Bayesian Network Structure Learning Using Characteristic Properties of Permutation Representations with Applications to Prostate Cancer Treatment.

Olivier Regnier-Coudert

A thesis submitted in partial fulfilment of the requirements for the degree

of Doctor of Philosophy

at the School of Computing

Robert Gordon University

Aberdeen, Scotland

May 2013

Principal supervisors

Prof. John McCall. IDEAS Research Institute, Robert Gordon University, Aberdeen, Scotland

Mr. Sam McClinton. Academic Urology Unit, University Of Aberdeen, Aberdeen, Scotland

Secondary supervisors

Mr. Thomas Lam. Academic Urology Unit, University Of Aberdeen, Aberdeen, Scotland

Dr. Robert Lothian. IDEAS Research Institute, Robert Gordon University, Aberdeen, Scotland

Prof. James N'Dow. Academic Urology Unit, University Of Aberdeen, Aberdeen, Scotland

Abstract

Over the last decades, Bayesian Networks (BNs) have become an increasingly popular technique to model data under presence of uncertainty. BNs are probabilistic models that represent relationships between variables by means of a node structure and a set of parameters. Learning efficiently the structure that models a particular dataset is a NP-hard task that requires substantial computational efforts to be successful. Although there exist many families of techniques for this purpose, this thesis focuses on the study and improvement of search and score methods such as Evolutionary Algorithms (EAs).

In the domain of BN structure learning, previous work has investigated the use of permutations to represent variable orderings within EAs. In this thesis, the characteristic properties of permutation representations are analysed and used in order to enhance BN structure learning. The thesis assesses well-established algorithms to provide a detailed analysis of the difficulty of learning BN structures using permutation representations. Using selected benchmarks, rugged and plateaued fitness landscapes are identified that result in a loss of population diversity throughout the search.

The thesis proposes two approaches to handle the loss of diversity. First, the benefits of introducing the Island Model (IM) paradigm are studied, showing that diversity loss can be significantly reduced. Second, a novel agent-based metaheuristic is presented in which evolution is based on the use of several mutation operators and the definition of a distance metric in permutation spaces. The latter approach shows that diversity can be maintained throughout the search while exploring efficiently the solution space. In

addition, the use of IM is investigated in the context of distributed data, a common property of real-world problems. Experiments prove that privacy can be preserved while learning BNs of high quality.

Finally, using UK-wide data related to prostate cancer patients, the thesis assesses the general suitability of BNs alongside the proposed learning approaches for medical data modeling. Following comparisons with tools currently used in clinical settings and with alternative classifiers, it is shown that BNs can improve the predictive power of prostate cancer staging tools, a major concern in the field of urology.

Acknowledgements

It took me four years to complete my PhD. Although there were some difficult times and a fair amount of moaning and groaning, the overall experience was a positive one. I believe it is mainly related to the stimulating and friendly environment I had the chance to be part of during that period of time. Thus, I would like to take the opportunity to thank the many people that made it be such a rewarding episode of my life, both from a professional and personal point of view.

First of all, I would like to thank my principal supervisor, John McCall for his help before and during my PhD in introducing me to the research world. I very much appreciated his methods of supervision, leaving me the freedom to decide which directions to give to my research. John also involved me in many other research projects and offered me the opportunity to travel to conferences, which further widened my interest in the field. I of course do not forget the many discussions and digressions that helped me during this process.

I would also like to thank the rest of my supervisory team and particularly Sam McClinton and Thomas Lam for their help in deciphering the medical knowledge. Our discussions gave to the cross-disciplinary nature of my work a nice savor. I also would like to thank the British Association of Urological Surgeons for providing me with the prostate cancer patient data that played a very important role in my thesis and my funding bodies, NHS and the Northern Research Partnership.

Most of my time in these past years was spent in lovely CTC where I met many fellow

research students that made it a convivial workplace. I would like to thank particularly Yanghui and Francois for all the discussions and code sharing we had and Michael for his patience in assisting me in using and crashing some of the parallel computing facilities. I would also like to thank the School of Computing IT team alongside Virginia and Duncan for their constant help throughout my PhD.

I also take this opportunity to thank my viva committee composed of Qingfu Zhang and Andrei Petrovski for their time and their valuable comments and suggestions.

Of course, my life outside work is also filled with fun and many exciting moments and I cease the opportunity to think about all the friends I have, met on the road between Courpiere and Aberdeen and now scattered across the world. I am glad to have them.

To finish this section, I would like to briefly switch languages to thank more personally a few people.

Nil, bu satırlar senin için. Sana sen olduğun için teşekkür etmek istiyorum, her gün yüzünde gördüğüm sevgi ve tebessüm için, ve benimle hazine avı, bahçıvanlık, aşçılık ve zanaatkarlık gibi türlü maceralara atıldığn için. Hiç bir zaman bir astronot olmayacak olsan da, bunların hepsi hayatımı heyecanla dolduruyor ve beni mutlu bir adam yapıyor.

Pour terminer, j'aimerais remercier mes parents Jocelyne et Roby pour tout ce qu'ils m'ont apporté durant ces 28 dernières années. Merci tout d'abord pour votre soutien financier et moral essentiel durant ces longues études mais aussi dans ma vie en géneral. Merci aussi pour tous ces bons moments, ces paris en pagailles, ces débats sur les chaussettes, cette valise trop chère, ce pique-nique aux crevettes, ce petit lapin ou encore ce chausson qui aimait tant les fleurs. Tout cela, j'en suis sur, m'a beaucoup servi. Enfin, merci d'avoir accepté de me laisser partir si loin pour mes études et d'accepter mes choix dans ma vie. Cette thèse est pour vous.

Published Papers

- Lam, T., Regnier-Coudert, O., McCall, J., Lothian, R. and McClinton, S. (2010), Prostate cancer staging nomograms: Validation on a British population, BJUI , Vol. 106, pp. 2-3.
- Regnier-Coudert, O. and McCall, J. (2011), Privacy-preserving approach to bayesian network structure learning from distributed data, in Proceedings of the 13th annual conference companion on Genetic and evolutionary computation, ACM, pp. 815-816.
- Regnier-Coudert, O., McCall, J., Lothian, R., Lam, T., McClinton, S. and NDow, J. (2011), Machine learning for improved pathological staging of prostate cancer: A performance comparison on a range of classifiers, Artificial Intelligence in Medicine, Vol. 55, Elsevier, pp. 25-35.
- Lam, T., Regnier-Coudert, O., McCall, J. and McClinton, S. (2012), Development and validation of a uk-specific prostate cancer staging predictive model: Uk prostate cancer tables, British Journal of Medical and Surgical Urology, Vol.5, Elsevier, pp. 224-235.
- Regnier-Coudert, O. and McCall, J. (2012), An island model genetic algorithm for bayesian network structure learning, in Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC), IEEE, pp. 1-8.
- Regnier-Coudert, O. and McCall, J. (2012), Competing mutating agents for bayesian network structure learning, Proceedings of Parallel Problem Solving from Nature XII, Springer, pp. 216-225.

Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

ACO Ant Colony Optimization

AIC Akaike Information Criterion

ANN Artificial Neural Network

ARI Aberdeen Royal Infirmary

AUC Area Under the ROC Curve

BAUS British Association of Urological Surgeons

BD Bayesian Dirichlet

BDe likelihood-equivalence Bayesian Dirichlet

BDeu uniform BDe

BIC Bayesian Information Criterion

BN Bayesian Network

BOA Bayesian Optimization Algorithm

c-index Concordance Index

CH Cooper Herskovits score

CI Conditional Independence

CMI Conditional Mutual Information

COMMA COMpeting Mutating Agents

COMMA$_\mathbf{d}$ COMpeting Mutating Agents with degradation

CPT Conditional Probability Table

CS Clinical Stage

DAG Directed Acyclic Graph

DRE Digital Rectal Examination

EA Evolutionary Algorithm

EBNA Estimation of BN Algorithm

EDA Estimation of Distribution Algorithm

EP Evolutionary Programing

EPE Extra-Prostatic Extension

FDC Fitness Distance Correlation

GA Genetic Algorithm

GS Gleason Sum score

IM-COMMA Island Model COMpeting Mutating Agents

IM-COMMA$_\mathbf{d}$ Island Model COMpeting Mutating Agents with degradation

IMK2GA Island Model K2 Genetic Algorithm

IPSS International Prostate Symptom Score

k-NN k-Nearest Neighbours

K2GA K2 Genetic Algorithm

KTD Kendall-Tau Distance

LNI Lymph Node Involvement

LR Logistic Regression

MDL Minimum Description Length

MI Mutual Information

MIMIC Mutual Information Maximization for Input Clustering

MLE Maximum Likelihood Estimation

MLP MultiLayer Perceptron

MPE Most Probable Explanation

MRI Magnetic Resonance Imaging

NB Naive Bayes

OC Organ Confined

PGM Probabilistic Graphical Model

PLS Probabilistic Logic Sampling

PS Pathological Stage

PSA Prostate Specific Antigen

QoL Quality of Life

RBF Radial Basis Function

RF Random Forest

SHD Structural Hamming Distance

SVI Seminal Vesicle Involvement

SVM Support Vector Machine

TAN Tree Augmented Naive Bayes

TRUS TRansrectal UltraSound

UMDA Univariate Marginal Distribution Algorithm

Chapter 1

Introduction

1.1 Background

Bayesian Network (BN) (Pearl, 1988) is a powerful data modelling tool to reason under uncertainty and has been used in many applications. Learning BNs is not straightforward as the number of possible network structures that can be built from a set of variables becomes very large with the increase of the number of features. There exist several approaches to learning BN structures, from using conditional independence tests to applying metaheuristics. Metaheuristic techniques can also be distinguished from each other. While some focus on searching directly for a BN structure, others aim at finding the most representative ordering of variables from which structures can be derived using deterministic methods. This latter family of algorithms is treated in this thesis, following work that has been conducted in implementing *Evolutionary Algorithms (EAs)* (Back et al., 1997) to search for an optimal ordering of variables that can be used in conjunction with a greedy search to construct BN structures (Larrañaga, Kuijpers, Murga and Yurramendi, 1996). In such problems, variable orderings are represented as *permutations*. In a nutshell, EAs aim at improving the quality of potential solutions of a given problem by evolving them according to some strategies that differ between types of EAs. For example, the well known *Genetic Algorithm (GA)* (Holland,

1975) builds a population of solutions and score each of them according to some fitness function. A selection process is performed in which some promising solutions are chosen and recombined in order to create new solutions. Since, new solutions are generated from selected solutions, they present higher chances to be good as well. As this process is repeated, the quality of the population is improved and so are the chances to find the best solution to the problem.

In recent years, efforts were made in defining novel ways to search the space of variable orderings efficiently. Consequently, several population-based search and score algorithms were implemented (Larrañaga, Kuijpers, Murga and Yurramendi, 1996; Hsu et al., 2002; Kabli et al., 2007; Wu et al., 2010). Despite showing interesting results, some of them suffer from early convergence related to the rapid drop in diversity within their populations that can be caused by the selection and recombination processes. In this thesis, we touch upon this issue and present ways around it that lead to several themes of study. These include:

- a detailed analysis of the factors that can affect population diversity in permutation representations;
- the investigation of the performance characteristics of existing approaches within an Island Model framework designed to promote diversity;
- the investigation of the performance characteristics of a novel search and score approach that focuses on maintaining a steady level of diversity throughout the search.

These themes are also put in the context of medical data modelling where the use of machine learning and of BNs more specifically have been identified as specially appropriate to model medical reasoning with the uncertainty it implies (Oniśko, 2008). Particular applications related to prostate cancer treatment are studied in the present work. Three aspects are investigated, the specific problem of prostate cancer staging, general data modelling by means of BNs and the challenge of privacy-preserving data

modelling from distributed data.

The themes and applications covered in this thesis are relevant to several research domains such as BN structure learning, permutation representations or parallel metaheuristics. Although the focus is mainly set on the aforementioned, presented methods and findings may also benefit research in other domains where BNs can be applied. These include for example the fields of general urology (Kabli et al., 2008), cardiology (Wiggins et al., 2008), neurosciences (Oteniya, 2008), oil and gas engineering (Fournier et al., 2010), meteorology (Cano et al., 2004), risk management or sensor validation (Pourret et al., 2008).

1.2 Research Questions

As already mentioned, this thesis addresses several issues. However, the research objective raised by all themes can be defined in three questions:

- What is the influence of diversity in BN structure learning population-based search and score approaches?
- Can information on diversity be used in order to enhance the search?
- How do BNs and diversity analysis benefit data modelling with respect to the semantic meaning of the data and in particular for data relevant to prostate cancer?

1.3 Summary of Thesis

Below is a brief description of the content of each chapter.

Chapter 2. Background. This chapter presents general background information on techniques that are dealt with in this thesis. It describes Bayesian Networks (BNs) and different approaches to BN structure learning, including EAs and considers the

key studies from literature. Finally, tools used to assess BN structure quality are presented.

Chapter 3. Difficulty of Node Ordering Permutation Optimization. This chapter highlights the current limitations of search and score algorithms used for BN structure learning that are based on permutation representations. These observations are supported by analyzing the fitness landscapes associated with permutation search spaces, with a particular focus on the node ordering representation. As a result, key aspects that need improvement are exposed, setting the motivation for the following chapters.

Chapter 4. Combining Local Optima by Means of Island Models. This chapter presents an approach based on the use of island models to tackle the problem of early convergence of a specific GA implementation, K2GA, on BN structure learning problems in presence of local optima.

Chapter 5. Size of Neighborhood and Population Diversity. In this chapter, a novel method is described that focuses on maintaining high diversity in a population of agents. Different roles are assigned to each agent according to its rank in the population by means of distance-based mutation. This ensures constant exploration of the search space and exploitation of the best areas.

Chapter 6. Applications to Prostate Cancer Staging. Techniques implemented in the previous chapters are applied on British prostate cancer patient data. Benefits of the use of BNs for prostate cancer staging are presented following comparison with alternative techniques currently used in a clinical setting.

Chapter 7. Discussion and Directions for Further Research. This chapter presents the conclusions of the thesis and describes potential areas of research that can rise from this work.

