y_1, y_2 \mid x_1, x_2) \cdot \frac{P_1'(y_1) \cdot P_2'(y_2)}{P_1'(x_1) \cdot P_2'(x_2)} = P(y_1, y_2) \cdot S(x_1, x_2 \mid y_1, y_2)$$

The case where $X = Y$ is straightforward. We conclude that T_C has detailed balance. □

For example, when A_C is associated with the uniform recombination S_{unif} - that generates two children from two parents - the algorithm has detailed balance [62]. We denote this transition matrix with T_{unif}.

Another example is the *population acceptance rule* A_P [33], where the current population competes against the proposed population. With all recombinations, A_P has detailed balance, because

all individuals from the candidate population are accepted or all rejected. However, in practice, such an acceptance rule is not always desired, since it is not selective at individual level. For example, usually, individuals with higher and lower probabilities are proposed; with A_C the fit individuals can be rejected and the acceptance of less fit individuals depends on the family's fit individuals.

In some cases the standard acceptance rule is equivalent with the coupled and population acceptance rules.

Proposition 7.1 *A symmetrical recombination distribution which generates one child from $p \geq 2$ parents, can use the standard MH acceptance rule A to replace one of the parents with the child.*

Proof. Such an acceptance is equivalent with $A(y_i \mid x_i^{(t)})$, where y_i is the generated child, and $x_i^{(t)}$ is the parent candidate for replacement. □

For example, when the difference translation recombination, S_{dif}, is associated with a standard acceptance rule, A, the resulting transition matrix has detailed balance.

Laskey and Myers [61] use an acceptance rule which resembles an MH acceptance rule - one generated child against the parent that might be replaced - but which computes the proposal probabilities for their non-symmetrical operator over the whole population. We consider this a population acceptance rule.

We conclude that, in general, for detailed balance, the individuals that interact in the proposal distribution should also interact in the acceptance rule.

Detailed balance at population level

Most EMCMCs use family recombinations where, each generation, all individuals are randomly grouped such that each individual belongs to exactly one group. If the children generated with recombination are all accepted or all rejected with an acceptance rule as suggested in Theorem 7.2, we obtain *family transition probabilities* with detailed balance. At individual or family level, these transitions are not MCMCs, since their proposal probabilities are not stationary - they depend on how the individuals are grouped. At population level, for all possible groupings of the current population, the transition distribution is stationary. Then, the *population transition probabilities* obtained by combining the family transitions have detailed balance and define an MCMC.

In the sequel, for population recombinations, we need to use the population acceptance rule A_P, to obtain EMCMCs with detailed balance that converge, at population level, to the target distribution.

The coupled acceptance rule A_C vs. the standard MH acceptance rule A

It is interesting to notice that the MH algorithms generated with $T_{1.1}$ have a higher probability of acceptance than the algorithms generated with T_C. As a consequence, for the same proposal distribution, the algorithm determined by $T_{1.1}$, in general, is more efficient than an algorithm that uses T_C.

Proposition 7.2 *Consider an irreducible recombinative proposal distribution S that generates two candidate individuals from two parents. The MH algorithms that accepts/rejects both children has a lower acceptance probability than an MH algorithm where each child competes against one of the parents.*

Proof. Let's consider the population size $N = 2$ and a symmetrical proposal distribution S that generates two children y_1 and y_2 from two parents $x_1^{(t)}$ and $x_2^{(t)}$. In $T_{1.1}$ each child y_i competes against the corresponding parent $x_i^{(t)}$, where $i = \{1, 2\}$, in the standard MH acceptance rule. In T_C

both children compete against their parents in the coupled acceptance rule A_C. We rewrite $T_{1.1}$ given the transition matrix of an algorithm that would use the coupled acceptance rule, $T_C = S \cdot A_C$.

If both children are different from their parents, $x_1 \neq y_1$ and $x_2 \neq y_2$, we have that

$$T_{1.1}(y_1, y_2 \mid x_1, x_2) = T_C(y_1, y_2 \mid x_1, x_2)$$

When one child is equal with one of the parents, let's say $x_1 = y_1$, the transition probability of $T_{1.1}$ adds to the transition probability of T_C all the transition probabilities to a less fit individual y' than x_1, $P'(y') < P'(x_1)$, that is generated and rejected. Then

$$T_{1.1}(x_1, y_2 \mid x_1, x_2) = T_C(x_1, y_2 \mid x_1, x_2) + \sum_{y' \mid P'(y') < P'(x_1)} S(y', y_2 \mid x_1, x_2) \cdot [1 - A(y' \mid x_1)] \cdot A(y_2 \mid x_2)$$

Therefore, the transition probabilities of $T_{1.1}$ are equal with the transition probabilities of T_C plus some probabilities to individually accept good states. As a consequence, the rejection probabilities of $T_{1.1}$ are lower than of T_C. □

7.2.2 Target distributions of EMCMCs

In the following, we study the target distribution of EMCMCs with detailed balance. Note that irreducible EMCMCs that use the coupled A_C or the population A_P acceptance rule converge to $\prod_{i=1}^{N} P(\cdot)$. This target distribution is given by the product of distributions in the MH acceptance rule. By replacing this product with other mathematical functions (e.g. maximum of two values as in the next example), the corresponding EMCMC converges to a different distribution.

Example

We sample from the *order two statistics* distribution

$$P_{2:1}(\cdot, \cdot) = \max\{P(\cdot), P(\cdot)\}$$

using the coupled MH acceptance rule

$$A_{2:1}(y_i, y_j \mid x_i^{(t)}, x_j^{(t)}) = \min\{1, \frac{\max(P'(y_i), P'(y_j))}{\max(P'(x_i^{(t)}), P'(x_j^{(t)}))} \cdot \frac{S(x_i^{(t)}, x_j^{(t)} \mid y_i, y_j)}{S(y_i, y_j \mid x_i^{(t)}, x_j^{(t)})}\}$$

where $\max$ is the maximum for the values of two individuals, and $S(\cdot \mid \cdot)$ is any proposal distribution.

Proposition 7.3 *Consider an EMCMC that proposes two individuals with some irreducible proposal distribution S and accepts or rejects them all using the acceptance rule $A_{2:1}$. This EMCMC has detailed balance and converges to the order two statistic distribution $P_{2:1}$.*

Proof. According with Theorem 7.2, an EMCMC that proposes two candidate individuals from two parents and accepts/rejects them both with $A_{2:1}$ has detailed balance. If the proposal distribution is also irreducible, this EMCMC converges to the stationary distribution $P_{2:1}(\cdot, \cdot)$. □

7.2.3 Related work

In the previous sections we have investigated whether various recombinative proposal distributions in combination with some acceptance rule, result in algorithms with detailed balance. In the following we present existing EMCMC algorithms with detailed balance.

Many EMCMCs for sampling use a variant of SA for the parallel MCMCs to speed up the exploration of the searching space: each chain has attached a temperature according with a fixed cooling schedule. This algorithm, called *parallel tempering* (PT) [32], is itself an EMCMC since the chain with a higher temperature exchanges information with the colder chain to improve mixing. Since these algorithms use a coupled acceptance rule, they have detailed balance. Some algorithms use various proposal distributions that include the recombination operators; usually, these algorithms accept all proposed individuals or reject them all, thus they have detailed balance.

Parallel tempering (PT)

Parallel tempering [32] is a parallel MCMC with N chains each having a different stationary distribution $P_i(\cdot)$, where $P'_i(\cdot) = P'(\cdot)^{\frac{1}{Temp_i}}$, $i = 1, \ldots, N$. The temperatures have increasing values $Temp_1 < \ldots < Temp_N$ with $Temp_1 = 1$. Then, $P_1(\cdot) = P(\cdot)$. The temperatures $Temp_i, (2 \leq i \leq N)$ are given a constant value, typically according to a geometrically increasing series.

The candidate states are generated using mutation and accepted with the standard Metropolis-Hastings acceptance rule. Chains in PT exchange information by swapping states. Two chains i and j interact by trying to exchange their current states $x_i^{(t)}$ and $x_j^{(t)}$ with the probability 0.5. We denote this swapping recombination with S_S, where $S_S(x_j^{(t)}, x_i^{(t)} \mid x_i^{(t)}, x_j^{(t)}))$ is the probability that the states $x_i^{(t)}$ and $x_j^{(t)}$ are swapped. The states proposed with S_S are accepted using the coupled acceptance rule, A_C

$$\min\{1, \frac{P'_i(x_j^{(t)}) \cdot P'_j(x_i^{(t)})}{P'_i(x_i^{(t)}) \cdot P'_j(x_j^{(t)})} \cdot \frac{S_S(x_i^{(t)}, x_j^{(t)} \mid x_j^{(t)}, x_i^{(t)})}{S_S(x_j^{(t)}, x_i^{(t)} \mid x_i^{(t)}, x_j^{(t)})}\}$$

A_C accepts with probability 1 an exchange of states if the more fit state is inserted into the chain with the lower temperature. We can write the transition matrix of the PT algorithms as a cycle between mutation transitions, T_m and the swapping states transitions, T_S, where $T_S = S_S \cdot A_C$. Then,

$$T_{PT} = T_m \times (S_S \cdot A_C)$$

This cycle is irreducible, since from each population, we can arrive to each other population, and converges to the product of the target distributions for each chain $\prod_{i=1}^{N} P_i(\cdot)$ [32]. To increase the acceptance rate the two chains usually have adjacent temperatures ($|i - j| = 1$). Heuristically, PT improves mixing: better states of a warmer chain can be inserted in a colder chain that is mixing slower.

Evolutionary Monte Carlo (EMC)

Liang and Wong [62, 64] propose the evolutionary Monte Carlo algorithm, which incorporates recombination into the parallel tempering (PT) algorithm to speed up the search and preserve good building blocks of the current states of the chains. Like PT, EMC has a population of MCMC chains with constant and (geometrically) increasing temperatures. Chains interact through the swapping recombination S_S that attempts to exchange states between chains with adjacent temperature. They are then accepted with the coupled acceptance rule A_C. This way, the good individuals sampled in the warmer chain are transferred to a colder chain where they may be preserved longer. The lowest temperature chain $Temp_1 = 1$ converges to the target distribution $P(\cdot)$, where $P'_i(x_i) = \exp^{f(x_i)}$.

The candidate states are generated in two different ways and accepted by two different rules. With probability $1 - p_r$ each chain in the population generates a new state by mutation and accepts it

with the standard Metropolis-Hastings acceptance rule. With probability p_r, new states are generated by recombination of two states from different chains, and the offspring states are accepted with the coupled acceptance rule, A_C.

EMC uses a mixture and a cycle. The mixture has two components: mutation and recombination. The second step in the cycle swaps states between chains with different target distributions, T_S. EMC is irreducible - because $T_{EMC}(\cdot \mid \cdot) > 0$ - and has detailed balance.

$$T_{EMC} = T_S \times ((1 - p_r) \cdot T_{m,u} \oplus p_r \cdot T_{unif})$$

Liang and Wong discussed experimental results for a model selection problem and a time series change-point identification problem. These experiments showed the effectiveness of EMC as compared to PT.

Real-coded Evolutionary Monte Carlo (rEMC)

Liang and Wong [63] propose the real-coded evolutionary Monte Carlo algorithm (rEMC), which incorporates recombination into parallel MH chains to speed up the search and preserve good building blocks of the current states of the chains. rEMC uses a mixture and a cycle. The mixture has three components: mutation, a swapping recombination and the snooker operator.

With probability p_m each chain in the population generates a new state by normal mutation $S_{m,n}$ and accepts it with the standard Metropolis-Hastings acceptance rule. With probability p_r, new states are generated by the uniform recombination S_{unif}, and the offspring states are accepted with the coupled acceptance rule, A_C. With probability $1-p_m-p_r$, rEMC uses the snooker proposal distribution to generate a candidate state, y_i from a uniformly random chosen parent, $x_i^{(t)}$ and a second state, $x_j^{(t)}$ chosen using the target distribution $y_i = x_i^{(t)} + \gamma \cdot \overrightarrow{r}_i$, where $\overrightarrow{r}_i = (x_i^{(t)} - x_j^{(t)})/\|x_i^{(t)} - x_j^{(t)}\|$. The scalar γ_i is sampled from a distribution $P(x_i^{(t)} + \gamma_i \cdot \overrightarrow{r}_i) \cdot |1 - \gamma_i|^{\ell-1}$ that resembles the target distribution. Then, the snooker recombination samples directly from the target distribution, all samples are accepted. However, for experiments, the authors use a fixed normal distribution as approximation to the distribution for γ_i, in which case the snooker operator does not exploit the correlations between dimensions. This snooker recombination is also respectful, non-symmetrical, and, for the normal approximation of γ_i, it does not exploit the correlations between dimensions. We denote this transition with T_s.

The second step in the cycle swaps states between chains which interact through the swapping recombination S_S that exchange states between chains with different target distributions. They are then accepted with the coupled acceptance rule A_C. rEMC is irreducible - because $T_{rEMC}(\cdot \mid \cdot) > 0$ - and has detailed balance.

The authors show some experimental results for some mixture of Gaussian, galaxy data and two Bayesian neural network examples. These experiments showed the effectiveness of rEMC as compared to other parallel MHs that do not use uniform and snooker recombination.

Discrete spaces Evolutionary MCMC (DeMCMC)

Strens [85] proposes a population based MCMC algorithm for discrete search spaces. Besides mutation and uniform recombination, Strens uses the xor proposal operator to generate candidate individuals. Recall that xor is a special case of the total distance recombination, S_{dif} where, for each position h, the hamming distance between the child y_i and the parent $x_i^{(t)}$ is the same as the hamming distance between two different other parents chosen at random from the current population, $x_{i+1}^{(t)}$ and $x_{i+2}^{(t)}$.

Strens, in his discrete EMCMC (DeMCMC), uses a mixture with three components: mutation with probability $1 - p_r - p_{xor} > 0$, uniform recombination with probability p_r, and xor recombination with probability p_{xor}. DeMCMC accepts the candidates proposed with uniform mutation $S_{m,u}$ with the standard acceptance rule, A, and the candidates proposed with uniform recombination S_{unif} with the coupled acceptance rule A_C and the candidates proposed with the xor recombination S_{xor} with the standard acceptance rule.

$$T_{DeMCMC} = (1 - p_r - p_{xor}) \cdot S_{m,u} \cdot A \oplus p_r \cdot S_{unif} \cdot A_C \oplus p_{xor} \cdot S_{xor} \cdot A$$

For $1 - p_r - p_{xor} > 0$, DeMCMC is irreducible because mutation is irreducible. It also has detailed balance since all component transitions have detailed balance. Thus, it converges to the product of the target distributions of each chain $\prod_{i=1}^{N} P_i(\cdot)$. Strens shows the efficiency of DeMCMC on an application problem from medical diagnostics.

Population MCMC (popMCMC)

Laskey and Myers [61] introduced popMCMC where a population of MCMC chains exchange information through a population-based recombination proposal distribution. As before the Boltzmann distribution is used: $P'(x) = \exp^{f(x)}$. Each generation a locus $x_i^{(t)}[h]$ is randomly picked to be mutated. An allele $y_i[h]$ on the h-th locus of the candidate individual is generated using a distribution $(N(y_i[h]) + 1)/(N + |\Omega(x[\cdot])|)$, where $N(y_i[h])$ is the number of individuals in the current population that have the allele $y_i[h]$ on the h-th locus and $|\Omega(x[\cdot])|$ is the total number of alleles. y_i is accepted using the standard MH acceptance rule $A(y_i \mid x_i^{(t)})$. Note that this recombination is similar with S_{tree} but without considering the relationship between different loci in the current population.

PopMCMC has detailed balance and converges to the stationary distribution of N independently sampled points from $P(\cdot)$. The authors show experimentally that popMCMC finds highly likely Bayesian network structures faster than multiple independent MCMC chains.

Differential Evolution Markov chain (DE-MC)

Ter Braak [87] proposes a symbiosis between the recombinative operators from Differential Evolution (DE) - a genetic algorithm in real coded space - and a population of EMCMCs. DE-MC exchanges information between chains using a sum of the differential recombination and uniform mutation. A candidate individual y_i is generated by adding to a parent $x_i^{(t)}$ the difference between two other distinct states uniform randomly selected from the population, $x_j^{(t)}$ and $x_k^{(t)}$ and a number generated with uniform mutation, γ_m. Since both the differential recombination and mutation does not exploit the correlations between dimensions, for each locus h, we have $y_i[h] = x_i^{(t)}[h] + \gamma_i \cdot (x_j^{(t)}[h] - x_k^{(t)}[h]) + \gamma_m$, where $\gamma_i \in [0.4, 1.0]$ is a constant, $\gamma_m \in [-0.0001, 0.0001]$, and $i \neq j \neq k$. The author shows that this operator is irreducible and symmetrical. This candidate state, y_i, is accepted using a standard Metropolis rule $A(y_i \mid x_i^{(t)})$. Since only one individual is proposed with a symmetrical proposal distribution, the algorithm has detailed balance. Then, DE-MC has detailed balance and it converges to the product of the target distributions for each chain.

Furthermore, for population sizes that go to infinity and for mutation rates that go to 0, the proposal distribution of DE-MC, at population level, resembles the target distribution if the individuals from the population are independently sampled from the target distribution. Note that the computational effort for such a proposal distribution, as population level, is linear with the population size and, since DE-MC uses the differential recombination is linear with the number of dimensions; thus, it is $\mathcal{O}(\ell \cdot N)$.

The author shows the efficiency of the DE-MC algorithm on normal and multivariate distributions as compared with the standard single chain MCMC.

Direct search MCMC (DS-MCMC)

Strens et al. [86] includes in the rEMC from Liang and Wong [63] framework various recombination proposal distributions closely related with the snooker recombination or the differential recombination. Strens noticed that the snooker recombination from Liang and Wong [63] is symmetrical when γ is sampled from a uniform symmetric distribution. The translation vector $\overrightarrow{r}_i$ is the unity vector and defines only the direction for translation which is the same with the translation vector from the child y_i to the parent $x_i^{(t)}$. However, this proposal distribution does not follow the target distribution and the candidate individuals are accepted with the standard MH acceptance rule, A. Furthermore, this snooker generator does not exploit the correlations between dimensions and is respectful.

In addition, Strens et al. [86] propose two other proposal distributions that are closely related with the differential translation recombination in the sense that the translation vector does not depend on the current individual, $x_i^{(t)}$. Then $y_i = x_i^{(t)} \cdot \kappa_1 \cdot \exp \kappa_2 + (1 - \kappa_1 \cdot \exp \kappa_2) \cdot \overrightarrow{r}_i$, where $(\overrightarrow{r}_i$ is the unity vector that depends on all individuals from the current population $X^{(t)}$ except the current one, $x_i^{(t)}$, and κ_1 and κ_2 are uniformly sampled from $\{-1, 1\}$. The authors proof that this proposal distribution is symmetrical and accepts the proposed individual with the standard acceptance rule, A. Note that it is also respectful and does not exploit the correlations between dimensions. Furthermore, this snooker algorithm does not adapt the jump of a child from its parent according to the distance between its parents.