Chapter 2

Background

2.1 Bayesian Networks

Bayesian Network (BN) is a type of Probabilistic Graphical Model (PGM) developed in (Pearl, 1988). BNs were inspired by the way humans process information and organize their knowledge. Typically, when facing a problem, humans are able to divide the information available into smaller sub-problems and to handle uncertain information based on their knowledge. A famous case is to find out what the cause of a car being unable to start is. A car is composed of several parts and a breakdown can possibly be caused by many of them. When facing this problem, the human brain splits the bigger problem into smaller ones and starts solving them individually. One may propose different possible areas of investigation such as the lack of fuel in the tank, the state of the battery or the quality of the sparks for instance. If one of these shows signs of potential failure, the other possible causes are likely to be disregarded. Behind this decision lies a probabilistic model that represents our knowledge built for example from experience or from literature. Since one may think that having two simultaneous distinct failures is very unlikely to cause the car breakdown, efforts will be made to fix the first identified problem while assuming everything else is operational.

Similarly, in a BN, the problem knowledge is also factorized. To do so, conditional relationships between problem variables X_i are identified and represented by means of *nodes* and directed *edges*. In a BN, the set of all edges and all nodes is a graph and is called the *structure* of the BN. To be a valid BN, its final structure requires to be a Directed Acyclic Graph (DAG). In addition, each node in a BN is given a set of *parameters* in the form of a *Conditional Probability Table (CPT)* which is conditioned on the state of its parents $Pa(X_i)$ in the graph as shown in Figure 2.1. In a BN, the set of all CPTs is often refered to as its parameters.Each random variable X_i can be instantiated with a value x_i whose distribution is determined from the BN parameters. One of the main benefits of BN lies in the fact that it factorizes the *joint probability distribution* P of a set of n variables X_i according to their respective parents $Pa(X_i)$ as shown in (2.1).

$$P = P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{2.1}$$

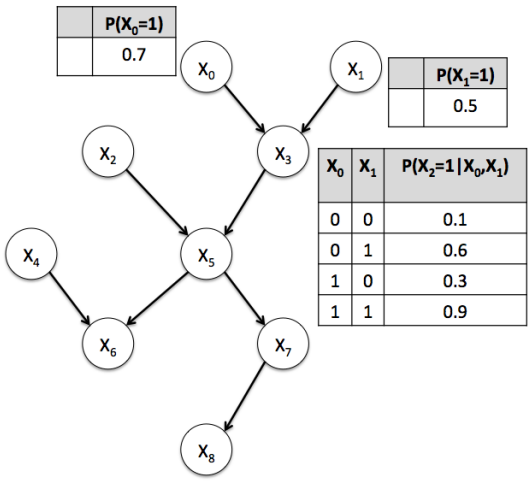


Figure 2.1: Example of Bayesian network with CPTs shown for 3 variables X_0, X_1 and X_3

One of the properties that can be studied on a BN is its topology, or in other terms its *depth* and *width*. Such characteristics are imporant to consider because some topologies

tend to affect structure learning methods. In order to express the definition of the width and depth of a BN, the notion of *level* should be introduced. A level represents the set of all nodes whose parents have already been assigned to a previous level. The set of nodes with no parent constitutes the level 0. Based on this definition, the depth of a BN is defined by the number of levels it exhibits. On the other hand, the width of a BN is defined by the number of nodes that compose its largest level.

The two main tasks in learning BNs are the structure and parameter learning steps. Both steps can either be done by gathering expert knowledge or by mining data related to the problem. *Evidence* can then be injected in the BN, such as for example the observation that the battery of the car is empty. The variable in the BN related to the state of the battery is instantiated with the value corresponding to its empty state. Once set, evidence propagates to the rest of the BN according to the conditional relationships previously defined. Evidence propagation affects other variables by changing their state probabilities. An illustration is given in Figure 2.2 for the simplified car start problem previously described. In Figure 2.2a, a BN with conditional relationships and prior knowledge is represented. It shows that $CarStarts$ depends on the state of the three variables $Sparks$, $Battery$ and $Fuel$. Each of these three variables has a prior probability, that is each of these components has a 90% likelihood to be operational (OK state). On the other hand, the CPT of $CarStarts$ is conditioned on the three other variables. Hence, without any observation on the car state, a 71 % chance is given for the car to start. In Figure 2.2b, evidence has been set on the $CarStarts$ node after observing that the car cannot start. This evidence propagates to the rest of the BN and new probabilities are given to the three remaining variables, showing that the *likelihood* of each component to be faulty has been increased. In this example, the three parent variables have differerent level of dependency on $CarStarts$, defined in the CPT of $CarStarts$. This explains why their respective post-inference probabilities are altered in distinct ways. Figure 2.2c shows that following the observation of a fault in the battery, a new evidence is set in $Battery$, that propagates once again to the other nodes. This affects $Sparks$ and $Fuel$ by decreasing significantly their respective

probabilities to be $faulty$ and *empty*. This calculation of probability for any variable is referred to as *inference*. Inference is based on Bayes' rule that enables manipulation of probabilities and which is given in (2.2) for two events A and B. An event defines the information that is known on a set of variables. These variables can either be instantiated or not. The states of instantiated variables are known while those of non-instantiated variables follow a distribution defined by the parameters of the BN.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{2.2}$$

Bayes' rule allows propagation of evidence through edges in any direction. Several methods have been developed in order to perform inference in a BN based on (2.2). For example, one of the best known inference methods is called variable elimination (Cozman, 2000) and consists in inserting evidence in (2.1). Variables X_i that are neither in A or B are removed from (2.1) through calculating independently the individual $P(X_i|Pa(X_i))$ and $P(X_j|Pa(X_j))$, with $X_i \notin \{A, B\}$, $X_j \in \{A, B\}$ and $X_i \in Pa(X_i)$. Note that this process highly reduces the number of entries that need to be handled to calculate a particular event probability in comparison with dealing with an unfactorized expression of the joint probability table.

(a) No evidence

(b) Evidence: car does not start

(c) Evidences: car does not start and battery is faulty

Figure 2.2: BN representation of the simplified car start problem[1]. The effect of evidence insertion is shown by displaying the probabilities relevant to each state of each variable.

Application wise, BNs are used in several fields and for various tasks such as classification (Friedman et al., 1997), decision support (Nicholson et al., 2008) or problem understanding (Kabli et al., 2008). For more details on BNs, we refer the reader to Jensen and Nielsen's 2007 work (Jensen and Nielsen, 2007).