Using these proposal distributions, Strens et al. [86] build different EMCMC algorithms that are a mixture of the normal mutation and one of the recombinations discussed above. They compare the performance of their EMCMC algorithms with rEMC on a mixture of two Gaussians.

7.2.4 Nested transition distributions

In the following, we propose a method to integrate the transition distributions without detailed balance in MH algorithms with detailed balance. To achieve this, we need to accept or reject all the individuals generated with an MH algorithm without detailed balance.

Definition 7.1 *A* nested MH algorithm *is an MH algorithm where individuals are proposed using a transition distribution, and are further all accepted or all rejected by a coupled MH acceptance rule. A* nested transition distribution *is the transition distribution used as proposal distribution by the nested MH algorithm.*

Furthermore, the nested transition distribution that generates individuals with a recombination distribution is itself a recombinative proposal distribution: from two or more parents we propose two or more children.

Proposition 7.4 *The nested MH algorithm has detailed balance. The nested transition distribution composed by a respectful recombination proposal distribution and an acceptance rule is by itself a respectful recombination proposal distribution.*

Proof. The proof is immediate if we consider the nested transition distribution as a proposal distribution. If parents have identical values at certain positions, then the individuals generated by respectful recombination have - by definition - the same values at those positions. An acceptance rule simply selects from parents and children, therefore the accepted individuals have the same values on those positions. □

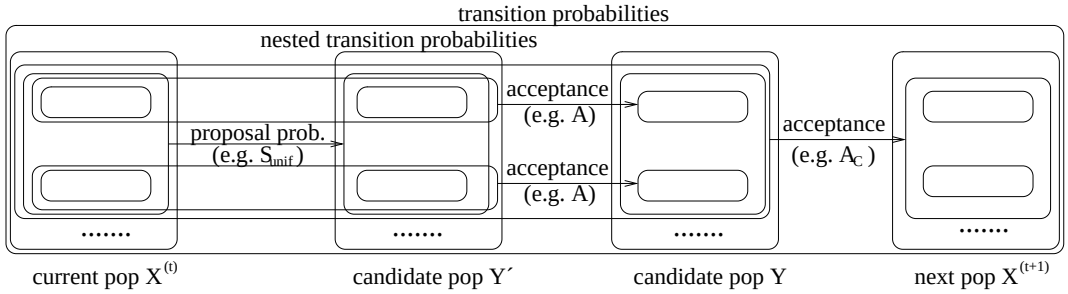

Figure 7.1: Nested EMCMC framework: a candidate population Y' is proposed with some proposal distribution S from the current population X^t and some children are accepted with some MH acceptance rule. These accepted children and the parents that are not replaced form the candidate population Y and competes against X^t such that the resulting EMCMC has detailed balance.

Nested transitions are, usually, non-symmetrical. Thus, we need to compute these probabilities. In Figure 7.1, we graphically depict the nested EMCMC framework.

Example

Consider the *nested EMCMC* that uses as proposal distribution the nested transition distribution, $T_{1.1}$ where two candidate individuals are proposed from two parents with some recombinative proposal distribution, S, and each child competes against one of the parents randomly chosen from the population with a standard MH acceptance rule. The candidate individuals proposed with $T_{1.1}$ are, at their turn, accepted with the coupled acceptance rule, A_C. The nested EMCMC's transition distribution is

$$T_{nEMCMC} = T_{1.1} \cdot A_C = (S \cdot A \cdot A) \cdot A_C$$

where the coupled acceptance rule is

$$A_C(y_1, y_2 \mid x_1, x_2) =$$

$$\min\{1, \frac{P'(y_1) \cdot P'(y_2)}{P'(x_1) \cdot P'(x_2)} \cdot \frac{T_{1.1}(x_1, x_2 \mid y_1, y_2)}{T_{1.1}(y_1, y_2 \mid x_1, x_2)}\}$$

We observe that the nested EMCMC eliminates the influence of the proposal distribution on $T_{1.1}$'s target distribution with the coupled acceptance rule, A_C.

In the following proposition, we express T_{nEMCMC} as a function of $T_{1.1}$ and the proposal distribution S.

Proposition 7.5 *Consider that the proposal distribution S generates y_1 and y_2 from x_1 and x_2. If both children are accepted, $\{x_1, x_2\} \neq \{y_1, y_2\}$, the nested transition distribution is*

$$T_{nEMCMC}(y_1, y_2 \mid x_1, x_2) = T_{1.1}(y_1, y_2 \mid x_1, x_2)$$

If one child is accepted, y_1, and one rejected, y_2, then

$$T_{nEMCMC}(y_1, x_2 \mid x_1, x_2) = T_{1.1}(y_1, x_2 \mid x_1, x_2) \cdot \min\{1, \frac{S(x_1, y_2 \mid y_1, x_2)}{S(y_1, y_2 \mid x_1, x_2)}\}$$

Otherwise, if both children are rejected,

$$T_{nEMCMC}(x_1, x_2 \mid x_1, x_2) = 1 - \sum_{y_1, y_2 \neq x_1, x_2} T_{nEMCMC}(y_1, y_2 \mid x_1, x_2)$$

Proof. If both children are accepted, then

$$T_{1.1}(y_1, y_2 \mid x_1, x_2) = S(y_1, y_2 \mid x_1, x_2) \cdot A(y_1 \mid x_1) \cdot A(y_2 \mid x_2)$$

The reverse transition probability is $T_{1.1}(x_1, x_2 \mid y_1, y_2) = S(x_1, x_2 \mid y_1, y_2) \cdot A(x_1 \mid y_1) \cdot A(x_2 \mid y_2)$. In this particular case, the detailed balance condition holds $P(x_1) \cdot P(x_2) \cdot T_{1.1}(y_1, y_2 \mid x_1, x_2) = P(y_1) \cdot P(y_2) \cdot T_{1.1}(x_1, x_2 \mid y_1, y_2)$ and the coupled acceptance probability is 1. The nested transition probability now is $T_{nEMCMC}(y_1, y_2 \mid x_1, x_2) = T_{1.1}(y_1, y_2 \mid x_1, x_2)$.

If one child is accepted and the other rejected, for example y_2 is rejected and y_1 is accepted, then

$$T_{1.1}(y_1, x_2 \mid x_1, x_2) = S(y_1, y_2 \mid x_1, x_2) \cdot A(y_1 \mid x_1) \cdot [1 - A(y_2 \mid x_2)]$$

The reverse transition probability is $T_{1.1}(x_1, x_2 \mid y_1, x_2) = S(x_1, y_2 \mid y_1, x_2) \cdot A(x_1 \mid y_1) \cdot [1 - A(y_2 \mid x_2)]$. Now, the coupled acceptance is $A_C(y_1, x_2 \mid x_1, x_2) = \min\{1, S(x_1, y_2 \mid y_1, x_2)/S(y_1, y_2 \mid x_1, x_2)\}$, since $P(y_1)/P(x_1) = A(x_1 \mid y_1)/A(y_1 \mid x_1)$. The second equation from the proposition now follows directly.

If both children are rejected then

$$T_{1.1}(x_1, x_2 \mid x_1, x_2) = 1 - \sum_{y_1, y_2 \neq x_1, x_2} T_{1.1}(y_1, y_2 \mid x_1, x_2)$$

The last equation from the proposition follows directly by replacing the above equation in the definition of T_{nEMCMC}. □

From the above proposition, we note that the difference between the two transition distributions, T_{nEMCMC} and $T_{1.1}$, is given by the fraction $S(x_1, y_2 \mid y_1, x_2)/S(y_1, y_2 \mid x_1, x_2)$.

In the following proposition, we show that, for the same proposal distribution, the performance of a nested algorithm is situated in-between the performance of an MH algorithm that use the standard acceptance rule and an algorithm that uses the coupled acceptance rule. But unlike $T_{1.1}$, T_{nEMCMC} has detailed balance.

Proposition 7.6 *Consider a proposal distribution S that generates two children y_1 and y_2 from two parents x_1 and x_2. T_{nEMCMC} has a higher or equal acceptance rate than T_C but smaller or equal than $T_{1.1}$. If $S(x_1, y_2 \mid y_1, x_2) = S(y_1, y_2 \mid x_1, x_2)$, then T_{nEMCMC} is equivalent with $T_{1.1}$. If S is irreducible, than T_{nEMCMC} has detailed balance and converges to the target distribution $\prod_{i=1}^{N} P(\cdot)$.*

Proof. From Proposition 7.5, we know that the probability to accept both proposed individuals is equal for T_{nEMCMC}, T_C and $T_{1.1}$. Otherwise T_C rejects both individuals, whereas T_{nEMCMC} and $T_{1.1}$ have a non-zero probability to accept one child and to reject one child. Therefore, T_C has the lowest acceptance probability from the three algorithms. $T_{nEMCMC} = T_{1.1} \cdot A_C$ has a lower acceptance probability than $T_{1.1}$ because the children accepted with $T_{1.1}$ can be still rejected with some probability by the coupled acceptance rule.

If $S(x_1, y_2 \mid y_1, x_2) = S(y_1, y_2 \mid x_1, x_2)$, then $S(x_1, y_2 \mid y_1, x_2)/S(y_1, y_2 \mid x_1, x_2) = 1$ and, from Proposition 7.5 we have that $T_{nEMCMC} = T_{1.1}$.

The last property follows directly from Proposition 7.4. □

To use T_{nEMCMC}, we have to compute the proposal probabilities to calculate the acceptance probabilities; then, mixtures and restricted position cycles are preferred to unrestricted cycles that are computationally infeasible.

A nested algorithm that would use T_C as nested transition distribution is equivalent with T_C. We have thus $T_C \cdot A_C = T_C$ which means that T_C is invariant with the coupled acceptance rule. Thus, this algorithm is useful only if the nested transition distribution does not have detailed balance.

7.3 Elitist acceptance rules for EMCMCs for optimization

In the previous sections, we have investigated recombinative EMCMCs for sampling. When we are interested in exploring optimal solutions, we want an algorithm that finds local optima and, at the same time, escapes from these local optima to search for other local and/or global optima. For these algorithms, the detailed balance condition is not important since optimization is the goal rather than sampling. In the following, we propose and analyze recombinative EMCMCs that can be used to sample disproportionately often from high fit individuals, and thus can be used for exploring optimal solutions. We increase the efficiency of these algorithms by assigning temperatures to the existing chains such that high fit individuals remain longer in the population whereas the low fit ones are used for exploration of the search space. Unlike SA whose exploration capability decreases when temperature is lowered, our scheme keeps its exploration and exploitation behaviour during the entire run.

7.3.1 Related work

Most of the EMCMCs used for optimization [65, 80, 35] modify the target distribution toward the high regions with a simulated annealing (SA) type of algorithm. Hajek [40] showed that detailed balance is not necessary for SA algorithms; instead we need to know the lowest probability with which an individual can escape from a local optima.

Parallel recombinative simulated annealing (PRSA)

Mahfoud and Goldberg [65] proposed a population-based simulated annealing algorithm which made use of recombination. All individuals from the population have the same temperature which decreases every generation according to a cooling schedule. New individuals are generated using mutation and one point recombination between two individuals $x_i^{(t)}$ and $x_j^{(t)}$. PRSA uses the logistic acceptance rule to accept a candidate individual. Two possible competitions are considered: single acceptance/rejection holds two competitions between a parent vs. the child formed from its own right-end and the other parent left-end, or double acceptance/rejection holds one competition between both parents vs. both children using the coupled acceptance rule:

$$\min\{1, 1/(1+\exp\frac{-f(y_i)-f(y_j)+f(x_i^{(t)})+f(x_j^{(t)})}{Temp^{(t)}})\}$$

When the candidate individuals are accepted/rejected with this coupled acceptance rule, for a fixed temperature, PRSA has detailed balance because the proposal distribution is symmetrical. Otherwise, when one child competes against one parent, the algorithm does not have detailed balance.

Recently, Chen and Pitt [14] proposed a variant of PRSA for the Traveling Salesman Problem (TSP). They showed that exploiting commonalities with recombination helps in finding the optima for this particular problem. However, their PRSA variant still does not achieve detailed balance.

7.3.2 Elitist coupled acceptance rule (*ECA*)

The ECA algorithm is a family competitive acceptance rule where the best two solutions from the family of four is kept if at least one of them is a child. Otherwise, when both children have a lower fitness than both their parents, the children can probabilistically replace the parents.

ECA can be viewed as a combination between the elitist replacement rule from regular GAs [88] and the coupled acceptance rule A_C. When compared with the elitist replacement, ECA is more exploratory but less elitist since it still accepts with some probability less fit individuals. When compared with A_C and A acceptance rules, ECA is more elitist but less exploratory. With ECA, if a child and a parent are the two most fit individual states from two parents and their children, they are *always* accepted whereas with A the other child will be accepted with some probability. Thus, this algorithm is climbing towards a local optima since it is very probable that a good solution remains a long time in the population to generate better solutions. However, the probability to escape from the basin of attraction of a peak, as we show in the next paragraphs, is rather poor when compared with the transition distribution T_C generated with the same proposal distribution and the coupled acceptance rule A_C.

We now describe the transition distribution generated by accepting with ECA the individuals proposed with the proposal distribution S. We denote it with T_{ECA}, and, to establish the properties of the corresponding target distribution, we compare it with T_C. We call $\max_2$ the function returning the two most fit solutions. The transition probability to accept or reject both children, $\{y_1, y_2\}$, proposed with the proposal distribution S is only non-zero if both children are better or worse than their parents, $\{x_1, x_2\}$ - that is $max_2\{x_1, x_2, y_1, y_2\}$ is either $\{y_1, y_2\}$ or $\{x_1, x_2\}$. Then

$$T_{ECA}(y_1, y_2 \mid x_1, x_2) = S(y_1, y_2 \mid x_1, x_2) \cdot \min\{1, \frac{P'(y_1) \cdot P'(y_2)}{P'(x_1) \cdot P'(x_2)} \cdot \frac{S(x_1, x_2 \mid y_1, y_2)}{S(y_1, y_2 \mid x_1, x_2)}\}$$

In this case, the transition probability of *ECA* is equal with $T_C(y_1, y_2 \mid x_1, x_2)$.

If one of the children, let's say y_1, and one of the parents, let's say x_2 are the most fit, then the transition probability to go from $\{x_1, x_2\}$ to $\{y_1, x_2\}$ is the sum over the probabilities to propose a state y_2 that is smaller or equal than $P'(y_1)$ and $P'(x_2)$ and, then to reject it. Then

$$T_{ECA}(y_1, x_2 \mid x_1, x_2) = \sum_{y_2 \mid \{y_1, x_2\} = \max_2\{y_1, y_2, x_1, x_2\}} S(y_1, y_2 \mid x_1, x_2)$$

If $\{y_1, y_2\} \cap \{x_1, x_2\} = \emptyset$, then $T_{ECA}(y_1, y_2 \mid x_1, x_2) = 0$. To make this transition distribution stationary we take the rejection probability

$$T_{ECA}(x_1, x_2 \mid x_1, x_2) = 1 - \sum_{\{y_1, y_2\} \neq \{x_1, x_2\}} T_{ECA}(y_1, y_2 \mid x_1, x_2)$$

For simplicity of the discussion, let us consider a symmetrical proposal distribution. Then, only when both children are worse than their parents this algorithm rejects the two candidate individuals with some probability. Otherwise, ECA always accepts at least one child. As a consequence, the probability to accept at least one proposed child is the largest from all the previous acceptance rules. Thus, an algorithm that uses ECA behaves more similar to a standard GA than to a sampling algorithm. As a consequence, the target distribution of an ECA algorithm is disproportionally high for more probable states at the expense of less probable states.

In the following proposition, we show that ECA, in combination with an irreducible proposal distribution, generates an algorithm without detailed balance but which converges to some target distribution.

Proposition 7.7 *Consider a proposal distribution S that generates two children from two parents which are then accepted or rejected with the ECA acceptance rule. This algorithm does not have detailed balance for any non-uniform distribution S. If S is irreducible then this MH algorithm converges to a stationary distribution $\prod_{i=1}^{N} R(\cdot)$.*

Proof. We show that ECA does not have detailed balance for any non-uniform distribution. Let's consider three states such that $P'(x_1) > P'(x_2) > P'(y_1)$. The probability $T_{ECA}(y_1, x_2 \mid x_2, x_1) = 0$ whereas $T_{ECA}(x_1, x_2 \mid y_1, x_1) \neq 0$. Therefore, T_{ECA} does not have detailed balance for any S.

The transition matrix of ECA is stationary by definition. When S is irreducible, this algorithm is irreducible since it can generate any state from any other state with a non-zero probability. Therefore, the target distribution of ECA exists. □

Note that the target distribution of ECA is biased towards high regions of $P(\cdot)$: the highest states are sampled the most often in the detriment of the lower states. In the following, we propose a mechanism to increase the exploration abilities of the ECA algorithm which, at the same time, preserves its elitist properties.

7.3.3 Fitness ordered tempering (FOT)

Fitness ordered tempering, FOT, is similar with the parallel tempering (PT) technique [32]; whereas PT maintains a population of N MCMCs with a fixed temperature $Temp_i$, FOT attaches, each generation, a temperature to a state according to its fitness value. Contrary to parallel tempering, here the temperature assignment depends on the fitness of the solutions relative to the fitness of the other solutions in the current population. Since we want to remain in the vicinity of good solutions, we prefer to assign the lower temperatures to the better solutions. This ensures the exploitation properties of this algorithm. With less fit individuals we ensure the exploration of the search space: since they are assigned the high temperatures, almost proposed all individuals (fit and less fit) are accepted.

The population of N solutions (or states) is sorted according to their fitness (or probability), and the solution at rank i gets the temperature $Temp_i$, where the most fit solution gets $Temp_1 = Temp^{min}$, and the worst solution $Temp_N = Temp^{max}$. As in parallel tempering, FOT has a - typically geometrically - increasing series of temperatures $Temp_1 = Temp^{min} < \ldots < Temp_N = Temp^{max}$. Therefore a solution has lower temperature than any solution worse than itself, unless they are copies of each other. In case there are multiple copies of the same solution ties within the sorted population are broken randomly, so each copy gets an adjacent temperature. Copies receive a different temperature to avoid that multiple copies of good solutions will remain in the population almost indefinitely.

In case there are multiple solutions with the same fitness but which are not copies of each other, the temperature ladder has to be recomputed so that each unique solution with the same fitness gets the same temperature. The number of different temperature values $Temp_i$ at each generation is therefore equal to the number of solutions with different fitness value, unless they are copies, in which case they also get a different temperature values. This scheme is necessary to avoid that some solution might get another temperature in an identically composed population, and ensures that FOT has a homogeneous Markov chain transition matrix at population level. FOT is an EMCMC that exchanges information between the states by means of the temperatures.

Note that the target distribution of such an algorithm depends very much on the tempering schedule. For temperatures less than 1, the target distribution is peaked and concentrated in the fit states. Thus, the probability to escape from a local optimum is low and the exploration properties of FOT are weak. For temperatures much higher than 1, the exploitation properties are lost: the target distribution is flat since all the states are accepted with high probability regardless of their probability in the initial search space. Furthermore, if all the temperatures are close to 1, the exploration

and exploitation properties of this algorithm are very similar to those of MIC. We therefore set $Temp^{min} \ll 1 \ll Temp^{max}$.

7.3.4 ECA+FOT

We combine the ECA algorithm – designed to focus on local maximum – with the FOT temperature schedule, which is designed to keep exploring the search space while maintaining the highly probable states in the population. This way, we increase the exploitation capabilities of both algorithms without decreasing FOT's exploration characteristics.

Each generation, *ECA+FOT* first assigns the temperatures following the FOT schedule as explained above. Then, we generate two candidate individuals with a proposal distribution S and we accept the generated individuals with the ECA acceptance rule.

Since *ECA+FOT* is basically the same algorithm as *ECA* - only the states have been attached various different temperatures - *ECA+FOT* has the same properties as *ECA*. Therefore, *ECA+FOT* does not have detailed balance and it converges to a distribution that is biased towards high regions of $P(\cdot)$.

Chapter 8

Experiments

The goal in MCMC algorithm design is to maximize their mixing behavior - that is, how fast and how well the algorithm samples from the target distribution. Previously, we have shown that to achieve detailed balance most recombination operators need to be combined with the coupled acceptance rule. Here we will show experimentally that the coupled acceptance comes at a severe cost: when applied with mutation the resulting MCMC shows worse performance than an MCMC with mutation and the standard Metropolis acceptance rule. However, the experiments also show that the gains obtained by using recombination can make up for the performance loss suffered by applying the coupled acceptance rule.

In the next section, we perform two analytical experiments. Although this analysis offers us an exact and fair comparison between MCMCs, its computational cost increases exponentially with the dimensionality of the search space. In the next section, we show that a population of MCMCs can outperform the performance of a single long MCMC, and that recombination can improve the performance of MCMC on two toy problems. In Section 8.2, we perform experiments on larger search spaces. In fact, the larger the search space the less exact is the experimental methodology in assessing the performance of MCMCs. Therefore, in Section 8.3, for the real-coded MCMCs, we compare a histogram over the target distribution with the histogram obtained from the sampled states to assess the performance of MCMCs. When one represents a continuous probability distribution with a histogram, information is lost because we approximate the information in a bin using the mean of the states and not their variance.

8.1 Analytical experiments

In this section, we analytically compare the performance of some discrete space MCMCs and EMCMCs on two toy problems. We first show the advantage of using a population of independent MCMCs, MIC, instead of one long MCMC. Then, we show that recombination improves the performance of a population of MCMCs.

To measure and compare the efficiency of the two EMCMC algorithms we compute: i) the second largest eigenvalue of the transition matrices [33] and ii) the mean hitting times of arriving from a peak to another peak (from all 0's to all 1's) and from the low valued states to a peak also using the transition matrix. An MCMC is considered to mix fast when the second eigenvalue λ_2 of its transition matrix is bounded away from 1: an MCMC with a smaller λ_2 is considered more efficient than an MCMC with a higher λ_2 [7]. λ_2 bounds the time after which an MCMC converges.

The hitting time, t_v^W, is the mean time necessary to reach a subset W from the state v with an MCMC chain. In terms of matrices, we have $t^W = (I - B)^{-1} \cdot 1'$, where B is obtained from the transition matrix by eliminating the columns and rows with $v \in W$, I is the identity matrix, $1'$ is a vector with all elements one. The vector of mean hitting times, $t_V^W \in t^W$, is the minimal non-negative solution. For this problem we measure two hitting times: i) to go from one peak to another peak, we denote this time with t_{peak}^{peak} and ii) to go from half ones that is the middle of low valued region to one peak, we denote this time with t_{valley}^{peak}. Note that the cost of this analysis increases exponentially with the string size ℓ. Therefore, in this section we will use distributions on small discrete search spaces.

8.1.1 MIC vs. single long MCMC

In this paragraph we analytically show the usefulness of *MIC* versus the standard *MCMC* on a toy problem. This problem has two peaks with high valued individuals separated by a long valley with many low valued individuals. Let us assume that a single chain *MCMC* spends a lot of time in one of the peaks. Then, with 2 or more chains started in the valley with low values, the probability to sample from both peaks is larger than with a single chain.

The tested distribution

It has two equal valued maxim: one when all bits are 0's and one for all bits 1's. Further, the strings with the same amount of 1's have the same fitness value. Then, the hamming distance between the two peaks is maximal, ℓ. We consider binary strings of size $\ell = \{8, 20, 32\}$ and we take the width of a peak (the minimal hamming distance between a peak and a low valued individual from the valley) $x = 3$. The analytical expression of this function is $f_{2peaks}(x) = \max\{0.01, 3 - \min\{x, \ell - x\}\}$. To study how the performance of the two algorithms varies when increasing or decreasing the height of the two peaks, we associate to the fitness values a temperature $Temp > 0$ such that $P'(x) = f_{2peaks}(x)^{1/Temp}$. In Figure 8.1 a), the higher the temperature, the closer the peaks' values are to the valley' values, and thus, the closer the target distribution to the uniform distribution. If $Temp = 1$ then $P'(x) = f_{2peaks}(x)$.

For this example, we reduce the computational cost by grouping individuals with the same number of ones and zeros in one group [81]. These individuals have the same fitness value and thus the same acceptance probability. For a building block of size b, after reduction, we obtain strings of size l/b which can take $(b + 1)^{l/b}$ values.

The tested MCMCs.

We compare the performance of the following two MCMCs.

1. *MCMC*: a standard, single algorithm that proposes new states with the uniform mutation $S_{m,u}$ with a mutation rate $p_m = 1/\ell$ and accepts (rejects) them using the Metropolis acceptance rule A.

2. *MIC*: two independent MCMCs that propose new states with $S_{m,u}$ the same mutation rate and accept (reject) them using the Metropolis acceptance rule A.

Results

It is interesting to note that λ_2 and the hitting times of *MIC* and the standard *MCMC* are equal since they are basically the same algorithm; the only difference is that *MIC* runs a population of *MCMC*s.

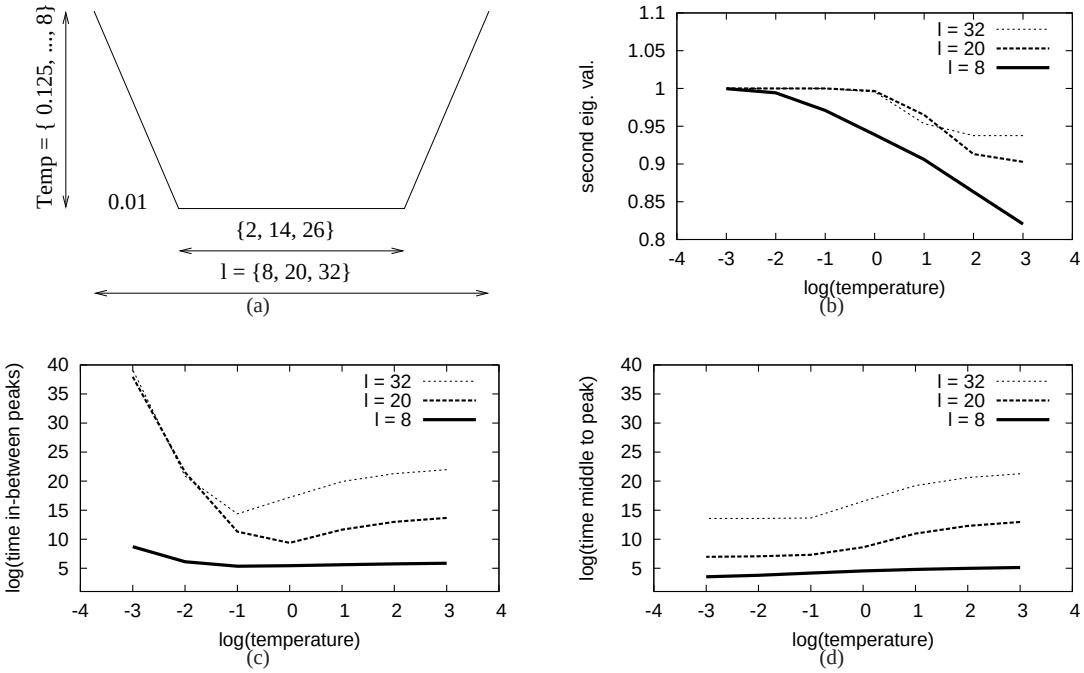

Figure 8.1: a) The 2 peaks problem where we vary the length of the string l and the temperature values, and thus the peak height, $Temp$. b) Second largest eigenvalues for the 2 peaks problem of *MCMC* and *MIC* algorithms. c) Mean hitting time to go from one peak to one other peak and d) from middle of the valley to one peak, where each line represents a string size.

In Figure 8.1 b), we show the second largest eigenvalues for various settings of this function. The larger the string size and the lower the temperature, the closer are these eigenvalues to 1. Except for the string of 8 bits, the eigenvalues for $Temp \leq 1$ are practically 1; thus both *MCMC* and *MIC* mix quite poorly for this function.

To show that *MIC* samples better the two peaks than a single long *MCMC*, we compute the difference between the two hitting times: i) t^{peak}_{peak}, see Figure 8.1 c), and ii) t^{peak}_{valley}, see Figure 8.1 d). The bigger the discrepancies between these two times, the more useful *MIC* is as compared with the standard *MCMC*; then, *MIC*, with its two chains, is able to visit faster both peaks. Consider that the starting states of *MCMCs* are generated at random. Then, the most probable states are the low valued ones; there are only two states with maximum values but there are $2^{\ell/2}$ half ones states.

From Figure 8.1 c) and d), we observe that two hitting times - that are t^{peak}_{peak} and t^{peak}_{valley} - decrease with the string size and with the temperature height. For strings of size 8 and for $Temp > 1$ there is little difference between the two hitting times; also for these settings *MIC* and *MCMC* are mixing well. The difference between times drastically increases with the decreasing of temperature; the lower the temperatures, and the more peaky the target distribution is, the hardest for a single chain to escape from one peak. The larger the temperatures, the longer it takes to go from strings with half of the bits equal to 1 to one of the peaks; with low temperatures, once a peak is found, the chain accepts with high probability the states which have higher fitness than the current state. When the temperature is high, the probability to accept a less fit individual is high and thus the *MCMC* can jump out of the

peak without sampling the largest valued individual. The hitting times also increase with the string size: the larger the length of the valley, the larger the amount of time needed to cross it.

Since it is easier to arrive from the middle of the search space to one peak than to cross the low valued region, we conclude that a population of independent *MCMC*s can improve mixing of single long chain *MCMC* when two or more important regions of the target distribution are separated by a long region with unfit individuals. Then, the probability to visit two or more peaks is larger given a "small" number of samples for a *MIC* than for an *MCMC*.

8.1.2 MIC vs. recombinative EMCMCs

In the following, we compare the performance of recombinative and non-recombinative population-based MCMCs on a toy problem, the *hyper-geometrical* distribution. We show that recombinative EMCMCs can outperform the standard MCMCs. Furthermore, we show that the algorithms that use the coupled acceptance rule A_C are less efficient than the algorithms that use the standard acceptance rule A.

The tested MCMCs.

We compare the performance of four population based MCMCs: two non-recombinative MCMCs with two recombinative EMCMCs. We take the size of population $N = 2$.

1. *MIC*: 2 independent MCMCs that propose new states with $S_{m,u}$ with the mutation rate $p_m = 1/\ell$ and accept (reject) them using the Metropolis acceptance rule A.

2. $mut + A_C$: a non-recombinative population-based MCMC that proposes each generation 2 new states with the same $S_{m,u}$ and accepts (rejects) all of them using the coupled acceptance rule A_C.

3. *rEMCMC*: generates two individuals with a cycle between $S_{m,u}$ and parameterized uniform recombination, S_{unif}, with $p_r = 50\%$, and then accepts them with the standard Metropolis acceptance rule A.

4. *rEMCMC* + A_C: generates two individuals with a cycle between $S_{m,u}$ and S_{unif} and then accepts them with the coupled acceptance rule A_C.

As shown in previous sections, three of these algorithms – *MIC*, *mut*+A_C and *rEMCMC*+ A_C – converge to the target distribution, $\prod_{i=1}^{N} P(\cdot)$; the marginal distribution of a single chain of *rEMCMC* is $P(\cdot)$.

The tested distribution

A *hyper-geometric distribution* (Hyper) is

$$P'(x) = \begin{cases} h_2 \cdot \frac{w - \Delta(x,x_0)}{w} & \text{if } \Delta(x,x_0) < w \\ 0.01 & \text{if } \Delta(x,x_0) = w \\ h_1 \cdot \frac{\Delta(x,x_0) - w}{\ell - w} & \text{otherwise} \end{cases}$$

with ℓ the string size, w the number of bits-1 in the individuals with the lowest value 0.01, individual x_0 with all bits equal to 0 is the second largest peak h_2, and the individual with all bits equal to 1 is the largest peak h_1. We set $\ell = 8$ and $h_1 = 1$.

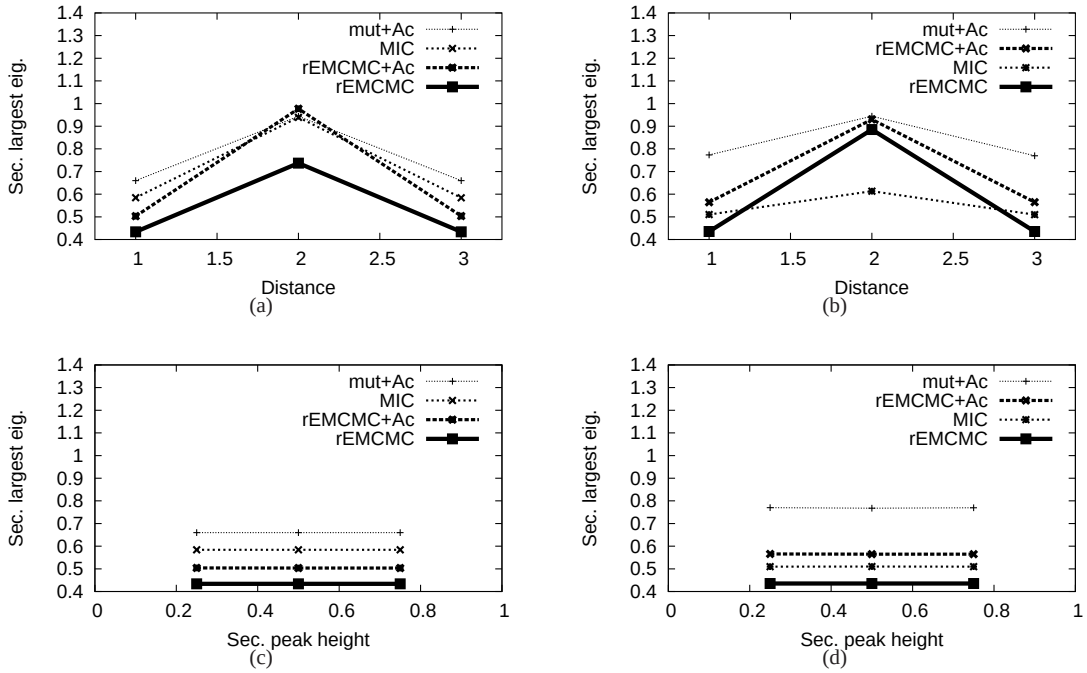

Figure 8.2: Second largest eigenvalues for Hyper-geometrical function on 8 bits, that is 2 blocks each of 4 bits, where (a,b) $w = \{1, 2, 3\}$ and the peak heights are set to $h_1 = 1$ and $h_2 = 0.75$, and (c,d) $h_2 = \{0.25, 0.5, 0.75\}$ and the distance to the highest peak is set to $w = 3$.

Results

To compare MH algorithms analytically, we compute the second largest eigenvalue of the transition matrices of the corresponding (E)MCMCs. In the first experiment, see Figure 8.2 a) and b), we vary the distance of the lowest valued states to the optimum, $w = \{1, 2, 3\}$ and we set the value of the second largest state to $h_2 = 0.75$. In this case, the local and global optimum have a close value and we vary their basin of attraction: the greater the distance from the local optimum, the smaller the basin of attraction of the global optimum. Second, we vary h_2 from 0.25 to 0.75 with a step size of 0.25 and we set $w = 3$. In this case, the optimum is isolated and its importance is decreasing with the height of the second largest peak. In Figure 8.2 a) and c) we show results for high mutation and swapping rates, 0.5; in Figure 8.2 b) and d) we have low mutation and swapping rates 0.125. We set the low mutation rate for the cycle $S_{m,u} \times S_{unif}$ to 0.125. Again, we reduce the computation costs by grouping individuals with the same number of ones and zeros in one individual since these individuals have the same fitness value and therefore, the same acceptance probability.

In Figure 8.2 a) and b), for $w = 2$, the basin of attraction is equal for the two peaks. Then, we obtain the highest eigenvalues, and thus the lowest performance, for all the four algorithms. Here we have the largest amount of low fit states that separate two narrow regions with fit individuals; a random sampler, see MIC with mutation rate of 0.5 in Figure 8.2 b), is the best algorithm since it covers a large area with low equal values in short time. For the other values of w, the basin of attraction of one of the peaks is wider than the basin of attraction of the other peak; the narrower one region is, the harder

to find and sample it. For $w = \{1, 3\}$ we have the lowest eigenvalues and, furthermore, the highest difference between the algorithms. The non-recombinative (E)MCMCs do well because the narrow peak is reduced now to one point. The recombinative EMCMCs do better than the non-recombinative (E)MCMCs with the same acceptance rule because recombination generates with higher probability more fit individuals by combining the two building blocks of this function, In Figure 8.2 c) and d), we observe that the performance of all the (E)MCMC algorithms varies very little with the height of the second largest peak h_2. Thus, these eigenvalues are (approximatively) the same with the eigenvalues for $w = \{1, 3\}$ from Figure 8.2 a) and b).

To conclude this example, we observe that due to the structure of the problem recombinative EMCMCs have provably a better performance than the non-recombinative EMCMCs. The performance of MCMCs are diminished by the coupled acceptance rule A_C; *MIC* is sampling more efficient than *mut*+A_C and *rEMCMC* is better than *rEMCMC* + A_C. The mutation rate greatly influences the performance of non-recombinative MCMCs; a high mutation rate decreases the performance of the algorithm. The swapping probability influences less the efficiency of the recombinative EMCMCs. *rEMCMC* and *rEMCMC* + A_C perform best for high swapping probabilities, whereas *MIC* and *mut*+A_C perform best for low mutation rates.