[1] illustration generated from Hugin Expert (Accessed: 22 February 2013), http://www.hugin.com/

2.2 Benchmark Bayesian Networks

2.2.1 Sampling benchmark Bayesian Networks

In order to objectively assess the abilities and performance of BN learning algorithms, a range of BN problems needs to be selected. A common approach to understand the performance of search methods for BN structure learning is to sample known BNs to generate data that can then be used by the search methods to learn a structure. In order to do so and given a BN, Probabilistic Logic Sampling (PLS) (Henrion, 1986) is applied. In PLS, a state is sampled for each node depending on its CPT and consequently on its parents' respective states. Thus, PLS starts generating a value for variables represented by nodes that do not have parent in the BN. More generally, PLS only generates a value for a given variable if all of its parents in the BN structure have already been sampled. Since BN structures are DAGs, PLS is a straightforward task. Once learnt, structures can be compared to the original ones used to create the data. If an algorithm is able to retrieve the same structure, it can be considered as successful at the task. Of course, with problems of different size and complexity, it can be very challenging to retrieve the true structure and approaching it can become an objective. However, it is important to keep in mind that there exist equivalent BN structures among all possible structures. As explained in (Chickering, 1995), two BN structures are equivalent if the set of distributions that they represent is identical. As a consequence, there exist BNs whose structures differ but that will have similar likelihood scores, that is metrics that define the relationship between a given structure and a given dataset. Scores are described in more depth in Section 2.3.

BN learning is a common field of research and thus, it is important to evaluate our methods in a coherent way to what has been done already. However, as the number of studies on the topic is relatively large, the number of benchmark problems that have been used is also large. Here, we describe the problems that are used for the experiments throughout the thesis. The choice of these six particular problems was motivated by

literature and by problem characteristics. *asia*, *car* and *alarm* are some of the most used benchmarks (Cooper and Herskovits, 1992; Kabli et al., 2007; Wu et al., 2011) and were chosen to allow comparisons with other studies. On the other hand, *tank*, *credit* and *boerlage* are less commonly referred to, but offer some distinct properties. While *tank* and *credit* exhibit a rather wide and shallow structure, *boerlage* has a deeper and narrower structure with respect to their respective topologies. Studying the efficiency of the proposed methods on problems with different properties helps reaching more objective conclusions. Illustrations of the different network structures presented in this section were obtained using the Netica software [2] for BN management.

2.2.2 Asia

The *asia* network (Lauritzen and Spiegelhalter, 1988) is also referred to as *chest clinic* in some studies. The *asia* network, shown in Figure 2.3, is a very simple fictitious benchmark that illustrates relationships that exist between characteristics of a patient (visit to Asia, smoking habits, medical test results and presence of bronchitis, tuberculosis or lung cancer). This network is composed of 8 nodes and 8 edges and has a total of 36 parameters. This is usually used to assess the basic functionality of learning algorithms. Algorithms that fail at retrieving the correct structure of *asia* will usually not be able to learn other structures.

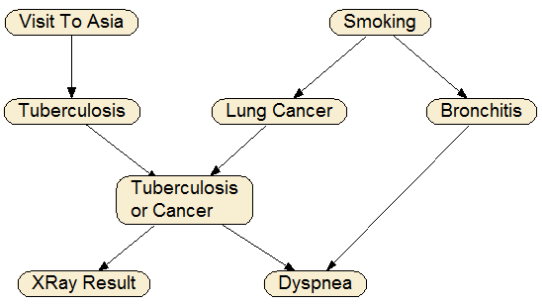


Figure 2.3: Structure of *asia*

[2] available from Norsys (Accessed: 22 November 2011), http://www.norsys.com

2.2.3 Tank

The *tank* network [3] is shown in Figure 2.4. It is composed of 14 nodes and 20 edges, for a total of 92 parameters. The *tank* benchmark BN focuses on the diagnostic of possible explosion in a tank, based on sensor readings. *tank* can be qualified as wide. It has 4 levels and is 5-node wide.

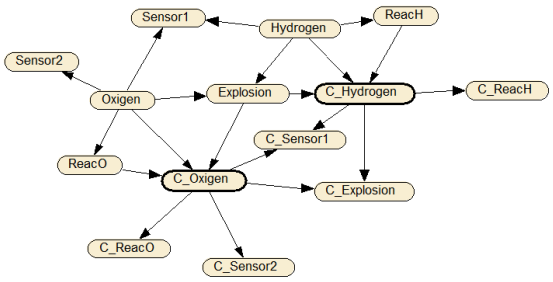


Figure 2.4: Structure of *tank*

2.2.4 Credit

The *credit* network [3] is shown in Figure 2.5. It is composed of 12 nodes and 12 edges for a total of 166 parameters. The *credit* benchmark BN models the relationships between the credit worthiness of an individual and some of his personal details such as employment information and previous bank history. Similarly to *tank*, *credit* is wide and not deep with 4 levels for a 7-node width. However, both differ in the sense that *credit* contains 7 independent nodes, while *tank* only has 2.

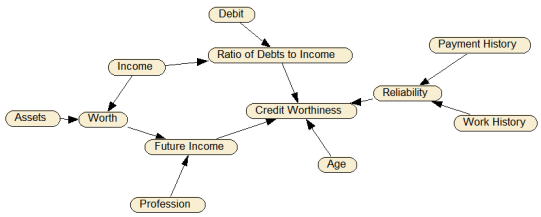


Figure 2.5: Structure of *credit*

[3] Genie and Smile (Accessed: 22 February 2013), http://genie.sis.pitt.edu/

2.2.5 Car

The *car* network [2] is shown in Figure 2.6. It can be used to inverstigate causes related to a car start failure. The *car* benchmark BN is composed of 18 nodes and 17 edges for a total of 121 parameters. *car* can be split in 5 levels and is 10-node wide.

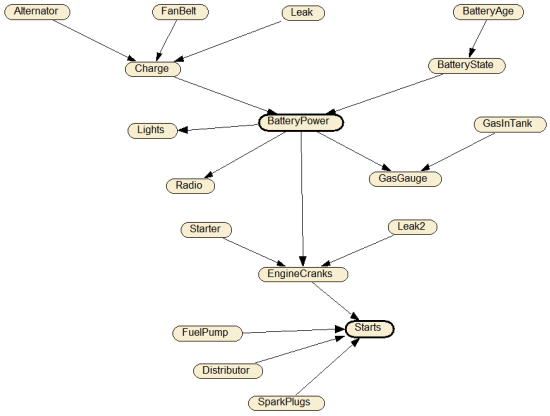


Figure 2.6: Structure of *car*

2.2.6 Boerlage

The *boerlage* network (Boerlage, 1992) is shown in Figure 2.7. It is composed of 23 nodes and 36 edges for 172 parameters. The *boerlage* benchmark BN models relationships between separate events that would not be considered as inter-related at first. For example, in the *boerlage* BN, the relation between the presence of "lots of traffic on 5th street" and the fact that "Tom's cousin is visiting him" can be evaluated. *Boerlage* exhibits the deepest and narrowest structure from the benchmark suite as it is 14-level deep and 3-node wide. Although the problem modelled by this BN may seem unrealistic, it is considered for its topological characteristics that differ from the rest of the benchmarks. Thus, *boerlage* is analysed from a probabilistic perspective and the semantic meaning of its variables is not investigated further.