8.2 Experiments with discrete space EMCMCs

In this section we compare various (E)MCMCs for discrete search spaces. In the following, we have performed experiments with the *binary quadratic programming problem* (BQP) to show, on a more elaborated example, that recombination can improve the performance of EMCMCs.

The fitness function of an individual x is $f(x) = \sum_{j=1}^{\ell} \sum_{k=1}^{\ell} Q[j][k] \cdot x[j] \cdot x[k]$, where $Q[j][k]$ is the element on the j-th row and on the k-th column of a matrix Q of integers, both positive and negative. Then, Qs size is $\ell \cdot \ell$.

The interaction between two or more positions of the BQP problem depends on the matrix Q's density, which is defined as the number of non-zeros elements divided by the number of total number of elements in the matrix. The density is then between 0 and 1, where 0 means no interaction between positions and 1 means maximum interaction - that is every position depends on every other position. For our experiments, we generate random matrices with density 0.1.

These fitness values are positive and negative. However, the probabilities of a distribution can be only greater than 0. Therefore, we add to all the fitness values a fixed positive integer $transl$; every value that now is equal or below 0 is assigned with the value 0.01. The unnormalized probabilities are $P'(x) = f(x) + transl$, when $f(x) > transl$ and, otherwise, $P'(x) = 0.01$.

8.2.1 Sampling from BQP

In this section, we show that recombination can improve sampling. We first discuss the experimental methodology available to measure and compare the performance of EMCMCs. Second, we show experimental results on a BQP problem on 20 bits. By expanding the target distribution, we are able to compute the distance between this distribution and the true distribution. At last, we show results on a larger search space, for $l = 100$ bits. Unlike for the previous example, we are not able to expand this distribution, and therefore we are constrained to use less precise methods to assess the performance of (E)MCMCs. For both experiments, we compare the five (E)MCMCs algorithms described above: three non-recombinative (E)MCMCs - that are one long chain *MCMC*, *MIC* and *mut* + A_C - and two recombinative EMCMCs - that are *rEMCMC* and *rEMCMC* + A_C.

Experimental methodology

To asses the efficiency of various EMCMCs we focus on monitoring how fast an MCMC is mixing and how well the samples spread over the entire target distribution after a fixed and rather small number of generated individuals. There is no generally acknowledged methodology on measuring how "close" a set of samples generated with a real-coded MCMC is to the true target distribution. Wolpert and Lee [92] argue that a good approach is to use the Kullback-Leibler (KL) distance between a discrete approximation of the sampled distribution and a discrete approximation of the true distribution.

To measure the speed with which an algorithm samples the search space, Roberts et al. [77] recommend to monitor the acceptance probability of an algorithm. They analytically and experimentally study the behavior of a standard MCMC using a normal distributed mutation with fixed and equal variances in all dimensions. The target distribution is a multivariate normal distribution with standard deviation of 1.0 in all dimensions and no correlations. They conclude that a very high or very low acceptance rate of the MCMC indicates slow mixing, and a good acceptance rate is between $0.2 - 0.5$. A high acceptance rate and a high performance (e.g. the KL-measure close to 0) indicates a well performing algorithm that mixes fast. Analytically computing the optimal acceptance probability is only feasible for very simple target and proposal distributions and when using the Metropolis acceptance rule. Here, we restrict ourselves to experimentally monitoring the acceptance probability.

For the tested recombinative EMCMCs, we have good performance (e.g. KL distance) even for very high acceptance rates that shows that recombination can improve the mixing of MCMCs. Furthermore, we show that algorithms with similar acceptance probabilities can have rather different performance.

A 20 bits BQP

For the first experiment, we set the string length to $\ell = 20$ and $transl = 50$. Since the Q's density is 0.1, only 40 elements of Q have non-zero values. The non-zero integers are generated at random from the interval $[-100, 100]$. For the generated matrix Q, we have found the maximum fitness value 146; when this is translated, the maximum unnormalized probability value is 196. We group the individuals with the same value to generate the histogram and we also store the number of individuals with the same value.

We set the population size $N = 20$. Each generation, all individuals are randomly coupled in $N/2$ pairs such that each individual belongs to exactly one pair. We have performed experiments for various mutation rates (from 0.05 to 0.5) and swapping probabilities for the uniform recombination (from 0.05 to 0.5). With each algorithm, we generate 20000 individuals; our measurements are averaged over 10 runs. We throw away the first 10000 generated individuals to diminish the impact of the starting points over the performance of the algorithms. This is called the *burn-in* period. Thus, in total, we sample 10000 "useful" individuals from which we generate Table 8.1 and the graphs from Figure 8.3.

The search space is 2^{20}. By expanding the target distribution, we are able to compare the frequencies of samples generated with the tested (E)MCMCs with their value in the true distribution. In Table 8.1, we compute the KL distance and acceptance ratio for the five (E)MCMCs. The best algorithm, with the lowest KL distance and the highest acceptance ratio, is the recombinative EMCMC, *rEMCMC*. The only difference between *MIC*, the algorithm with second best KL distance, and *rEMCMC* is that *rEMCMC* uses recombination and *MIC* does not. Further, we observe that the two recombinative EMCMCs, that are *rEMCMC* and *rEMCMC*+A_C, have a higher acceptance probability than the three non-recombinative EMCMCs, that are *MCMC*, *MIC* and *mut*+A_C. That indicates

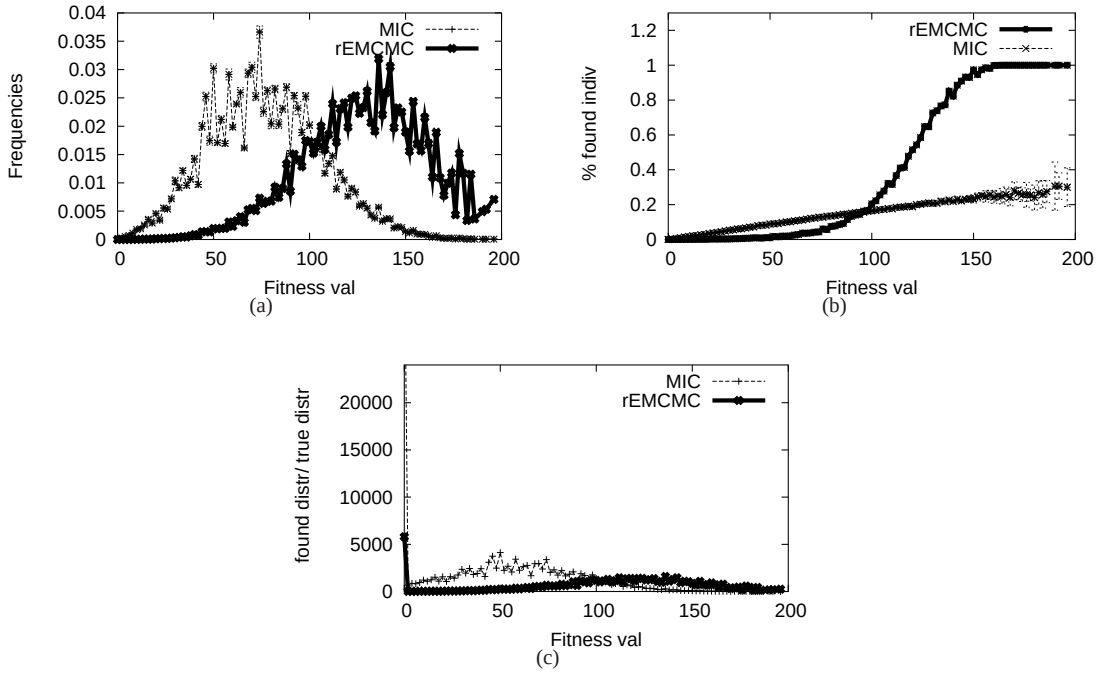

Figure 8.3: a) The frequencies and b) the percent of found solutions for each probability value for *MCMC* and *rEMCMC* on BQP on 20 bits; c) how many times the sampled frequencies differs from the true distribution.

that the recombinative proposal distribution $S_{m,u} \times S_{unif}$, by exploiting the commonalities of the search space, is a "better" proposal distribution than $S_{m,u}$.

Furthermore, as we already observed in the analytical experiments, the coupled acceptance rule A_C has a negative influence over both recombinative and non-recombinative EMCMCs. Even though using A_C, the recombinative *rEMCMC*+A_C has the third best KL distance and the second acceptance ratio, whereas *mut*+A_C is the worst algorithm of all. We explain the good behavior of *rEMCMC*+A_C by synchronizing the individuals in the family with the uniform recombination: children that inherit the common parts of their parents have similar fitness with the parents and the algorithm accepts more individuals. In opposition, uniform mutation independently proposes two individuals in random directions; then, if one of candidates has very low fitness, there is a big probability that both children are rejected. As a consequence, *mut*+A_C has a low acceptance rate and, thus, performance.

In accordance with the analytical results from the previous section, we observe that *MIC*, by using populations of MCMC chains has a lower KL distance than the standard *MCMC*. Note that the acceptance ratio for these two algorithms is the same, but their KL distance quite different.

In Figure 8.3, we show experimental results for the two most performant (E)MCMCs presented in the previous section: *MIC* and *rEMCMC*. By using recombination, *rEMCMC* is a better sampler than *MIC* is: in frequencies, *rEMCMC*, see Figure 8.3 a), is closer to the true target distribution than *MIC* is. If *rEMCMC* samples with predilection in the high values of the target distribution, in opposition, *MIC* samples the most the low fit individuals. Furthermore, *rEMCMC* finds more higher probable solutions than *MIC*, see Figure 8.3 b). Overall, in Figure 8.3 c), we notice that the distribution sampled

alg.	KL dist	accept prob
mut+A_C	$(1.30 \pm 0.70) \cdot 10^{-4}$	0.17 ± 0
MCMC	$(1.16 \pm 0.93) \cdot 10^{-4}$	0.28 ± 0.01
rEMCMC+A_C	$(0.75 \pm 0.50) \cdot 10^{-4}$	0.54 ± 0
MIC	$(0.68 \pm 0.33) \cdot 10^{-4}$	0.28 ± 0
rEMCMC	$(0.50 \pm 0.19) \cdot 10^{-4}$	0.78 ± 0.01

Table 8.1: Efficiency of (E)MCMCs for a BQP on 20 bits: the KL distances and acceptance probabilities

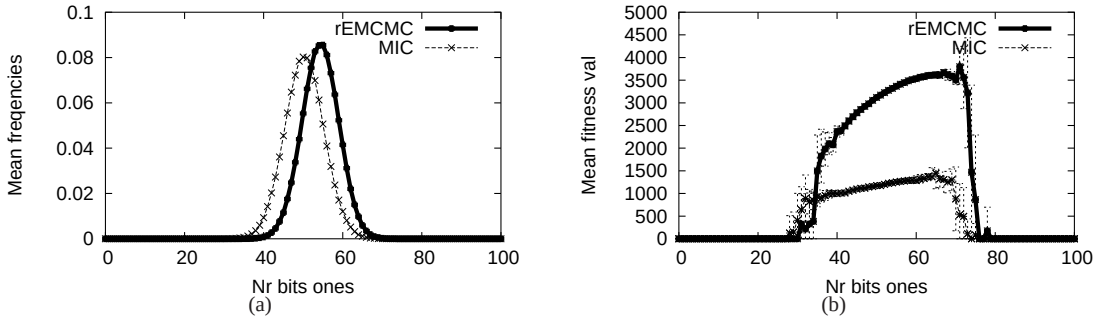

Figure 8.4: a) The frequencies and b) the average of the found solutions for each number of ones for *MCMC* and *rEMCMC* on BQP on 100 bits.

with *rEMCMC* is closer to the true distribution than *MIC* is. These results are in concordance with the ones in Table 8.1 from which we conclude that *rEMCMC* is the most performant algorithm for this particular problem by proposing individuals with recombination.

A 100 bits BQP

We now show that recombination can improve mixing on BQP with string size $\ell = 100$, for which it is impractical to generate the target distribution. In this experiment, we cannot compute the KL distance. Furthermore, we do not know the maximum value of this function or if the values are uniformly distributed in some interval. For this experiment, we compare *MIC* and *rEMCMC*. Assuming, that the unnormalized values of the distribution are in a very large range we group our samples using the individual's number of ones to compare two (E)MCMCs algorithms. Given this grouping, we compute the frequencies, Figure 8.4 a), and the mean value, Figure 8.4 b), for each such a group.

Again, we set the density of Q to 0.1; thus, approximatively 1000 elements of Q are non-zero. We generate these non-zero integers with a uniform random distribution from the interval $[-100, 100]$. To generate a distribution with positive values, we set $transl = 1250$. We set population size $N = 100$ and, each algorithm we run 10 times. With each algorithm, we generate 100000 individuals which we throw away, and we use the next 100000 generated individuals. Again, we vary the mutation rate and swapping rate from 0.05 and 0.5. In Figure 8.4, we show results for *MIC*'s mutation rate 0.2, and *rEMCMC*'s swapping probability 0.5 and mutation rate 0.01. Then, *MIC* has an acceptance rate of 30%, whereas *rEMCMC* an acceptance rate of 78%. We mention that we have performed

experiments with various mutation and swapping probabilities but the results are not very different from the ones we currently show.

In Figure 8.4 a), we notice that *rEMCMC* samples slightly more individuals with a higher number of ones than *MIC* does. Except that, the distributions sampled by *MIC* and *rEMCMC* are similar and both are sampling especially from individuals with half number of zeros and ones indicating that the target distribution is symmetrically distributed around these individuals. Despite that, the mean values of the sampled individuals are remarkably larger for *rEMCMC* than for *MIC*. It seems that 100000 individuals are not enough for *MIC*'s burn in whereas for *rEMCMC* it is. We also have performed experiments with single chain *MCMC*; we mention that the mean values are worse than of *MIC*. We explain that by the shape of this BQP: a lot of peaks with many low fit individuals. We therefore consider that *MIC* mixes slower than *rEMCMC*: $N = 100$ is not large enough to cover the number of these peaks and thus *MIC* will always have the problem to escape from these peaks to find the other useful ones. To sample the same amount of individuals, an increase in population size must be corroborated with an decrease in the *MCMC*'s time to run. The less time we allow an *MCMC* to run, the worse an *MCMC* samples from the search space and eventually, when population size goes to infinity, *MIC* is just a random sampler.

8.2.2 Optimization from BQP

In this section, we show with the same 20-bits BQP problem of Section 8.2.1 that recombination, in combination with elitist coupled acceptance rule and the FOT temperature scheduler can outperform SA. We monitor how many different individuals the EMCMC algorithm discovers in a limited time, as a measure of exploration power, and how fit these individuals are, as a measure of exploiting power of the algorithm. We also monitor the frequencies with which an algorithm samples from the distribution and its relationship with the true distribution; for *ECA+FOT* we also show the difference between the BQP's distribution and the sampled frequencies.

We compare two algorithms: a population based non-recombinative *SA* and *ECA+FOT*. In fact, we have tested three other algorithms for optimization: a long chain *SA*, a population-based *PT* where chains swap individuals in-between them, and a recombinative *PRSA*. Since, we could not notice a visible distinction between the recombinative and non-recombinative *SA* and *PT*, we have chosen to present results only for the two algorithms *SA* and *ECA+FOT*.

For *SA*, we propose individuals with the uniform mutation, $S_{m,u}$, where the mutation rate is $1/\ell$, and $\ell = 20$ the string size. We accept individuals with the standard Metropolis acceptance rule A. For both algorithms, we use a logarithmic annealing schedule, where we set the maximum temperature $Temp^{max} = 10$ and the minimum temperature $Temp^{min} = 0.01$. We set the population size to $N = 20$, and the number of generations to 10000; we run each algorithm 10 times. For *ECA+FOT*, after assigning to each of the current individual states the temperatures according with their probability, we randomly pair each individual in exactly one family. *ECA+FOT* proposes individuals with $S_{m\times unif}$ as the other EMCMCs algorithms tested, and accepts/rejects them with the ECA acceptance rule.

In Figure 8.5 a) and b), we observe that, like for sampling, the recombinative algorithm *ECA+FOT* finds more often more various probable solutions than the non-recombinative *SA*. Figure 8.5 c) compares the sampled distribution with the true distribution; clearly, the distribution sampled with *ECA+FOT* is closer to 1, and thus to the target distribution, than the distribution sampled with *SA*.

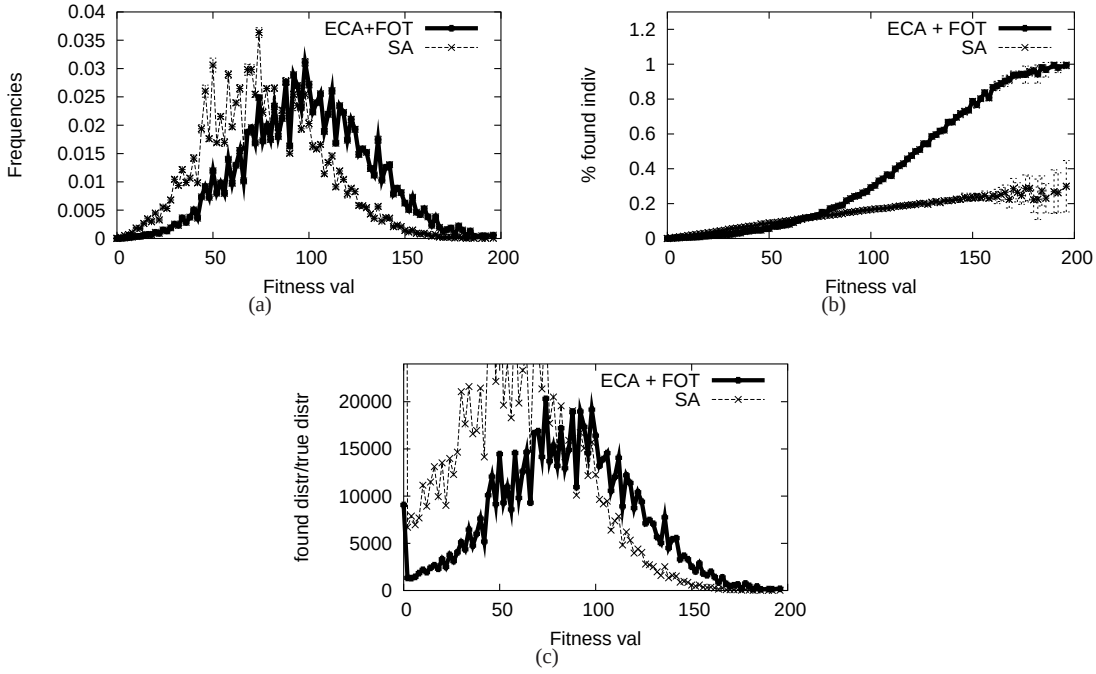

Figure 8.5: a) The frequencies and b) the percent of found solutions for each probability value for *SA* and *ECA+FOT* on BQP on 20 bits; c) how many times the sampled frequencies differs from the true distribution.

8.3 Experiments with real-coded EMCMC

In this section, we experimentally compare recombinative and non-recombinative MCMCs on real-coded search spaces. We consider two distributions: a multi-variate normal distribution and a mixture of multi-variate normal distributions approximating circular distributed data. We are particularly interested in multi-dimensional target distributions with correlations between the dimensions. To keep the performance measurements tractable, we restrict ourselves to $2D$ and $3D$ spaces. Multivariate normal distributions are commonly used for testing and comparing MCMCs due to their simplicity and elegant mathematical properties. In the second test distribution the correlations are introduced by positioning the component distributions of the mixture on a circle.

We show experimentally that coupled acceptance comes at a severe cost: when applied with mutation the resulting MCMC shows worse performance than an MCMC with mutation and the standard Metropolis acceptance rule. However, the experiments also show that the gains obtained by using recombination can make up for the performance loss suffered by applying the coupled acceptance rule.

We also compare the symmetrical and non-symmetrical, simplex geometrical recombinations. To test the recombinative EMCMCs, we have constructed the *amoeba* EMCMC which is a population based MCMC with $p = \ell + 1$ individual states that are transformed with simplex geometrical recombination operators (recall that ℓ is the dimensionality of the target distribution). To achieve detailed balance, the generated states are accepted or rejected with the standard acceptance rule if only one

individual state is proposed, and with the coupled acceptance rule if more than one individual is proposed. We show that amoeba EMCMCs outperform the standard MCMC and DE-MC [87] with population sizes comparable with the dimensionality, $N \approx p$. For this population size, the computational effort of DE-MC's proposal distribution at population level is similar with the ones of the symmetrical simplex geometrical recombination operators $\mathcal{O}(\ell \cdot N)$.

We use the same experimental methodology as in Section 8.2.1. The amoeba EMCMCs we have tested show good performance even for very high acceptance rates indicating that the simplex geometric recombination operators can improve the mixing of MCMCs. Furthermore, we show that algorithms with similar acceptance probabilities can have rather different performance. Because we sample target distributions whose parameters were set apriori we can approximate from the sampled states the correlations between the dimensions and the covariances of the dimensions. As a measure of the efficiency, we observe how close the approximated correlations and covariances are from the true ones.

8.3.1 Multi-variate normal distribution

A multi-variate normal distribution on ℓ dimensions with mean vector μ and covariance matrix Σ has the density

$$P'_{(\mu,\Sigma)}(x) = (2 \cdot \pi)^{-\frac{\ell}{2}} \cdot |\Sigma|^{-\frac{1}{2}} \cdot \exp\left(-\frac{1}{2} \cdot (x-\mu)' \cdot \Sigma^{-1} \cdot (x-\mu)\right)$$

where $|\Sigma|$ is the determinant of the covariance matrix and $(x-\mu)'$ is the transpose of the vector $(x-\mu)$. We consider a $3D$ space, and we set $\Sigma = \begin{bmatrix} 1 & 0.28 & 0.08 \\ 0.28 & 0.1 & 0.03 \\ 0.08 & 0.03 & 0.01 \end{bmatrix}$. Then, the correlation between the first and the second dimension is 0.9, the correlation between the first and the third dimension is 0.8, and the correlation between second and third dimension is again 0.9. We set $\mu = 0$. The fitness function is $f(x) = P'_{(\mu,\Sigma)}(x)$. Note that computing such fitness functions consumes most of the total computing time of the algorithm since it is at least quadratic with the number of dimensions (multiplication of vector of size ℓ with matrix of size $\ell \times \ell$ and with vector of size ℓ). We test MCMC algorithms that scale linearly with the number of dimensions and number of parents. Therefore, we consider that generating a fixed number of individuals for each MCMC provides a fair comparison of the algorithms.

We generate 20000 individuals and we run each algorithm 50 times to compute the mean and standard deviation for the mentioned efficiency measures: KL-distance, acceptance probability, correlation and covariance of the covariance matrix. We test how significant the differences between the values are with Wilcox test, where we consider a $p = 0.95$ confidence interval. We throw away the first 10000 generated individuals to diminish the impact of the starting points - the so called *burn-in* period - on the performance of the algorithm. Thus, in total, we sample 10000 "useful" individuals each run.

When the MCMC samples a state with very low probability, a numerical error might occur when computing the acceptance ratio. In addition we need to establish finite bounds on the sampling space. Therefore, we only consider fitness values above 0.01. When a proposal distribution proposes an individual state with a fitness value smaller than 0.01, the proposed individual is rejected. When we compute the KL measure, we consider the bins that have at least one individual above the minimum value 0.01. The maximum fitness value found was around 5.2. This way, we also bound the time to convergence for an MCMC: the probability to visit the minimum fitness states is about 520 times smaller than the probability to visit the most probable states. This truncation of the search space

has little influence over the values of the multi-variate normal distribution (e.g. variance of the first dimension is 0.99 instead of 1.0, and the correlation between the first and the second dimension is 0.9 as in the covariance matrix).

We construct a histogram of the distribution by splitting the interval $[-4, 4]$ in each dimension into 60 sub-intervals of equal size, forming 60^3 bins. To approximate the true distribution we generate in each bin 10 equally distanced states. Therefore, in total, we generate $60^3 \cdot 10^3$ states - most of them are empty since their value is below 0.01 - and we compute the normalized mean value of the 60^3 bins. To generate the histogram of the sampled distribution, we split the search space into 60^3 bins and we compute the frequencies in each bin of the sampled individuals. When one of the bins is not represented by a sample, we just ignore that bin in computing the KL-distance. We store in a different variable the number of bins that have no sample. After throwing away the bins that contain only individuals with fitness below 0.01, there remain 704 bins to approximate the target distribution.

The tested MCMCs. We compare seven MCMCs: three non-recombinative MCMCs and four recombinative EMCMCs.

1. *MCMC*: a standard, single algorithm that proposes new states with the normal distributed mutation $S_{m,n}$ and accepts (rejects) them using the Metropolis acceptance rule A.

2. *MIC*: multiple independent MCMCs that propose new states with the normal distributed mutation $S_{m,n}$ and accept (reject) them using the Metropolis acceptance rule A.

3. $mut + A_C$: a non-recombinative population-based MCMC that proposes each generation N new states with the normal distributed mutation $S_{m,n}$ and accepts (rejects) all of them using the coupled acceptance rule A_C.

4. *PCTX*: a mixture MCMC where 50% of the time individuals are proposed with the normal distributed mutation $S_{m,n}$ and accepted with the Metropolis acceptance rule A, the other 50% of the time $N = p$ individuals are proposed with symmetrical, parent centric, translation recombination, S_{pctx}, from p parents and accepted with the coupled acceptance rule A_C.

5. *aEMCMC*: amoeba EMCMC is a mixture MCMC where 50% of the time individuals are proposed with the normal distributed mutation $S_{m,n}$ and accepted with the Metropolis acceptance rule A, the other 50% of the time individuals are proposed with equal probability with the four symmetrical, simplex geometric recombinations that generate one child: rotation S_{rot}, differential recombination S_{dif}, scaling S_{scl} and reflection S_{ref}. The proposed individual is accepted (rejected) with the Metropolis acceptance rule A.

6. *naEMCMC*: the non-symmetrical amoeba EMCMC is similar to *aEMCMC* but now the recombination operators are the non-symmetrical uphill, simple, geometrical recombinations instead of the symmetrical ones. As before 50% of the time we generate individuals with the normal distributed mutation $S_{m,n}$ and accept (reject) them with the Metropolis acceptance rule A, the other 50% of the time we generate individuals with equal probability with the four uphill recombinations: uphill rotation $S_{up,rot}$, uphill scaling $S_{up,scl}$, uphill reflection $S_{up,ref}$, and uphill differential recombination $S_{up,dif}$. These recombinations are not symmetrical and we need to accept (reject) the proposed individual with the coupled acceptance rule to achieve detailed balance.

7. *DE-MC*: finally, we also include the differential evolution MCMC algorithm [87].

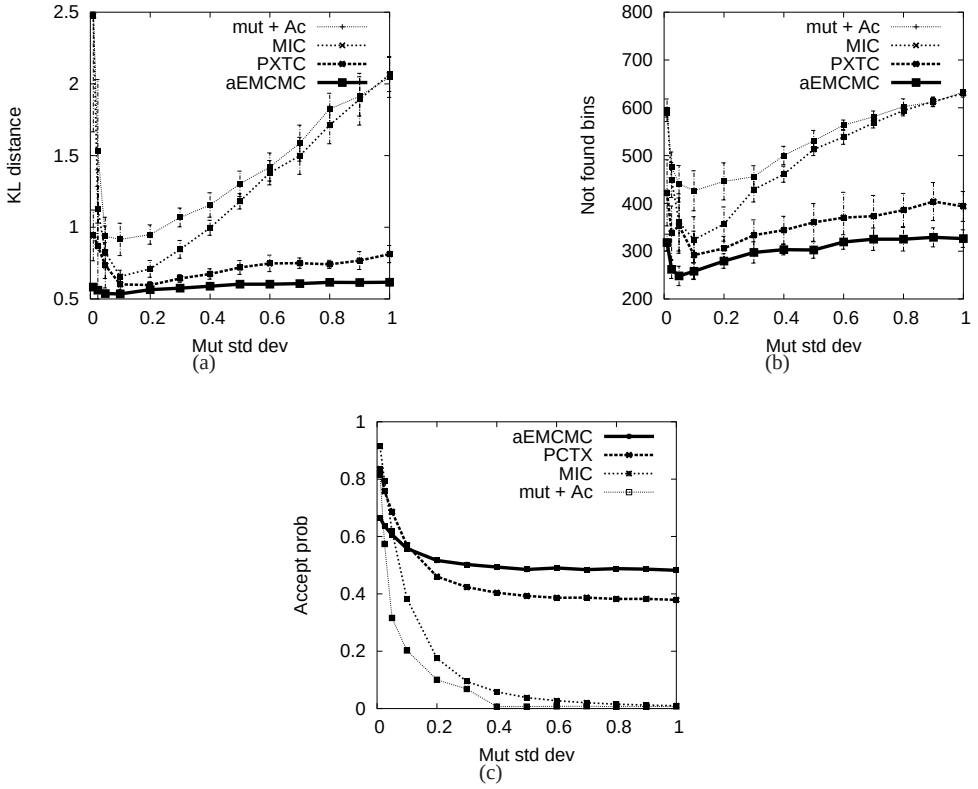

Figure 8.6: (a) KL measure, (b) the number of the bins not represented by a sample and (c) the corresponding acceptance ratio when the mutation's standard deviation is varied.

Setting the parameters. To set the parameters of the above MCMCs we have run a number of experiments. To find the optimal standard deviation, σ_m, for the normal mutation, $S_{m,n}$, which is used by all algorithms except *DE-MC*, we vary σ_m between 0.01 and 1.0. In Figure 8.6, we observe that the performance of non-recombinative MCMCs depends a lot on the value of σ_m. The recombinative EMCMCs are much more robust for this value. It is interesting to see that all algorithms perform best when $\sigma_m = 0.1$ and we will use this value in the experiments. Furthermore, it can also been seen that the KL-distance is lower and the acceptance ratio is higher for the recombinative EMCMCs compared to the non-recombinative MCMCs. We can also observe that the worse KL-distance of *aEMCMC* is lower, thus better, than the best KL-distance of non-recombinative EMCMCs. Note that the number of bins not represented by a sample follows the same pattern like the KL distance. *aEMCMC* has also the most visited bins and the lowest KL-distance. Thus, *aEMCMC* covers the search space best, and also samples the target distribution best. In Table 8.2, we show the results of the tested algorithms for the best parameters.

For S_{pctx} from the *PCTX* algorithm, we sample the statistical variable γ_i for the direction $\overrightarrow{a}_0$ with the normal distribution $\mathcal{N}(0, 0.3)$ and for the direction $\overrightarrow{a}_1$ with the distribution $\mathcal{N}(0, 0.05)$. We have varied the standard deviation of the $\overrightarrow{a}_0$'s normal distribution from 0.1 to 1.0 with a step 0.1; we have obtained similar KL-distances (e.g. 0.59 ± 0.04). We notice that the standard deviation for the

direction $\overrightarrow{a}_1$ should be rather small for a performant algorithm and the performance of *PCTX* varies more with its variation; it seems that this vector is generally larger than $\overrightarrow{a}_0$. For $\gamma_i \approx \mathcal{N}(0, 0.01)$ and $\gamma_{i+1} \approx \mathcal{N}(0, 0.3)$, the KL-distance is 0.77 ± 0.1 and the acceptance rate is 0.37 ± 0.

For S_{rot} from *aEMCMC*, we sample β_i for the rotation angle with $\mathcal{N}(0, 0.3)$. Again, the KL-measure does not vary much (e.g. the highest KL-measure is 0.58 ± 0.03) when the β_is standard deviation is in-between 0.1 and 1.0. For S_{diff} from *aEMCMC*, we sample γ_i with the normal distribution $\mathcal{N}(0, 0.3)$. The highest KL-measure when the γ_is standard deviation is in-between 0.1 and 1.0 is 0.57 ± 0.02. For S_{scl} from *aEMCMC*, we sample γ_i with the normal distribution $\mathcal{N}(0, 0.3)$. The highest KL-measure is 0.65 ± 0.02 when this γ_is standard deviation is in-between 0.1 and 1.0.

We have set similar parameters for the uphill recombinations from *naEMCMC*. The performance of this algorithm, however, varies much more with the values of σ_m than *aEMCMC*. The worst performance - that is the KL distance of 1.02 ± 0.05 and the acceptance rate 0.70 ± 0 - we obtain for $\sigma_m = 1.0$. Again, for the same σ_m, these performances are better than of the MICs but worse than of *PCTX* and *aEMCMC*; but the acceptance rate is higher than with any other tested algorithm. Like for the other recombinative EMCMC, the performance of *naEMCMC* does not vary much with the variation of the parameters for rotation and differential recombination. For this particular algorithm, we found the best results when generating individuals with all the uphill simplex geometrical recombination with equal probability; increasing the percentage of one operator generates EMCMCs with worse performance.

For *DE-MC*, we have chosen the parameters as indicated by their authors. Then, the acceptance ratio for a three dimensional function should be around 0.30; we set the constant $\gamma_i = 1$ and the uniform mutation $\gamma_m \in [-0.0001, 0.0001]$. The population size is $N = 6$, since $N \approx p$ as indicated before. This performance can be improved by setting a larger population size, or including more mutation.

The (in)efficiency of the coupled acceptance rule. For this experiment, we show that the algorithms that accepts the proposed individuals with the coupled acceptance rule are rather inefficient when used only with normal mutation and they do not scale with the number of coupled individuals. We vary the population size, and thus the number of individuals that are all accepted/all rejected with the MCMC algorithms *mut* + A_C and *PCTX*, from 2 to 8. Figure 8.7 shows that when only mutation is used with coupled acceptance, *mut* + A_C we obtain, at far, the worst algorithm. Whereas with *PCTX*, although we use the same acceptance rule, but also recombination, the algorithm outperforms even *MIC*. We explain the poor behavior of $mut + A_C$ by not being able to accept all the individuals unless most of them improve or do not alter the current individuals. If one of them has very low fitness, there is a big probability that all of them are rejected. We explain the good behavior of the recombinative EMCMC, *PCTX*, by synchronizing the individuals in the family with the parent centric translation recombination: children are sampled in the neighborhood of their parents, they have similar fitness and the algorithm accepts more individuals. Again, the number of bins not represented by a sample follows the same pattern like the KL distance. Thus, *PCTX* covers the search space best, and also samples the target distribution best. We observe, that the higher the population size, the worse the acceptance probability, see Figure 8.7 (c). Therefore, we expect that, eventually, for very large populations, the performance of *PCTX* also decreases below the performance of *MIC* whose performance varies the least with the number of individuals chains in MCMC. In Table 8.2, we show that *aEMCMC* outperforms *PCTX* by using the standard acceptance A instead of A_C.

Recombinative vs. non-recombinative EMCMCs. In Table 8.2, we compare the six tested MCMCs after 10000 generated individuals and 50 runs. The best algorithm of all is *aEMCMC* and the worst is *mut* + A_C. Note that, even though it uses coupled acceptance rule, *PCTX* is the third best algorithm; recall that *PCTX* is also the fastest from all recombinative EMCMC since, its computational effort,

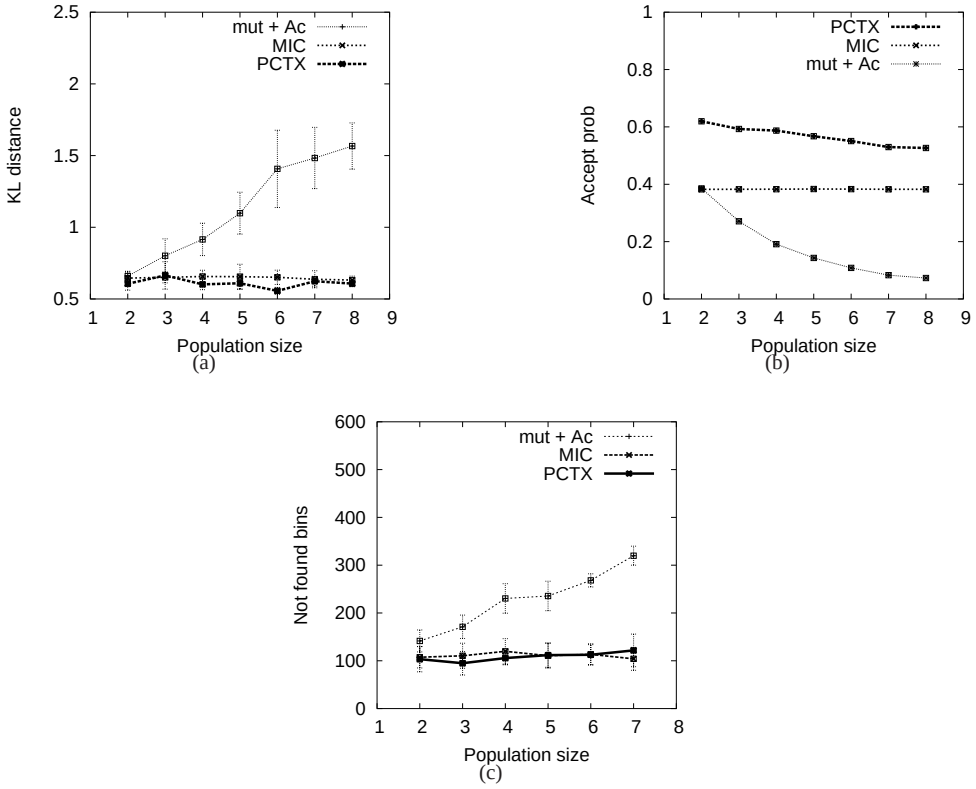

Figure 8.7: (a) KL measure, (b) the number of bins not represented by a sample and (c) the corresponding acceptance ratio when the population size is varied.

is equally to mutation. With Wilcox test, we found all results significantly different with $p = 0.95$, except for *MIC* and *MCMC*. Figure 8.8 shows how the KL distance of the MIC and aEMCMC algorithms decreases with the increase of the individuals generated. We observe that *aEMCMC* outperforms *MIC* even after 60000 generated individuals. We conclude that the recombinative EMCMCs mixes faster than non-recombinative MCMCs for the multivariate normal distributions.