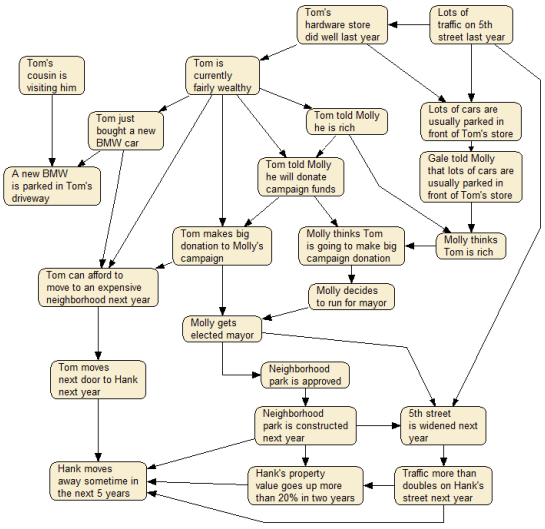


Figure 2.7: Structure of *boerlage*

2.2.7 Alarm

The *alarm* benchmark BN (Beinlich et al., 1989) shown in Figure 2.8 is one of the most widely used benchmark datasets in BN structure learning literature. *Alarm* is based on a real-world domain and provides information on medical diagnosis of false alarm in patient monitoring. The *alarm* network consists in 37 nodes and 46 edges for a total of 752 parameters. Its structure is composed of 11 levels and has a width of 12 nodes.

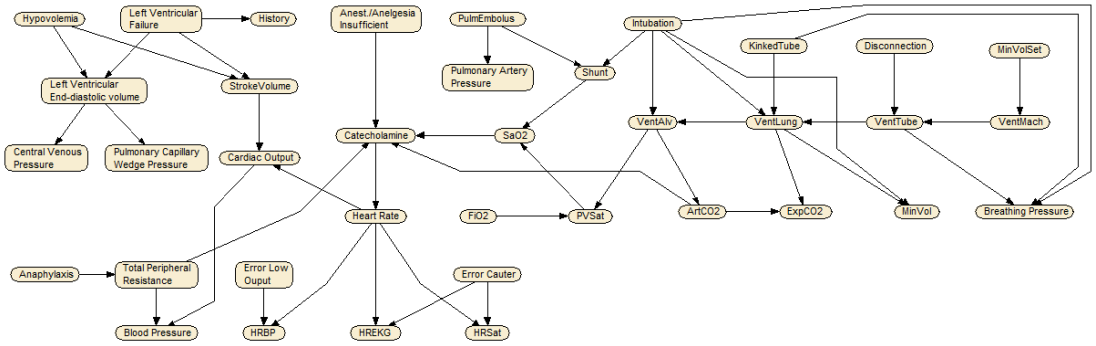


Figure 2.8: Structure of *alarm*

Number of variables	Number of structures
1	1
2	3
3	25
4	543
5	29281
6	3781503
7	1138779265

Table 2.1: Relation between number of variables and number of possible BN structures

2.3 Bayesian Network Structure Learning

Although learning BNs from data consists in learning both structure and parameters, the most challenging task remains to determine the network structure. This is considered as a NP-hard problem (Robinson, 1977) and the number of potential structures that can be drawn from a set of variables grows super-exponentially as the number of variables n increases as illustrated in Table 2.1 for $n \in [1; 7]$. It has been shown that the number of structures that exist is $O(n!2^{\frac{n}{2}})$ (Robinson, 1977). Exhaustive search is therefore not a sensible approach to the problem. Two main families of methods are commonly considered for BN structure learning, respectively based on Conditional Independence (CI) tests and on search and score techniques.

2.3.1 Using Conditional Independence

The core concept of CI-based methods (Cheng et al., 2002) is to discard potential edges that could be drawn. Several approaches exists in which the CI tests between pairs of variables are performed in different order. Yet, the first components that need to be studied are the CI tests. Although there exists a variety of tests, only one is presented in this section. This is to illustrate an example of CI-based method rather than giving a complete overview of the field. Hence, the Mutual Information (MI) is explained along with the process of learning Chow-Liu tree BNs.

Computing the MI between two variables X_A and X_B gives an indication on how related their distributions are. We denote by x_a and x_b the state of variables X_A and X_B respectively. The MI represents how much information will be gained by X_A if information is gained on X_B. Hence, it can be used to define how dependent of each other they are. MI can also be computed under the influence of a set of variables. In such case, the test is called Conditional Mutual Information (CMI). The MI between X_A and X_B is computed as

$$MI(X_A, X_B) \equiv \sum_{x_A} \sum_{x_B} p(x_A, x_B) \log_2 \left(\frac{p(x_A, x_B)}{p(x_A)p(x_B)} \right) \tag{2.3}$$

$$MI(X_A, X_B) = H(X_A) - H(X_A|X_B) \tag{2.4}$$

The MI metric provides a continuum of dependency values that range from 0, for independent variables, to the common entropy of a pair of variables. This property is illustrated in (2.4), which describes the relation between $MI(X_A, X_B)$ and the marginal entropy $H(X_A)$ of X_A and its conditional entropy $H(X_A|X_B)$ given X_B (Cover and Thomas, 1991). Hence, it is needed to set a dependency values threshold on $MI(X_A, X_B)$ beyond which two variables can be qualified as dependent. Varying the threshold makes the learning process discriminative to different extents and hence constraints the number of edges that are set in the structures.

Although Chow-Liu Tree BNs (Chow and Liu, 1968) are not used throughout this work, they represent a good illustration of the use of a CI measure. A BN can be represented as a Chow-Liu tree under two constraints. First, its structure needs to be rooted-tree-shaped, that is only one node in the structure has no parent. Second, each node of the structure can have at maximum one parent. This type of representation ensures that the structure only has a small number of edges. Furthermore, it has been shown that Chow-Liu structures can model efficiently any data distribution with maximum likelihood (Chow and Liu, 1968). Learning Chow-Liu structures is a straightforward

process in which a complete undirected graph is drawn at first. MI is calculated for every pair of variables and MI values assigned to each undirected edge. Based on this information, a maximum spanning tree can then be drawn in $O(e)$ operations, where e represents the number of edges in the original tree as proved in (Kruskal, 1956). The maximum spanning tree should connect all variables by maximizing the sum of the weights from all edges that are kept, or in other words the sum of the associated MI values. The remaining step is to direct all edges. Since the final structure should be a tree, a root node is selected at random. Edges are directed in the outward direction from the root node. Note that this example makes use of MI to determine CI between variables. Alternatives can be used. Chow-Liu trees have also been employed as a BN representation in order to reduce the search space in conjunction with search and score techniques (Carvalho et al., 2007).

As already mentioned, there are several CI-based algorithms. For instance, the PC algorithm (Spirtes et al., 2000) starts from a complete undirected structure and removes edges following CMI tests. As edges are being removed, the CMI tests are conditioned on more neighbors and the structure gets trimmed further. At the end of this step, an undirected structure is obtained. Unlike in the Chow-Liu method, directions are not added by randomly picking a root node, but by following a set of rules. The approach is to first identify potential v-structures in the graph, which are groups of any three variables X_A, X_B and X_C such that there is an edge between X_A and X_C and between X_B and X_C and that X_A and X_B are independent under conditions that do not involve X_C. Such patterns are directed as follows: $X_A \rightarrow X_C \leftarrow X_B$. The second rule aims at avoiding the addition of new v-structures. Thus, patterns such as $X_A \rightarrow X_C - X_B$ are turned into $X_A \rightarrow X_C \rightarrow X_B$. The third rule goes through the graph and directs any edge that could potentially lead to a cycle. Finally, if edges are left undirected, the fourth rule states that they should be directed at random.