8.3.2 Mixture of bivariate normal distributions

For our last experiment, we compare again the seven MCMCs from above: *MCMC*, *MIC*, *mut* + A_C, *DE-MC*, *PCTX*, *naEMCMC* and *aEMCMC*. Our function is now a mixture of 2D normal distributions, see Figure 8.8 (b), with 11 normal distributions with correlations between dimensions between 0.1 and 0.9 and variances between 0.01 and 0.2. If we consider $P\prime_{(\mu_i,\Sigma_i)}(x)$ the density of the i-th bivariate normal distribution, the fitness function now is $f(x) = \sum_{i=1}^{11} P\prime_{(\mu_i,\Sigma_i)}(x)$. Like in the previous experiment, we consider only the fitness values above the value 0.01; the maximum value is now around 4.5. To compute the histogram, in each dimension, we split the interval $[-4, 4]$ into 120 sub-intervals of equal size, creating 120^2 bins. To approximate the true distribution, in each dimension, we split each bin into 10 equal sub-sub-intervals, resulting in $(120 \cdot 10)^2$ bins. We found

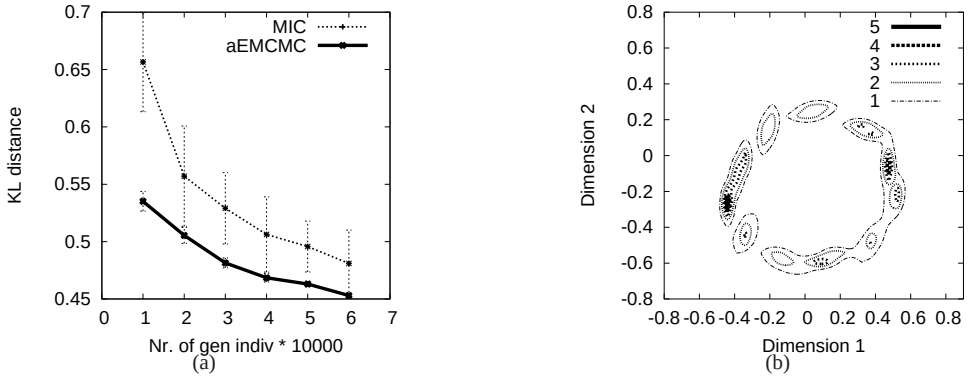

Figure 8.8: a) The improvement of the KL measure when the number of generated individuals increases. b) The mixture of 11 bivariate normal distributions

around 900 "useful" bins whose value is above 0.01. Again, we generate with each algorithm 20000 individuals from which we discard the first 10000; we run each algorithm 50 runs. Since this function is two dimensional, we set the size of the population for *MIC*, *mut* + A_C, *PCTX*, *naEMCMC* and *aEMCMC* to be $N = p = \ell + 1 = 3$. For *DE-MC*, we set $N = 6$.

In Table 8.3, the 2-nd and 3-rd column shows the performance of the seven algorithms for low parameters; that is $\sigma_m = 0.01$ for all algorithms except for *DE-MC* where $\gamma_i = 0.01$. The best performance, see the 4-th and the 5-th columns, we have for $\sigma_m = 0.3$, for all the algorithms but *DE-MC*, and $\gamma_i = 0.3$, for *DE-MC*. In the last two columns we present the algorithms performance for high parameters, $\sigma_m = 1$ and $\gamma_i = 1.0$. For *aEMCMC*, *PCTX* and *naEMCMC* we set all the parameters for recombination to the same value as for mutation; for example, when mutation standard deviation is $\sigma_m = 0.3$, we set the recombinations standard deviation also to 0.3. For *PCTX*, we set $\gamma_{i+1} = 0.05$ for the direction $\overrightarrow{a}_1$.

Note that *aEMCMC* is again the best algorithm since it is slightly, but significantly better than long chain *MCMC* and *MIC* that have a similar performance. Furthermore, all the recombinative EMCMCs, except *DE-MC*, vary less with the setting of the parameter than non-recombinative EMCMCs and their performance is better for the low parameters when the exploration is slow. Now, the second worst algorithm is *DE-MC*; with low mutation and small population size, *DE-MC* samples less efficient than the other algorithms that use much more mutation. We also observe that the algo-

alg.	KL dist	not found bins	accept prob	cov dim 1	cor dim 1 and 2
mut+A_C	0.92 ± 0.11	426 ± 42	0.27 ± 0	0.97 ± 0.44	0.88 ± 0.04
MIC	0.66 ± 0.04	324 ± 48	0.38 ± 0	0.97 ± 0.30	0.89 ± 0.03
MCMC	0.64 ± 0.05	320 ± 50	0.38 ± 0	0.97 ± 0.32	0.89 ± 0.03
naEMCMC	0.62 ± 0.03	257 ± 13	0.79 ± 0	1.0 ± 0.17	0.89 ± 0.02
PCTX	0.59 ± 0.04	299 ± 20	0.56 ± 0	1.1 ± 0.21	0.88 ± 0.02
DE-MC	0.59 ± 0.02	283 ± 11	0.31 ± 0	1.02 ± 0.1	0.89 ± 0.01
aEMCMC	0.54 ± 0.01	262 ± 14	0.54 ± 0.01	1.05 ± 0.09	0.91 ± 0.01

Table 8.2: Efficiency of (E)MCMCs for trivariate normal distribution

	low param		optim param		high param	
alg.	KL dist	accept	KL dist	accept	KL dist	accept
mut+A_C	2.07 ± 0.27	0.47 ± 0	0.72 ± 0.13	0.25 ± 0	1.15 ± 0.19	0.07 ± 0
DE-MC	2.47 ± 0.49	0.87 ± 0	0.58 ± 0.09	0.30 ± 0	1.07 ± 0.03	0.1 ± 0
naEMCMC	0.89 ± 0.04	0.87 ± 0	0.52 ± 0.02	0.75 ± 0	0.72 ± 0.05	0.65 ± 0
PCTX	0.72 ± 0.03	0.67 ± 0	0.45 ± 0.03	0.55 ± 0	0.76 ± 0.03	0.45 ± 0
MIC	2.07 ± 0.27	0.87 ± 0	0.39 ± 0.06	0.43 ± 0	0.90 ± 0.04	0.17 ± 0
MCMC	2.07 ± 0.27	0.87 ± 0	0.39 ± 0.07	0.43 ± 0	0.90 ± 0.04	0.17 ± 0
aEMCMC	0.61 ± 0.03	0.67 ± 0	0.36 ± 0.02	0.61 ± 0	0.59 ± 0.03	0.62 ± 0

Table 8.3: Efficiency of (E)MCMCs for the mixture of bivariate normal distributions

rithms that uses the coupled acceptance rule perform quite poorly, even though, with recombination the performance of *PCTX* is improved over *mut* + A_C, that is the worst algorithm.

Chapter 9

Conclusion

We have investigated how to integrate recombination in population MCMCs to obtain irreducible EMCMCs that converge to the target distribution. First, we have studied the properties of various recombination proposal distributions on discrete spaces. We have investigated how to use the MH acceptance rules with recombination to obtain EMCMCs with detailed balance. We have studied how to design proposal distributions using recombination operators to obtain efficient EMCMCs that adapt their proposal probabilities during the sampling process. By considering the individuals in the current population as points of a geometrical figure, we can exploit correlations present in a real-coded sampling space using linear transformations like rotation, translation, scaling and reflection. In particular, we have proposed and investigated recombination operators that transform simplex geometrical figures. We showed that our proposed recombinations, both discrete and real-coded, can exploit specific correlations, linear and non-linear, between the dimensions of the sampling space. Furthermore, they exploit this information efficiently since the computational cost is linear with the number of dimensions and the number of parents. We have studied how to integrate mutation in these EMCMCs to ensure irreducibility. We have integrated these proposal distributions into the general EMCMC framework. We have shown that, in general, to obtain detailed balance a recombinative EMCMC needs either to accept or to reject all the children proposed by the recombination operator. We have also discussed the conditions under which a recombinative EMCMC, which accepts or rejects individual states with the standard MH acceptance rule, can be used to sample from the target distribution. Analytical and experimental results show how recombination can be used to improve the efficiency of EMCMCs compared to standard MCMC sampling algorithms.

Chapter 10

Discussion and future work

In this thesis we have approached two subjects: the conditional log-likelihood MDL score and the evolutionary MCMC. In the following, we discuss our theoretical and experimental results on the two subjects and we highlight some possible future directions for this research.

10.1 The first part: conditional log-likelihood MDL

In the first part of this thesis, we propose and analyze a conditional log-likelihood MDL score function for learning simple, but performant, Bayesian network classifiers. For our MDL-FS scoring function, we use a two parts MDL: one part is encoding the conditional log-likelihood of a Bayesian network classifier from which we subtract the second part which is encoding the structure of the Bayesian network classifier. The first part of the MDL-FS function indicates how well a Bayesian network classifier represents the data; the more complex is the classifier, the better the classifier models the data and the larger the log-likelihood is. The second part is a penalty term that increases with the complexity of the classifier to prevent overfitting. We have theoretically and experimentally shown that the MDL-FS function is suited for the task of learning simple Bayesian network classifiers. It identifies relevant attributes for the class variable and also strong relationships between the relevant attributes given the class variable. For example, we use a greedy algorithm to add the most relevant attributes and their most strong relationships into the Bayesian network classifier.

Unlike the MDL-FS function, the standard two parts MDL function [10, 38] encodes the log-likelihood of the Bayesian network classifier instead of the conditional log-likelihood of the same classifier. Because that - the MDL function is encoding the joint probability rather than the conditional distribution like MDL-FS - the MDL function is less tailored to learn classifiers. In Section 3.2.1, we show that the MDL function identifies the strong relationships between attributes in the presence of the class variable. We further show that this is not enough to identify certain types of irrelevant attributes. The resulting classifiers are more complex and have lower classification performance than with the MDL-FS function.

Our method experimentally outperforms several feature selection specific algorithms. Using our definition for relevant attributes, we show on an example why our method that uses the MDL-FS function works better than these other algorithms.

Complex Bayesian network classifiers. We have shown that our conditional log-likelihood function exactly represents the conditional probability of the class variable given the attributes for fully connected Bayesian network classifiers and auxiliary networks. Furthermore, we theoretically and

experimentally have shown that the MDL-FS score function performs well for simple Bayesian network classifiers: Naive Bayes and TANs. As future work, we want to test the MDL-FS function on more complex Bayesian network classifiers than TANs. For example, we want to study polytrees that allow more than one parent for an attribute. But, there are no greedy algorithms that generates polytrees of maximum log-likelihood; in this case, a complex search algorithm, like a genetic algorithm, might be efficient in learning polytrees.

Continuous variables. The MDL-FS function is designed for discrete variables. Therefore, in our experiments, we discretize the continuous variables before we start the classification task. The algorithm could be more efficient if the MDL-FS function would handle continuous variables directly rather than first discretize them. As future work, we plan to design an MDL-FS function that deals with continuous variables. We would assume various distributions for the variables (e.g. Gaussian distributions).

Alternative MDL functions. The two part MDL function is considered outdated since there are alternative MDL functions [43] that are generalizations of this two parts MDL. Often, with these alternatives it is hard to learn Bayesian network classifiers. Since these MDL functions encode the joint probability distribution rather than the conditional distribution, we expect that they have a poor feature selection behavior. As future work, we want to design an MDL function that is more general than the two parts MDL but which encodes the conditional probability.

10.2 The second part: evolutionary MCMC

In the second part of this thesis, we have studied how to integrate recombination in population-based MCMC. We were interested in when and why recombination improves the sampling process. We have investigated discrete recombinations that exploit commonalities in the parent states, and real-coded recombinations that bias the search according to correlations and proximity properties present in the parents.

To sample from the target distribution, we need to generate candidate states with a proposal distribution, and accept or reject them with an acceptance rule. We have found a quite restrictive rule for EMCMCs when using recombination as proposal mechanism: to converge to the target distribution either all individuals generated by a recombination operator need to be accepted or all need to be rejected. We have also shown that such an all-or-nothing acceptance rule is less performant than a rule where some children can be accepted and some rejected. As a result, the advantages of recombination as proposal mechanism are somewhat diminished by the need to use a corresponding coupled acceptance rule.

A solution to this problem was to use a specific acceptance rule for which each child generated with recombination can compete against one parent. We have discussed the conditions under which this algorithm converges to the desired target distribution. Furthermore, this approach has led to the best experimental and analytical sampling results we have obtained.

Finally, we discuss a few directions in which this research could be continued.

- **Practical applications.**

  We have tested our EMCMC algorithms in a number of controlled experiments where we knew the target distribution. These controlled experiments allowed us to assess the performance of the algorithms. A next step is to apply the methods discussed in this work to a real word problem. For example, Ising models from theoretical physics use MH algorithms to estimate the magnetic susceptibility [69]. To test EMCMCs on this application, we would need to design recombination operators exploiting the problem particularities of this domain.

- **Adaptive samplers.**

  The proposal distributions for EMCMCs are restricted by the Markov property: the candidate individuals can be generated only by using the information from the current individuals and not from the anterior individuals. When the Markov property does not hold, a sampler needs other mechanisms than detailed balance to converge to the target distribution. Interesting future work would be to investigate new mechanisms to adapt proposal distributions that do not have the Markov property.

- **Sequential Monte Carlo.**

  Sequential Monte Carlo (SMC) are algorithms used to sample from dynamic environments. They are not necessarily Markov chains and thus they do not need the detailed balance condition. Currently, there are a few versions of SMCs which use recombination. However, none of them have provable convergence properties. Future work could study how to integrate recombination into SMC such that they provably sample from the desired target distribution.

Bibliography

[1] D.W. Aha and R.L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI'94 Workshop on Case-Based Reasoning*, pages 106–112, 1994.

[2] E. Alpaydin. *Introduction to Machine Learning*. Mit Press, 2004.

[3] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, pages 5–43, 2003.

[4] S. Baluja and S. Davies. Using optimal dependency-trees for combinational optimization. In *Proceedings of ICML'97*, pages 30–38, 1997.

[5] A. Barron, J. Rissanen, and Y. Bin. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

[6] L. Bauwens, C.S. Bos, H.K. Van Dijk, and R.D. Van Oest. Adaptive polar sampling, a class of flexibel and robust monte carlo integration methods. Technical Report 278, Erasmus University Rotterdam, Econometric Institute, 2002.

[7] J. Besag and P. J. Green. Spatial statistics and Bayesian computation. *Journal of the Royal Statistical Society, Series B*, 55:25–37, 1993.

[8] J.A. Bilmes. Dynamic Bayesian multinets. In *Proceedings of UAI*, pages 38–45, 2000.

[9] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[10] R. Bouckaert. *Bayesian Belief Networks: from construction to inference*. PhD thesis, Utrecht University, The Netherlands, 1995.

[11] G.E. Box and G.C. Tiao. *Bayesian Inference in Statistical Analysis*. Wiley Interscience, 1992.

[12] J. Burge and T. Lane. Learning class-discriminative dynamic Bayesian networks. In *Proceedings of ICML'05*, pages 97–104, 2005.

[13] R. Caruana and D. Freitag. How useful is relevance? In *Proceedings of AAAI technical report for the Fall'94 AAAI Symposium on Relevance*, pages 25–29, 1994.

[14] S. Chen and G. Pitt. Isolating the benefits of respect. In *Proceedings of GECCO*, pages 1601–1602, 2005.

[15] J. Cheng and R. Greiner. Comparing Bayesian Network Classifiers. In *Proceedings of UAI'99*, pages 101–110, 1999.

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

[17] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1:131–156, 1997.

[18] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 10(4):371–395, 2002.

[19] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[20] M. M. Drugan, D. Thierens, and L. C. van der Gaag. MDL-based feature selection for Bayesian network classifiers. pages 99–106, 2002.

[21] M. M. Drugan and L. C. van der Gaag. A new MDL-based feature selection for Bayesian network classifiers. pages 999–1000, 2004.

[22] M.M. Drugan and D. Thierens. Evolutionary Markov chain Monte Carlo. In *Proc. Artificial Evolution'03*, LNCS 2936, pages 63–76, 2004.

[23] M.M. Drugan and D. Thierens. Recombinative EMCMC algorithms. In *Proceedings Congress of Evolutionary Computation, CEC'05*, pages 2024–2031, 2005.

[24] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, New York, 1973.

[25] J.G. Dy and C.E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, 2004.

[26] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

[27] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

[28] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[29] G. Forman. Feature selection: We've barely scratched the surface. *IEEE Intelligent Systems*, 2005.

[30] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[31] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 721–741, 1984.

[32] C. J. Geyer. Markov Chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 156–163, 1991.

[33] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in practice*. Chapman & Hall, 1996.

[34] W.R. Gilks and G. O. Roberts. Strategies for improving MCMC. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, *Markov Chain Monte Carlo in practice*. Chapman & Hall, 1996.

[35] D. E. Goldberg. A note on Boltzmann tournament selection for Genetic Algorithms and Population-oriented Simulated Annealing. *Complex Systems*, 4:445–460, 1990.

[36] R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *Proceedings of AAAI/IAAI'02*, pages 167–173, 2002.

[37] D. Grossman and Domingos. Learning Bayesian networks classifiers by maximising conditional likelihood. In *Proceedings of ICML'04*, pages 361–368.

[38] P. Grunwald. *The Minimum Description Length Principle and Reasoning under Uncertainty*. PhD thesis, Amsterdam University, The Netherlands, 1998.

[39] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003.

[40] Bruce Hajek. Cooling schedules for optimal annealing. *Math. Oper. Res.*, 13(2):311–329, 1988.

[41] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the ICML*, pages 359–366, 2000.

[42] M. H. Hansen and B. Yu. Model Selection and the Principle of Minimum Description Length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[43] M. H. Hansen and B. Yu. Minimum description length model selection criteria for generalized linear models. *Science and Statistics*, 40, 2003.

[44] H.L. Harney. *Bayesian Inference: Parameter Estimation and Decision*. Springer, 2003.

[45] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[46] T. Jebara and T. Jaakola. Feature selection and dualities in maximum entropy discrimination. In *Proceedings of UAI'00*, pages 291–300, 2000.

[47] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant Features and the Subset Selection Problem. In CA Morgan Kaufman Publishers, San Francisco, editor, *Proceeding of the 11th International Conference of Machine Learning*, pages 121–129, 1994.

[48] M. Kearns, Y. Mansour, A.Y. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27:7–50, 1997.

[49] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[50] R. Kohavi and G. H. John. Wrappers for Feature Subset Selection. In *Artificial Intelligence Journal*, pages 273–324, 1997.