Literature on CI-based methods reports successes due to the individual independence tests computed between all variables, making it able to assess all relationships that exist between variables (Neapolitan, 2004). However, this feature of the method also

represents a drawback as results of individual independence tests may present failures at times, resulting in learning structures that do not reflect the true distribution of the data. Since the search is bounded by the CI test results, no other structure can be assessed (Koller et al., 2007). Such observations are particularly likely to occur when the number of variables involved in the learning process is small, increasing the importance of accurate individual test results (Neapolitan, 2004). In the next section, alternatives to CI-based methods are presented that search through the whole set of possible structures.

2.3.2 Search and Score Strategies

As the name may suggest, search and score techniques are composed of two main elements. First, a scoring metric needs to be defined to evaluate the quality of the candidate BN structures. Once BNs can be assessed, a search algorithm can use this information to generate and improve structures. Here, we start by presenting some of the existing scoring metrics for BN structures, before reviewing works that have been carried out on three different families of search and score approaches, namely greedy, evolutionary and estimation of distribution algorithms.

2.3.2.1 Scoring Metrics

Some of the metrics presented in this section use methods from BN parameter estimation. Hence, we first describe the process of Maximum Likelihood Estimation (MLE). A BN M can be represented by different possible parameter sets θ. Each of them implies a particular BN structure S_θ. The configuration of θ and S_θ affects the quality of M, that is its likelihood given a dataset D, composed of cases C_l as expressed in (2.5).

$$L(M|D) = \prod_{C_l \in D} p(C_l|M) \tag{2.5}$$

Because of the relation that exists between θ and S_θ, it can be stated that maximizing the likelihood of a BN is equivalent to finding the unique θ configuration, denoted as $\hat{\theta}$, that fits the best D. This process is performed using MLE:

$$\hat{\theta} = \underset{\theta}{argmax} L(S_\theta|D) \tag{2.6}$$

In the special case of dealing with complete data, that is without missing values, the likelihood of S can be computed as a product of all individual MLEs computed from the parameters relevant to each variable. In more concrete terms, all CPTs of the BN are used individually to compute local MLEs that are brought together in a product. Each local MLE can be defined by simply counting cases from the data. For example, $p(X_i|Pa(X_i)) = \frac{N(X_i,Pa(X_i))}{N(Pa(X_i))}$. Because it is generally easier to deal with sums, the log-likelihood is preferred and defined as:

$$LL(S|D) \equiv \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}\ log\left(\frac{N_{ijk}}{N_{ij}}\right) \tag{2.7}$$

The arrity of X_i is denoted as r_i and the total number of possible configurations of parent states as q_i. The number of instances of variable X_i in its k-th state, while its parents are in their j-th combination is noted as N_{ijk}, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, that is N_{ij} represents the number of instances where the parents of X_i are in their j-th state. Note that this notation is kept throughout the thesis.

One of the most simple scoring metrics directly based on MLE is the Euclidean distance. The term Euclidean distance originates from the continuous domain, in which it represents the distance between two points in a multi-dimensional real vector space. Yet, the concept can be extended to the domain of BNs. In order to measure the Euclidean distance between a given structure S and the data D, the distance between the two probability distributions P_S and P_D that they respectively model can be computed. Thus, a structure that can accurately approach the data distribution will be a good

structure. Of course, the distribution modeled by a BN does not depend only on its structure but also on its set of parameters. Since, S can imply many such distributions, MLE is used in order to determine the parameters $\hat{\theta}$ that bring the distribution closest to the one of the data. The Euclidean distance $dist(P_D, P_{S,\hat{\theta}})$ can be expressed over all covariate configurations x as:

$$dist(P_D, P_{S,\hat{\theta}}) = \sum_{\mathsf{x}} (P_D(\mathsf{x}) - P_{S,\hat{\theta}}(\mathsf{x}))^2 \tag{2.8}$$

Using solely the Euclidean distance in a search and score strategy would lead the best structure to be complete, that is to have edges between all nodes. This is because it is the complete structure that can model most accurately the data distribution. By using a complete structure, conditional dependencies are defined between all variables, in which case there is no factorization of the joint probability distribution, the main purpose of BNs. More generally, a BN with a large number of modelled conditional dependencies needs a greater number of operations and a greater processing time when performing inference than a BN with fewer dependencies. Because such complete structures are not wanted, Euclidean distance is often used in conjunction with penalty or parsimony functions, respectively penalising models for their complexity or rewarding them for their minimal use of resources. For instance, a score can be computed by calculating $dist(P_D, P_{S,\hat{\theta}})$ and subtracting a function of the total number of edges in S. Despite these efforts, there remains a major drawback which is the computational cost involved in the calculation of the Euclidean distance that requires a processing step for every covariate pattern and makes it usable only on very small-dimensional problems.

The Bayesian Information Criterion (BIC), also called Minimum Description Length (MDL) addresses these problems (Lam and Bacchus, 1994). It also takes into consideration both data fitting and structure complexity. For complete data of size N, the BIC score $BIC(S|D)$ can be computed as the difference of $LL(S|D)$ and the output of a penalty function and expressed as:

$$BIC(S|D) \equiv \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}\ log_2(\frac{N_{ijk}}{N_{ij}}) - \frac{log_2 N}{2} \sum_{i=1}^{n} q_i(r_i - 1) \tag{2.9}$$

Since the BIC score is essentially a count over all variables of the data, it is said that BIC is a *decomposable* fitness metric. It can be expressed as a sum of local BIC functions over all variables because both LL and penalty function are computed over all variables. Hence, local changes in the structures, such as addition of an edge for example, do not require computation of the whole BIC, but just of the local BIC scores affected by the alteration of the structure. Decomposability of a scoring metric is a property that is essential to deal efficiently with data of high dimension, without implying a heavy computational cost.

For information, the penalty function represented by the second term in (2.9) is sometimes replaced by $\sum_{i=1}^{n} q_i(r_i - 1)$. In that case, the scoring function is called Akaike Information Criterion (AIC) (Akaike, 1974).

There exists another family of scoring metrics, also decomposable and called *Bayesian* metrics. Such methods are based on approximating the probability of each variable to be in a given state, by approximating the probability distribution of its set of parameters. Given a BN structure, these values can then be used for every possible state of each variable and its parents.

To obtain such metrics, one should express the conditional joint probability $p(S, D|\xi)$ of having both the BN structure S and the data D, given the current state of information ξ, that is the set of all evidences inserted in the BN. Referring to the problem described in Figure 2.2 as an example, ξ could represent the inclusion of the evidence that the variable $CarStarts$ is set to No. However, BN structure learning from data typically does not involve prior knowledge and in most cases, such as in this thesis, ξ can be ignored. Yet and before investigating how $p(S, D|\xi)$ is derived, it is important to refer to the Bayes rule on the problem as in (2.10) in order to understand the distributions involved.

$$p(S|D,\xi) = \frac{p(D|S,\xi)p(S|\xi)}{p(D|\xi)} \tag{2.10}$$

In this thesis, only discrete variables are considered, that is variables that have a finite number of states. Because of this properties, variables in D follow a multinomial distribution. Since variables in S are also variables in D, variables in S follow a multinomial distribution and so do $p(D|S,\xi)$. Note that a multinomial distribution can also be referred to as a categorical distribution. Given this property of the likelihood function, if the prior probability $p(S|\xi)$ and the posterior probability $p(S|D,\xi)$ have conjugate distributions, they must follow a Dirichlet distribution themselves. Two probabilities have conjugate distributions if they have the same mathematical form, but with different parameters (George et al., 1993).

The probability distribution of the set of parameters Θ_X of a variable X can be defined as:

$$p(\Theta_X|D,\xi) = c\prod_{k=1}^{r}\theta_{X=k}^{N_k}p(\Theta_X|\xi) \tag{2.11}$$

X is r-dimensional and each of its states k has $\theta_{X=k}$ as parameter. N_k represents the number of occurrences of X in its k-th state in the dataset D. $p(\Theta_X|\xi)$ is the probability density function of Θ_X, given the current state of information ξ. Finally, c is a normalization constant.

One of the required step to use (2.11) is to define $p(\Theta_X|\xi)$. As seen previously, the Dirichlet distribution represents the probability of a set of rival events, given some information on previously observed occurences of these events (Heckerman et al., 1995). The general expression for the Dirichlet distribution $Dir(X)$ of variable X is given in (2.12). Note that Γ is the Gamma function which is equivalent to $\Gamma(a) = (a-1)!$, for any positive integer a.

$$Dir(X) = \frac{1}{B(N')} \prod_{i=1}^{r} X_i^{N'_i - 1}, \text{ where } B(N') = \frac{\prod_{k=1}^{r} \Gamma(N'_k)}{\Gamma(\sum_{k=1}^{r} N'_k)} \tag{2.12}$$

To simplify the understanding, $\frac{1}{B(N')}$ is regarded as a constant and will be referred to as the normalization constant c.

Using a Dirichlet distribution, it is possible to define the expectation of X to be in its k-th state by means of N'_k, function of ξ. For instance, such expectation $E(\Theta_X = k)$ can be formulated as (2.13). $E(\Theta_X = k)$ actually represents the mean of the distribution of X_k. We refer to the different N'_k as the hyperparameters of the Dirichlet distribution.

$$E(\Theta_{X=k}) = p(X = k|\xi) = \frac{N'_k}{N'}, \text{ where } N' = \sum_{k=1}^{r} N'_k \tag{2.13}$$

Finally, under Dirichlet distribution, (2.11) can be extended to (2.14).

$$p(\theta_{ij}|D, S, \xi) = c \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk} - 1 + N_{ijk}} \tag{2.14}$$

As demonstrated in (Heckerman et al., 1995), $p(D|S, \xi)$ can be expressed by means of expectations over all cases C_l of D, $1 \leq l \leq m$, leading to:

$$p(D|S, \xi) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \prod_{l=1}^{m} E(\theta_{ijk}|C_1, ..., C_{l-1}, S, \xi) \tag{2.15}$$

Since $p(S, D|\xi) = p(S|\xi)p(D|S, \xi)$ and from (2.13), (2.14) and (2.15), the joint probability $p(S, D|\xi)$ can be expressed as:

$$p(S, D|\xi) = p(S|\xi) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \tag{2.16}$$

It is now possible to use (2.16) to calculate the likelihood function of a particular BN

structure, that is (2.16) without computing $p(S|\xi)$. Note that $p(S|\xi)$ is not computed because it is a constant term when no evidence is inserted. This fitness function is called the Bayesian Dirichlet (BD) function. However, this metric cannot be used in practice as it would require to determine all N'_{ijk} hyperparameters of the Dirichlet distribution. As a solution, there exists a particular case of BD proposed by Cooper and Herskovits (Cooper and Herskovits, 1992), where it is assumed that $N'_{ijk} = 1$. This metric is called the CH score and is defined as:

$$p(S, D|\xi) \equiv p(S|\xi) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2.17}$$

Alternatively, the BD score can be re-formulated in order to only leave one hyperparameter. This can be done by assuming that $N'_{ijk} = N'p(X_i = x_{ik}, Pa(X_i) = w_{ij}|S)$ and that $N'_{ij} = N'p(Pa(X_i) = w_{ij}|S)$. Here, w_{ij} denotes the j-th combination of $Pa(X_i)$ and N' represents the sum of all hyperparameters of the Dirichlet distribution. When the BD score is represented as such, it is called the likelihood-equivalence Bayesian Dirichlet (BDe). This is still not a practical scoring metric since it is still necessary to calculate all $N'_{ijk} = N'p(X_i = x_{ik}, Pa(X_i) = w_{ij}|S)$ for all combinations of i, j and k. However, BDe represents a step towards another metric called the uniform BDe (BDeu) (Buntine, 1991), in which it is assumed that $p(X_i = x_{ik}, Pa(X_i) = w_{ij}|S) = \frac{1}{r_i q_i}$. This assumption means that all prior joint probabilities on X_i and $Pa(X_i)$ are uniform and equal to $\frac{1}{r_i q_i}$. Consequently, the BDeu score is defined as:

$$p(S, D|\xi) \equiv p(S|\xi) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\frac{N'}{q_i})}{\Gamma(\frac{N'}{q_i} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\frac{N'}{r_i q_i} + N_{ijk})}{\Gamma(\frac{N'}{r_i q_i})} \tag{2.18}$$

Although the implementation of BDeu has helped reducing the number of hyperparameters of the metric, the value of N' remains of importance. Yet, it is not possible to calculate this value for a given BN problem. Hence, in order to use BDeu, one should consider different N' values until reaching satisfactory results.

It is important to stress that the metrics that are presented in this section assume that the data is complete. In situations where this is not the case, some methods can be used in order to infer the missing values. We refer the reader to the Expectation Maximization algorithm (Dempster et al., 1977). Note that the data used throughout this thesis does not exhibit missing values.

2.3.2.2 Greedy Search: the example of K2

Since scoring metrics have been presented, search strategies should now be investigated. First, we focus on explaining the concept of greedy search and present a particular example of such methods, based on the CH score presented in (2.17). In a general greedy approach, a set of local operators is defined and applied to a solution in order to improve its quality. In the context of BN structure learning, such solution is a structure S. At each step of the greedy search, each defined operator is applied to S. The effect of each operation is evaluated and the most beneficial with respect to the improvement of the quality of S is determined. This represents the best neighbor S' of S. S' is evaluated and compared to S. If applying the operator is beneficial, that is the quality of S' is greater than S, S' replaces S as the current best structure. The cycle is repeated until no operator can improve S.