[51] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *Proceedings of ICML-96, the 13th International Conference on Machine Learning*, pages 284–292, 1996.

[52] I. Kononenko. On biases in estimating multi-valued attributes. In *Proceedings of IJCAI'95*, pages 1034–1040, 1995.

[53] P. Kontkanen, W. Buntine, P. Myllymaki, J. Rissanen, and H. Tirri. Efficient computation of stochastic complexity. In *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistic*, pages 181–188, 2003.

[54] P. Kontkanen, P. Myllymki, T. Silander, and H. Tirri. BAYDA: Software for Bayesian classification and feature selection. In *Knowledge Discovery and Data Mining*, pages 254–258, 1998.

[55] P. Kontkanen, P. Myllymki, T. Silander, and H. Tirri. On supervised selection of Bayesian networks. In *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 334–342, 1999.

[56] P. Kontkanen, P. Myllymki, and H. Tirri. Classifier learning with supervised marginal likelihood. In *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI'01)*, pages 277– 284, 2001.

[57] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–294, 1994.

[58] P. Langley. Selection of relevant features in machine learning. In *Proceedings of AAAI Fall Symposium on Relevance*, pages 140–144, 1994.

[59] P. Langley, W. Iba, and K. Thompson. An Analysis of Bayesian Classifiers. In *Proceedings of the 10th Conference on Artificial Intelligence*, pages 223–228, 1992.

[60] P. Langley and S. Sage. Induction of Selective Bayesian Classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 399–406, 1994.

[61] K. B. Laskey and J. W. Myers. Population Markov Chain Monte Carlo. *Machine Learning*, pages 175–196, 2003.

[62] F. Liang and W. H. Wong. Evolutionary Monte Carlo: Applications to C_p model sampling and change point problem. In *Statistica Sinica*, pages 317–342, 2000.

[63] F. Liang and W. H. Wong. Real-Parameter Evolutionary Monte Carlo with Applications to Bayesian Mixture Models. *Journal of the Amerian Statistical Association*, 96(454):653–666, 2001.

[64] F. Liang and W.H. Wong. Evolutionary Monte Carlo for protein folding simulations. In *Journal of Chemical Physics*, number 115, pages 3374–3380, 2001.

[65] S. W. Mahfoud and D. E. Goldberg. Parallel Recombinative Simulated Annealing: a Genetic Algorithm. *Parallel Computing*, pages 1–28, 1995.

[66] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[67] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[68] J.A. Nelder and R. Mead. *Computer Journal*, 7:308–313, 1965.

[69] M.E.J. Newman and G.T. Barkema. *Monte Carlo methods in Statistical Physics*. Clarendon Press, Oxford, 2004.

[70] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21:5–20, 2002.

[71] F. Pernkopf and J. Bilmes. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Proceedings of ICML'05*, pages 657–664, 2005.

[72] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.

[73] N.J. Radcliffe. Forma analysis and random respectful recombination. In *Proc. of the Fourth Inter. Conf. on Gen. Alg.*, pages 222–229, 1992.

[74] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[75] J. Rissanen. Hypothesis selection and testing by the MDL principle. *Computer Journal*, 42(4):260–264, 1999.

[76] J. Rissanen. Strong optimality of the normalized ML models as universal codes and information in data. *IEEE Transactions on Information Theory*, 47(5):1712–1717, 2001.

[77] G. O. Roberts, A. Gelman, and W.R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[78] M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53:23–69, 2003.

[79] Y.D. Rubinstein and T. Hastie. Discriminative vs informative learning. In *Knowledge Discovery and Data Mining*, pages 49–53, 1997.

[80] G. Rudolph. Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation*, pages 361–382, 1994.

[81] G. Rudolph. *Convergence Properties of Evolutionary Algorithms.* Verlag Dr. Kovac, 1997.

[82] C. E. Shannon. A mathematical theory of communication. *The Mathematical Theory of Communication*, 1949.

[83] M. Singh and G. M. Provan. A comparison of induction algorithms for selective and non-selective Bayesian classifiers. In *Proccedings of the ICML*, pages 497–505, 1995.

[84] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Opt.*, 11:341–359, 1997.

[85] M. J. A. Strens. Evolutionary MCMC sampling and optimization in discrete spaces. In *Proceedings of ICML*, 2003.

[86] M. J. A. Strens, M. Bernhardt, and N. Everett. Markov chain Monte Carlo sampling using direct search optimization. In *Proceedings of ICML*, 2002.

[87] C.J.F. ter Braak. Genetic algorithms and Markov chain Monte Carlo: Differential Evolution Markov Chain makes Bayesian computing easy. Technical report, Biometris, Wageningen UR, 2005.

[88] D. Thierens and D.E. Goldberg. Elitist recombination: an integrated selection-recombination GA. In *Proceedings of the First IEEE World Congress on Computational Intelligence*, pages 508 – 512, 1994.

[89] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, pages 1701–1762, 1994.

[90] I. Tsamardinos and C.F. Aliferis. Towards principled feature selection: Relevance, filters, and wrappers. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[91] T. van Allen and R. Greiner. Model selection criteria for learning belief nets: An empirical comparison. In *Proceedings of ICML'00*, pages 1047–1054, 2000.

[92] D.H. Wolpert and C.F. Lee. Adaptive Metropolis sampling and optimization with product distributions. Technical report, NASA Ames Research Center, 2005.

[93] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of The Twentieth International Conference on Machine Learning (ICML-2003)*, pages 856–863, 2003.

[94] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Machine Learning Journal*, 5:1205–1224, 2004.

[95] M. Zaffalon and M. Hutter. Robust feature selection by mutual information distributions. In *Proceedings of UAI*, pages 577–584, 2002.

[96] X. Zhu and X. Wu. Class noise vs. attribute noise: a quantitative study of their impacts. *Artificial Intelligence Review*, 22:177–210, 2004.

Summary

In the current society there is an increasing interest in intelligent techniques that can automatically process, analyze, and summarize the ever growing amount of data. Artificial intelligence is a research field that studies intelligent algorithms to support people in making decisions. An example of an artificial intelligent system is an expert system, a program that is able to make deductions and conclusions using an inference algorithm and knowledge stored in production rules. When an expert system is given particular inputs, it can use the inference algorithm and the production rules to automatically infer a diagnosis or other kinds of conclusions. Expert systems exist for more than thirty years now, but a problem of traditional expert systems is that the necessary knowledge should be acquired by a knowledge engineer that interacts with a human domain expert. Therefore the knowledge acquisition process is time consuming, expensive, and complex for constructing useful complex systems. An alternative for traditional expert systems is to use algorithms that are able to induce knowledge when it is given a database of examples for a particular domain. Algorithms that are able to induce knowledge from examples are researched in the field of machine learning. This thesis studies improvements of particular machine learning algorithms.

In the first part of this thesis we describe methods that are able to select useful attributes (or features) that can be used as inputs by a classification algorithm. Classification problems require a knowledge representation that maps a set of inputs to a discrete classification label. An example classification problem is to make a system that determines whether or not a person has a particular disease given a set of attributes or symptoms that describe the status of the patient. In this thesis we focus on Bayesian network classifiers that use Bayesian networks as knowledge representation. There has been a lot of research in constructing Bayesian networks and therefore our focus is on selecting relevant attributes that should be used as inputs for the Bayesian network classifier. It is of course possible to use all available attributes in the stored examples for constructing the Bayesian network classifier. However, selecting a subset of the attributes has a number of advantages. The first advantage is that the learned knowledge representation will be simpler and therefore more readable by human domain experts or users. The second advantage is that by selecting the most important input attributes, the performance of the Bayesian network classifier can be improved since a smaller knowledge representation generalizes better for classifying novel inputs.

For our goal to construct selective Bayesian network classifiers, we propose and investigate a score function that can evaluate Bayesian network classifiers and that indicates the simplest and the most performant classifier. This function is closely related with the minimum description length (MDL) function, often used for learning simple, but performant Bayesian networks. Whereas in Bayesian network classifiers the class variable is the most important variable, in Bayesian networks all the variables are considered equally important. As a consequence, the usual MDL function is well suited for learning Bayesian networks that encode a joint probability distribution over all variables, but is less useful for constructing selective Bayesian network classifiers. In contrast to the normal MDL function, our MDL-FS function encodes the conditional log-likelihood of the Bayesian network

classifier instead of its log-likelihood like the standard MDL function does. We theoretically and experimentally show that our conditional log-likelihood MDL is well suited for constructing simple and well performing Bayesian network classifiers.

In the second part of this thesis we integrate some methods from evolutionary computation into a MCMC sampler. Sampling is related to optimization, but whereas in optimization we are only interested in the state with the highest fitness, in sampling we are interested in the overall probability distribution over states. Sampling is useful to compute particular functions over an input space where different states in the input space have different probabilities. A possible function is to compute the center of mass of the distribution, which cannot be done analytically for complex distributions. Optimization can also be seen as computing the max-function over an input space, but contrary to particular complex sampling functions, computing the max-function only requires to store a single state (the optimum). The goal of sampling is to generate many samples according to some (unnormalized) target probability distribution, and to use the sampled states for further processing. In this thesis we are mostly interested in efficient sampling from a target distribution and less in further processing of the generated states.

To improve MCMC methods that are often used for sampling, we investigate the Evolutionary MCMC (EMCMC) framework, where population-based MCMCs exchange information between the individual states. To ensure convergence to the target distribution, we have to make sure that the EMCMCs are still MCMCs at population level. As starting point, we use the Metropolis-Hastings algorithm where candidate individuals are proposed with some proposal distribution that is not necessarily related with the target distribution. These proposed individuals are then accepted or rejected with some acceptance rule in order to sample from the desired target distribution.

Ideally, a MCMC algorithm proposes individual states (or individuals) directly from the target distribution. Such an algorithm would converge to the target distribution very fast since all states would be proposed in proportion with their target probability. Unfortunately, in practice one cannot sample directly from the target distribution. A standard MCMC typically proposes a new individual with a distribution (e.g. the Gaussian distribution) centred around the parent state. Such a proposal distribution does not use any information about the target distribution, and therefore generates many samples that are not very probable in the target distribution and often rejected by the acceptance rule. We design more 'intelligent' proposal distributions that adapt to the target distribution by using recombination operators. Due to the sampling process the population contains more often individual states with high probability in the target distribution. Our recombination operators exploit the structural information present in the parent states to generate new states which most likely also have a high probability in the target distribution. If the exploration bias of the recombination operators coincides with the structural relationships present in the target distribution then the proposed states will more likely be accepted by the acceptance rule, thus increasing the performance of the EMCMC sampling algorithm. For this aim, we describe a number of different recombination operators for discrete and continuous state spaces, and analyze some of their properties.

We also investigate the interactions between the recombination operators and the acceptance rules. In general, for an EMCMC to converge to the desired target distribution, all the candidate individuals proposed by recombination should be accepted or rejected. Such an acceptance rule is called a coupled acceptance rule, since all proposed states are coupled for being accepted or rejected. This is quite restrictive since some candidate individuals can be very fit - or very probable in the target distribution - and some very poor. The problem is that the coupled acceptance rule cannot accept only the good individuals and reject the worse ones. In Chapter 7 a number of experiments are described, and it is shown that the coupled acceptance rule reduces the performance of an EMCMC. To deal with the problems with the coupled acceptance rule, we study the use of single parent versus child acceptance rules, in which one child proposed with recombination competes against one of the parents in the

acceptance rule, and another child with another parent etc. We prove that under certain conditions the resulting algorithm can still be used to sample from the desired target distribution and does not have the problems of the coupled acceptance rule. The experimental results show that when we compare the coupled versus the uncoupled acceptance rule, the uncoupled acceptance rule leads to much more efficient sampling when used with different EMCMC methods.

Samenvatting

In de huidige samenleving is er steeds meer vraag naar intelligente technieken die de voortdurend toenemende hoeveelheden data automatisch kunnen verwerken, analyseren en samenvatten. Kunstmatige intelligentie is een onderzoeksgebied dat zich bezig houdt met slimme algoritmen die kunnen worden gebruikt om mensen te helpen met het nemen van beslissingen. Een voorbeeld van een kunstmatig intelligent systeem is een kennissysteem, een programma dat in staat is om met behulp van inferentie algoritmen en kennis opgeslagen in productie-regels, automatisch afleidingen te maken gegeven bepaalde gegevens die bekend zijn en ingevoerd kunnen worden. Kennissystemen bestaan al meer dan dertig jaar, maar een nadeel van traditionele kennissystemen is dat kennis verworven moet worden door de interactie van een kennis ingenieur met menselijke domein experts. Het aldus verkrijgen van kennis kost zeer veel tijd en maakt het construeren van complexe bruikbare systemen daarom tijdrovend, duur en complex. Een alternatief voor traditionele kennissystemen is het gebruik van algoritmen die in staat zijn om voorbeelden in de vorm van opgeslagen databestanden te gebruiken en deze om te zetten in bruikbare kennisrepresentaties. Algoritmen die in staat zijn om kennis uit voorbeelden te halen worden onderzocht in het deelgebied van de kunstmatige intelligentie genaamd “Machine Learning“. Dit proefschrift gaat dan ook over verbeteringen van bepaalde machine learning algoritmen.

In het eerste deel van dit proefschrift beschrijven we algoritmen die in staat zijn om bruikbare kenmerken te selecteren welke als input gebruikt kunnen worden voor een classificatie algoritme. Classificatie problemen vereisen een kennisrepresentatie welke een verzameling inputs afbeelden op een discrete output welke de classificatie van de input genoemd wordt. Een voorbeeld classificatie probleem is een systeem dat bepaald of iemand een bepaalde ziekte heeft gegeven de kenmerken of symptomen die een patient beschrijven. In het eerste deel van dit proefschrift focussen we op Bayesiaanse classificatie algoritmen welke een Bayesiaans netwerk gebruiken als kennisrepresentatie. Er is al veel onderzoek geweest naar Bayesiaanse netwerken en daarom is onze focus vooral op het selecteren van relevante kenmerken die als input gebruikt moeten worden. Natuurlijk is het mogelijk om alle input kenmerken die in de voorbeelden zijn opgeslagen te gebruiken, maar het selecteren van een deelverzameling van deze kenmerken heeft bepaalde voordelen. Ten eerste kan de kennisrepresentatie welke geleerd wordt simpeler en daarmee overzichtelijker worden voor menselijke experts en gebruikers. Ten tweede kan er door het selecteren van de meest bruikbare kenmerken de prestatie van de Bayesiaanse classifier verbeterd worden, omdat kleinere kennisrepresentaties beter kunnen generalizeren voor het classificeren van nieuwe inputs.

Voor ons doel om selectieve Bayesiaanse netwerk classifiers te construeren, hebben we een nieuwe score functie bedacht en bestudeerd welke Bayesiaanse classifiers can evalueren waardoor de simpelste en best werkende representatie gekozen kan worden. Onze nieuwe score functie is sterk gerelateerd aan de bekende “minimum description length“ of MDL functie die vaak gebruikt wordt voor het leren van simpele en goed presterende Bayesiaanse netwerken. Hoewel in Bayesiaanse netwerk classifiers de classificatie label het belangrijkste is, zijn in een Bayesiaans netwerk alle

variabelen van belang. Het gevolg hiervan is dat de MDL functie wel geschikt is voor het evalueren van Bayesiaanse netwerken welke een gezamelijke kansverdeling voor alle variabelen coderen, maar minder geschikt voor het construeren van selectieve Bayesiaanse netwerk classifiers. In tegenstelling tot de gewone MDL functie die de gezamelijke kansverdeling codeert, codeert onze MDL-FS functie de condionele kansverdeling van het classificatie label gegeven de andere variabelen. We tonen theoretisch en experimenteel aan dan onze MDL-FS functie erg geschikt is voor het constueren van simpele en goed presterende Bayesiaanse netwerk classifiers.

In het tweede deel van dit proefschrift integreren we bepaalde techieken van evolutionaire algoritmen in een "Markov chain Monte Carlo (MCMC) sampler". "Sampling" is gerelateerd aan optimalisatie, maar optimalisatie is geinteresseerd in het vinden van de toestand met de hoogste fitness waarde en sampling is geinteresseerd in de hele kansverdeling over de toestand ruimte. Sampling is nuttig voor het berekenen van bepaalde functies over een toestand ruimte waarin verschillende toestanden verschillende kansen hebben. Een voorbeeld functie is het berekenen van het centrum van de kansverdeling hetgeen niet analytisch gedaan kan worden voor complexe kansverdelingen. Optimalisatie kan ook gezien worden als het gebruiken van een functie die enkel het maximum selecteert in de toestand ruimte, maar in tegenstelling tot bepaalde complexe sampling functies, vereisen optimalisatie algoritmen enkel dat het maximale element wordt opgeslagen. Het doel van sampling is om veel samples te genereren volgens een bepaalde (niet-genormalizeerde) kansverdeling en deze samples te gebruiken voor verdere verwerking. In dit proefschrift zijn we vooral geinteresseerd in efficiente sampling technieken en minder in de verwerking van de gegenereerde samples.

Voor het verbeteren van MCMC methoden welke gewoonlijk gebruikt worden voor sampling, onderzoeken we het evolutionaire MCMC (EMCMC) framewerk, waarin een populatie van individuen informatie kunnen uitwisselen. Voor het convergeren naar de gewenste kansverdeling, moeten we ervoor zorgen dat de EMCMC nog steeds een MCMC sampler is op populatie nivo. Als beginpunt, maken we gebruik van het Metropolis-Hastings algoritme welke bepaalde kandidaat individuen voorstellen met een kansverdeling die niet noodzakelijk gerelateerd is aan de gewenste kansverdeling. De voorgestelde individuen worden dan geaccepteerd of verworpen door een acceptatie regel zodat het algoritme naar de gewenste kansverdeling convergeert.

In het ideale geval zou een MCMC algoritme individuen voorstellen door direct gebruik te maken van de gewenste kansverdeling. Als dat mogelijk zou zijn dan zou het algoritme heel snel convergeren naar de gewenste kansverdeling omdat alle toestanden volgens hun gewenste kans voorgesteld zouden worden. Helaas is het in de praktijk vrijwel altijd onmogelijk om direct volgens de gewenste kansverdeling individuen voor te stellen. Een standaard MCMC algoritme stelt een kandidaat individu meestal voor met een bepaalde kansverdeling, zoals de normale verdeling, gecentreerd rond het huidige individu welke meestal de ouder genoemd wordt. Een dergelijke kansverdeling heeft echter geen kennis van de gewenste kansverdeling en daarom worden er doorgaans veel kandidaat individuen voorgesteld die een lage kans hebben en vervolgens verworpen worden door de acceptatie regel. Wij bestuderen 'slimmere' methoden die zich aanpassen aan de gewenste kansverdeling door gebruik te maken van recombinatie operatoren. Vanwege het sampling proces zal de populatie vaker individuen hebben met een hoge kans volgens de gewenste kansverdeling. Daarom kunnen de recombinatie operatoren structurele informatie die aanwezig is in de ouder individuen uitbuiten om nieuwe individuen te genereren die ook een hoge kans hebben. Als de exploratie bias van de recombinatie operatoren enigzins overeenkomt met de structurele relaties in de gewenste kansverdeling, dan worden de nieuwe kandidaat individuen ook vaker geaccepteerd door de acceptatie regel waardoor het ECMCM algoritme veel efficienter is dan een standaard MCMC. Voor dit doel, beschrijven we een aantal verschillende recombinatie operatoren voor discrete en continue toestand ruimtes en analyseren we hun eigenschappen.

We analyseren vervolgens ook de interactie tussen de recombinatie operatoren en de acceptatie

regels. In het algemeen geldt dat convergentie naar de gewenste kansverdeling plaats vindt als alle kandidaat individuen die voorgesteld zijn door middel van een recombinatie operator allemaal geaccepteerd of allemaal verworpen worden. Een dergelijke acceptatie regel noemen we een gekoppelde acceptatie regel, omdat alle kandidaat individuen gezamelijk worden verworpen of geaccepteerd. Dit beperkt het accepteren van nieuwe individuen omdat het makkelijk kan gebeuren dat bepaalde kandidaat individuen een hoge kans en andere een lage kans hebben. Het probleem van de gekoppelde acceptatie regel is dat het niet enkel de goede kandidaat individuen kan accepteren en de slechtere kan verwerpen. In hoofdstuk 7 beschrijven we een aantal experimenten waarmee duidelijk wordt aangetoond dat deze gekoppelde acceptatie regel de prestatie van een EMCMC algoritme kan verslechteren. Om dit probleem op te heffen, bestuderen we het gebruik van enkele ouder versus kind acceptatie regels waarin een kandidaat die voorgesteld is met een recombinatie operator het tegen een ouder opneemt, een ander voorgestelde kandidaat tegen een andere ouder, enzovoorts. We bewijzen dat onder bepaalde condities het resulterende algoritme nog steeds convergeert naar de gewenste kansverdeling zonder dat het de problemen van de gekoppelde acceptatie regel heeft. De experimentele resultaten tonen aan dat als we de gekoppelde met de niet gekoppelde acceptatie regels vergelijken, de niet gekoppelde acceptatie regel - in combinatie met verschillende EMCMC methoden - veel efficienter samples genereert volgens de gewenste kansverdeling.

Acknowledgments

This work was carried out under supervision of prof. dr. ir. Linda van der Gaag and dr. ir. Dirk Thierens. Thanks to Dirk Thierens, my daily supervisor, for guiding me through the Evolutionary Computation field. I really enjoyed doing research in this field. Thanks to Linda van der Gaag for the useful comments on the first part of this thesis. Thanks to the members of the reading committee: Thomas Bäck, John-Jules Meyer, Jose Peña, Han la Poutré and Arno Siebes.

I also had the opportunity to interact with some people from the Decision Support System group for broading my research interests: Petra, Steven, Edwin, Silja, Janneke, Eveline, Michael, Theodoros and Peter. Thanks to Mirela, Karina, Jaesook, Ion and Ovidiu for non-scientifically but pleasant conversations.

Marco Wiering, my dearest husband, thank you for reading my research from time to time and for your support and love. Thanks to Nettie and DJ, Annemarie and Réne, Elena, Maxim and Johan for the nice family reunions in Holland. I am grateful to my parents for helping me to raise my child during my PhD period. I am thankful to my brother Ovidiu and his family, Andreea and Cristina, and my grandmother for encouraging me. Finally, I want to mention that apart for being happy that I have finished and I am delighted that Marius, my son, is a very lovely boy.

CURRICULUM VITAE

Higher education:

- 1999 - 2000: Postmaster in Informational Systems, Computer Science Department, Technical University of Cluj-Napoca, Romania. I have graduated with the final project *SharedPlans for a Multi-Agent system* into the Multi Agents domain.
- 1994 - 1999: Master degree in Computer Science, Computer Science Department, Technical University of Cluj-Napoca, Romania. I have graduated with the final project *An Implementation of Iterated Prisoners Dilemma with Tierra* into Artificial Life and game theory domains.

Work experience:

- 2006-on: postdoc in the Bioinformatics group at Biomolecular Mass- Spectrometry, Department of Pharmaceutical Sciences, Utrecht University.
- 2001 - 2006: PhD candidate in the Decision Support Systems at Computer Science Department, Utrecht University.
- 2000 - 2001: assistant researcher at Software Technology, Stan Ackermans Institute, Technical University of Eindhoven. I have completed courses and software projects individually and in teams. In teams, we have won the award of the best Software Architectures.
- 1999-2000: assisted teaching for the courses *Fundamental Elements of Graphics, Numeric Computers, Programming Engineering* at Technical University of Cluj-Napoca. I have prepared practical work and I have assisted examinations.

Social status:

- 1 child
- married

List of publications:

Conditional log-likelihood MDL

- Drugan, M.M., and van der Gaag, L.C., *A New MDL-based function for feature selection for Bayesian network classifiers.*, Proceedings of ECAI'04, pp. 999-1000, 2004
- Drugan, M.M., and Thierens, D., and van der Gaag, L.C., *MDL-based feature selection for Bayesian network classifiers.*, Proceedings of BNAIC, pp. 99-106, 2002.

- Drugan, M.M., and van der Gaag, L.C., *Feature selection for Bayesian Network Classifiers using the MDL-FS score*., to be submitted, 60 pages.

Evolutionary MCMC

- Drugan, M.M., and Thierens, D., *Recombinative EMCMC algorithms*., Proceedings of CEC 2005, pp. 2024-2031, 2005.
- Drugan, M.M., and Thierens, D., *Evolutionary Markov chain Monte Carlo*., In P. Liardet, P. Collet, C. Fonlupt, E. Lutton and M. Schoenauer (Eds.), LNCS 2936, Springer, pp. 63-76, 2004.
- Drugan, M.M., and Thierens, D., *Geometrical Recombination Operators for Real-Coded Evolutionary MCMCs*., submitted at Evolutionary Computation, 31 pages.
- Drugan, M.M., and Thierens, D., *Evolutionary MCMCs for Sampling and Optimization*., submitted at IEEE Transactions on Evolutionary Computation, 37 pages.

Miscellaneous

- Jong, E.D. de, and Wiering, M.A., and Drugan, M.M., *Post-Processing for MCMC*., UU-CS (Ext. r. no. 2003-021). Utrecht: Utrecht University: Information and Computing Sciences, 2003.
- Cenan, C., and Drugan, M.M., *An Implementation of Iterated Prisoners Dilemma with Tierra*., Proceedings of The International Scientific Symposium SINTES 10., 2000.

SIKS Dissertatiereeks

==== 1998 ====

- 1998-1 Johan van den Akker (CWI) DEGAS - An Active, Temporal Database of Autonomous Objects
- 1998-2 Floris Wiesman (UM) Information Retrieval by Graphically Browsing Meta-Information
- 1998-3 Ans Steuten (TUD) A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
- 1998-4 Dennis Breuker (UM) Memory versus Search in Games
- 1998-5 E.W.Oskamp (RUL) Computerondersteuning bij Straftoemeting

==== 1999 ====

- 1999-1 Mark Sloof (VU) Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products
- 1999-2 Rob Potharst (EUR) Classification using decision trees and neural nets
- 1999-3 Don Beal (UM) The Nature of Minimax Search
- 1999-4 Jacques Penders (UM) The practical Art of Moving Physical Objects
- 1999-5 Aldo de Moor (KUB) Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
- 1999-6 Niek J.E. Wijngaards (VU) Re-design of compositional systems
- 1999-7 David Spelt (UT) Verification support for object database design
- 1999-8 Jacques H.J. Lenting (UM) Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

==== 2000 ====

- 2000-1 Frank Niessink (VU) Perspectives on Improving Software Maintenance
- 2000-2 Koen Holtman (TUE) Prototyping of CMS Storage Management
- 2000-3 Carolien M.T. Metselaar (UVA) Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.
- 2000-4 Geert de Haan (VU) ETAG, A Formal Model of Competence Knowledge for User Interface Design
- 2000-5 Ruud van der Pol (UM) Knowledge-based Query Formulation in Information Retrieval.
- 2000-6 Rogier van Eijk (UU) Programming Languages for Agent Communication
- 2000-7 Niels Peek (UU) Decision-theoretic Planning of Clinical Patient Management
- 2000-8 Veerle Coup, (EUR) Sensitivity Analyis of Decision-Theoretic Networks
- 2000-9 Florian Waas (CWI) Principles of Probabilistic Query Optimization
- 2000-10 Niels Nes (CWI) Image Database Management System Design Considerations, Algorithms and Architecture
- 2000-11 Jonas Karlsson (CWI) Scalable Distributed Data Structures for Database Management

==== 2001 ====

- 2001-1 Silja Renooij (UU) Qualitative Approaches to Quantifying Probabilistic Networks
- 2001-2 Koen Hindriks (UU) Agent Programming Languages: Programming with Mental Models

- 2001-3 Maarten van Someren (UvA) Learning as problem solving
- 2001-4 Evgueni Smirnov (UM) Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
- 2001-5 Jacco van Ossenbruggen (VU) Processing Structured Hypermedia: A Matter of Style
- 2001-6 Martijn van Welie (VU) Task-based User Interface Design
- 2001-7 Bastiaan Schonhage (VU) Diva: Architectural Perspectives on Information Visualization
- 2001-8 Pascal van Eck (VU) A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
- 2001-9 Pieter Jan 't Hoen (RUL) Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
- 2001-10 Maarten Sierhuis (UvA) Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design
- 2001-11 Tom M. van Engers (VUA) Knowledge Management: The Role of Mental Models in Business Systems Design

==== 2002 ====

- 2002-01 Nico Lassing (VU) Architecture-Level Modifiability Analysis
- 2002-02 Roelof van Zwol (UT) Modelling and searching web-based document collections
- 2002-03 Henk Ernst Blok (UT) Database Optimization Aspects for Information Retrieval
- 2002-04 Juan Roberto Castelo Valdueza (UU) The Discrete Acyclic Digraph Markov Model in Data Mining
- 2002-05 Radu Serban (VU) The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents
- 2002-06 Laurens Mommers (UL) Applied legal epistemology; Building a knowledge-based ontology of the legal domain
- 2002-07 Peter Boncz (CWI) Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
- 2002-08 Jaap Gordijn (VU) Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
- 2002-09 Willem-Jan van den Heuvel(KUB) Integrating Modern Business Applications with Objectified Legacy Systems
- 2002-10 Brian Sheppard (UM) Towards Perfect Play of Scrabble
- 2002-11 Wouter C.A. Wijngaards (VU) Agent Based Modelling of Dynamics: Biological and Organisational Applications
- 2002-12 Albrecht Schmidt (Uva) Processing XML in Database Systems
- 2002-13 Hongjing Wu (TUE) A Reference Architecture for Adaptive Hypermedia Applications
- 2002-14 Wieke de Vries (UU) Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems
- 2002-15 Rik Eshuis (UT) Semantics and Verification of UML Activity Diagrams for Workflow Modelling
- 2002-16 Pieter van Langen (VU) The Anatomy of Design: Foundations, Models and Applications
- 2002-17 Stefan Manegold (UVA) Understanding, Modeling, and Improving Main-Memory Database Performance

==== 2003 ====

- 2003-01 Heiner Stuckenschmidt (VU) Ontology-Based Information Sharing in Weakly Structured Environments
- 2003-02 Jan Broersen (VU) Modal Action Logics for Reasoning About Reactive Systems
- 2003-03 Martijn Schuemie (TUD) Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
- 2003-04 Milan Petkovic (UT) Content-Based Video Retrieval Supported by Database Technology
- 2003-05 Jos Lehmann (UVA) Causation in Artificial Intelligence and Law - A modelling approach
- 2003-06 Boris van Schooten (UT) Development and specification of virtual environments
- 2003-07 Machiel Jansen (UvA) Formal Explorations of Knowledge Intensive Tasks
- 2003-08 Yongping Ran (UM) Repair Based Scheduling
- 2003-09 Rens Kortmann (UM) The resolution of visually guided behaviour

- 2003-10 Andreas Lincke (UvT) Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
- 2003-11 Simon Keizer (UT) Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
- 2003-12 Roeland Ordelman (UT) Dutch speech recognition in multimedia information retrieval
- 2003-13 Jeroen Donkers (UM) Nosce Hostem - Searching with Opponent Models
- 2003-14 Stijn Hoppenbrouwers (KUN) Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
- 2003-15 Mathijs de Weerdt (TUD) Plan Merging in Multi-Agent Systems
- 2003-16 Menzo Windhouwer (CWI) Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
- 2003-17 David Jansen (UT) Extensions of Statecharts with Probability, Time, and Stochastic Timing
- 2003-18 Levente Kocsis (UM) Learning Search Decisions

==== 2004 ====

- 2004-01 Virginia Dignum (UU) A Model for Organizational Interaction: Based on Agents, Founded in Logic
- 2004-02 Lai Xu (UvT) Monitoring Multi-party Contracts for E-business
- 2004-03 Perry Groot (VU) A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
- 2004-04 Chris van Aart (UVA) Organizational Principles for Multi-Agent Architectures
- 2004-05 Viara Popova (EUR) Knowledge discovery and monotonicity
- 2004-06 Bart-Jan Hommes (TUD) The Evaluation of Business Process Modeling Techniques
- 2004-07 Elise Boltjes (UM) Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes
- 2004-08 Joop Verbeek(UM) Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politile gegevensuitwisseling en digitale expertise
- 2004-09 Martin Caminada (VU) For the Sake of the Argument; explorations into argument-based reasoning
- 2004-10 Suzanne Kabel (UVA) Knowledge-rich indexing of learning-objects
- 2004-11 Michel Klein (VU) Change Management for Distributed Ontologies
- 2004-12 The Duy Bui (UT) Creating emotions and facial expressions for embodied agents
- 2004-13 Wojciech Jamroga (UT) Using Multiple Models of Reality: On Agents who Know how to Play
- 2004-14 Paul Harrenstein (UU) Logic in Conflict. Logical Explorations in Strategic Equilibrium
- 2004-15 Arno Knobbe (UU) Multi-Relational Data Mining
- 2004-16 Federico Divina (VU) Hybrid Genetic Relational Search for Inductive Learning
- 2004-17 Mark Winands (UM) Informed Search in Complex Games
- 2004-18 Vania Bessa Machado (UvA) Supporting the Construction of Qualitative Knowledge Models
- 2004-19 Thijs Westerveld (UT) Using generative probabilistic models for multimedia retrieval
- 2004-20 Madelon Evers (Nyenrode) Learning from Design: facilitating multidisciplinary design teams

==== 2005 ====

- 2005-01 Floor Verdenius (UVA) Methodological Aspects of Designing Induction-Based Applications
- 2005-02 Erik van der Werf (UM)) AI techniques for the game of Go
- 2005-03 Franc Grootjen (RUN) A Pragmatic Approach to the Conceptualisation of Language
- 2005-04 Nirvana Meratnia (UT) Towards Database Support for Moving Object data
- 2005-05 Gabriel Infante-Lopez (UVA) Two-Level Probabilistic Grammars for Natural Language Parsing
- 2005-06 Pieter Spronck (UM) Adaptive Game AI
- 2005-07 Flavius Frasincar (TUE) Hypermedia Presentation Generation for Semantic Web Information Systems
- 2005-08 Richard Vdovjak (TUE) A Model-driven Approach for Building Distributed Ontology-based Web Applications

- 2005-09 Jeen Broekstra (VU) Storage, Querying and Inferencing for Semantic Web Languages
- 2005-10 Anders Bouwer (UVA) Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
- 2005-11 Elth Ogston (VU) Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
- 2005-12 Csaba Boer (EUR) Distributed Simulation in Industry
- 2005-13 Fred Hamburg (UL) Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
- 2005-14 Borys Omelayenko (VU) Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
- 2005-15 Tibor Bosse (VU) Analysis of the Dynamics of Cognitive Processes
- 2005-16 Joris Graaumans (UU) Usability of XML Query Languages
- 2005-17 Boris Shishkov (TUD) Software Specification Based on Re-usable Business Components
- 2005-18 Danielle Sent (UU) Test-selection strategies for probabilistic networks
- 2005-19 Michel van Dartel (UM) Situated Representation
- 2005-20 Cristina Coteanu (UL) Cyber Consumer Law, State of the Art and Perspectives
- 2005-21 Wijnand Derks (UT) Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

==== 2006 ====

- 2006-01 Samuil Angelov (TUE) Foundations of B2B Electronic Contracting
- 2006-02 Cristina Chisalita (VU) Contextual issues in the design and use of information technology in organizations
- 2006-03 Noor Christoph (UVA) The role of metacognitive skills in learning to solve problems
- 2006-04 Marta Sabou (VU) Building Web Service Ontologies
- 2006-05 Cees Pierik (UU) Validation Techniques for Object-Oriented Proof Outlines
- 2006-06 Ziv Baida (VU) Software-aided Service Bundling - Intelligent Methods Tools for Graphical Service Modeling
- 2006-07 Marko Smiljanic (UT) XML schema matching – balancing efficiency and effectiveness by means of clustering
- 2006-08 Eelco Herder (UT) Forward, Back and Home Again - Analyzing User Behavior on the Web
- 2006-09 Mohamed Wahdan (UM) Automatic Formulation of the Auditor's Opinion
- 2006-10 Ronny Siebes (VU) Semantic Routing in Peer-to-Peer Systems
- 2006-11 Joeri van Ruth (UT) Flattening Queries over Nested Data Types
- 2006-12 Bert Bongers (VU) Interactivation - Towards an e-cology of people, our technological environment, and the arts
- 2006-13 Henk-Jan Lebbink (UU) Dialogue and Decision Games for Information Exchanging Agents
- 2006-14 Johan Hoorn (VU) Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
- 2006-15 Rainer Malik (UU) CONAN: Text Mining in the Biomedical Domain
- 2006-16 Carsten Riggelsen (UU) Approximation Methods for Efficient Learning of Bayesian Networks
- 2006-17 Stacey Nagata (UU) User Assistance for Multitasking with Interruptions on a Mobile Device
- 2006-18 Valentin Zhizhkun (UVA) Graph transformation for Natural Language Processing
- 2006-19 Birna van Riemsdijk (UU) Cognitive Agent Programming: A Semantic Approach
- 2006-20 Marina Velikova (UvT) Monotone models for prediction in data mining
- 2006-21 Bas van Gils (RUN) Aptness on the Web
- 2006-22 Paul de Vrieze (RUN) Fundaments of Adaptive Personalisation
- 2006-23 Ion Juvina (UU) Development of Cognitive Model for Navigating on the Web
- 2006-24 Laura Hollink (VU) Semantic Annotation for Retrieval of Visual Resources
- 2006-25 Madalina Drugan (UU) Conditional log-likelihood MDL and Evolutionary MCMC