The K2 greedy algorithm (Cooper and Herskovits, 1992) is a specific greedy algorithm that makes use of an ordering of nodes that is provided as an input. The K2 algorithm evaluates addition of potential edges between variables. An edge can only be added from a variable A to a variable B if A belongs to the predecessors of B in the ordering. This constraint ensures that no cycle can be introduced in the structure and thus that the structure remains a DAG. In the K2 algorithm, local CH scores are computed for each variable X_i, given different parent sets $Pa(X_i)$. The local $CH(X_i, Pa(X_i)$ score for a variable X_i and its parent set is taken from (2.17) and is summarized as

$$CH(X_i, Pa(X_i)) \equiv \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2.19}$$

At each step, one edge is added between X_i and the variable that maximizes the local CH score over all variables that are predecessors of X_i in the ordering, that is the set $Pred(X_i)$. Note that adding a variable to the set of parents of a given node does not always improve the related CH score as described in Appendix A. In addition, a threshold $maxParents$ on the number of parents a node can have can be decided prior to running K2. This is to ensure the structure that is learnt remains reasonably complex. Of course, this threshold needs to be set manually, with respect to the problem domain. Algorithm 1 presents the outline of K2.

Algorithm 1: $K2$

for each variable X_i, $i \in [0; n[$ **do**
  $Pa(X_i) = \varnothing$
  $stop = false$
  while $\overline{stop}$ and $|Pa(X_i)| < maxParents$ **do**
    Choose X_l, such that $CH(X_i, Pa(X_i) \cup X_l) > CH(X_i, Pa(X_i) \cup X_k)$, for any $X_k \in Pred(X_i)$, $k \neq l$
    if $CH(X_i, Pa(X_i) \cup X_l) > CH(X_i, Pa(X_i))$ **then**
      $Pa(X_i) = Pa(X_i) \cup X_l$
    else
      $stop = true$
    end if
  end while
end for

With deterministic algorithms such as K2, a starting ordering of nodes is required. Modification of the ordering leads to different candidate BN structures. In (Cooper and Herskovits, 1992), starting orderings were always such that the K2 process could retrieve the exact structure. Alternatively, K2 can be used based on random starting orderings (Carvalho, 2011), but this is unlikely to lead to the optimum structure. Ideally one would assess all possible orderings to allow K2 to reach an optimal solution. Although this is possible on small problems, it can become impractical on larger dimensions as the number of orderings for a dataset of n variables is $n!$. To search both spaces of

structures or orderings, it has become sensible to turn to methods that are able to iteratively improve solutions such as EAs.

2.3.2.3 Bayesian Networks and Evolutionary Algorithms

EA is a paradigm inspired by nature that aims at generating candidate solutions to a given problem based on characteristics of previously selected solutions. A key distinguishing feature of EAs lies in the fact that their search is driven by an evaluation-based selection. As a result, solutions are improved over time, as the search progresses, mimicking natural evolution until reaching a satisfactory solution quality.

GAs (Holland, 1975) are probably the most well known type of EAs. In a GA, solutions are generated at random to form a so-called *population*. Each solution in the population is evaluated using a fitness function. Some solutions are then selected relative to their fitness and recombined in order to create new solutions, called *offspring*. The idea is to mix characteristics of good solutions to enhance the chances for the offspring to exhibit high fitness values. Note that in order to evaluate solutions and perform recombination, a way to represent a solution to the problem needs to be thought of. For example, solutions can be represented as bit strings, real value vectors or as permutations. A permutation defines a particular combination that can be created from a given set of variables, where each of the variables is represented exactly once. In the context of BNs, node orderings can be represented as permutations.

Since GAs are inspired by genetic evolution, a solution is often regarded as a *chromosome* or an *individual*, while each element of it is referred to as a *gene*. In a GA, there are typically two recombination operators, which are the crossover and the mutation. During crossover, parts of the selected solutions are exchanged as blocks. The effect of crossover depends on the type of crossover and the domain in which it is applied. For example, one point crossover picks at random a point pt in two selected solutions S_1 and S_2. Two solutions C_1 and C_2 are generated. The first child solution C_1 is composed of all genes placed before pt in S_1 and all genes placed after pt in S_2. Similarly C_2 takes

genes from S_2 and S_1. One point crossover is probably the simplest operator and can be applied on many domain, such as bit string representations. Yet, one point crossover is not applicable to the permutation domain. Although several crossover types exist for combinatorial problems, we present the process of using cycle crossover (Oliver et al., 1987), which is used throughout this work. In the example illustrated in Figure 2.9, a random gene position (position 0) is selected and genes at this position (8 and 1) exchanged between S_1 and S_2. Since the new gene imported in S_1 is duplicated, the next genes to exchange are in position 7, where the element 1 can be found in S_1. Once the exchange is done, 6 is the new duplicated gene in S_1. Thus, the other instance of 6 (at position 6) is exchanged. Because the new element imported in S_1 is 8, the starting element, a cycle has been completed and the crossover is complete, exhibiting two new offsprings C_1 and C_2. Intermediate steps are shown in Figure 2.9.

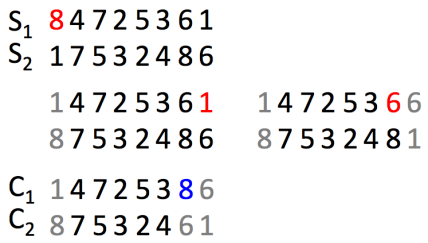


Figure 2.9: Example of cycle crossover

The second recombination operator is mutation, where a solution is altered randomly. This is motivated by the need to avoid early convergence in GAs. As the search progresses, characteristics of the good solutions replace those of less fit ones. This may lead the population to reach convergence where all solutions are similar. Hence, at times, it may be good to introduce perturbation in the search in order to explore a different area of the search space. On permutation representations, there are several mutation types. One of the simplest is the swap mutation in which two randomly picked genes are swapped within the same chromosome. Since mutation types are the topic of Chapter 5, they are not investigated in further details here. Because mutation is disruptive with respect to the chromosome, a mutation rate is set that defines how likely mutation is to happen. Similarly, a crossover rate can be set. Once produced,

offspring solutions are evaluated and compared with the population. If an offspring is better than the worse solution, it takes its place and the population is re-ordered. Depending on the GA implementation, the number of offspring can vary. This process is called a *generation* and is repeated until reaching a stopping criterion.

EAs also encompass other heuristics that have proved successful on many problems including BN structure learning. Ant Colony Optimization (ACO) (Colorni et al., 1991) mimics the behavior of ants foraging for food, that is exploring possible paths to reach a food source. ACO is a technique particularly well adapted to problems that can be represented as graphs, such as BN structure learning, where nodes can be drawn within a graph, or combinatorial problems more generally. Similarly to real ants, ACO ants explore possible solutions by choosing their next destination (e.g. next node) based on traces of pheromones left on the different paths by previous ants. As the search progresses, the best paths, which represent parts of the best solutions to the optimization problems, gather more pheromones. This makes these paths more prone to be part of the next solutions produced by future ants. The two examples of EAs chosen here, namely GA and ACO, have been used for BN structure learning in the past (Larrañaga, Kuijpers, Murga and Yurramendi, 1996; De Campos et al., 2002; Parpinelli et al., 2002; Kabli et al., 2007; Wu et al., 2010).

Finally the last EA technique to be introduced here is Evolutionary Programing (EP) (Fogel et al., 1966), which has seen a few applications to BN structure learning. In EP, a set of operators is defined. Starting from a random population of solutions, EP goes through several generations in which a chosen number of solutions is selected in a similar fashion as in a GA. Each selected solution undergoes a mutation step where an operator from the list of operators is applied. Once mutated, solutions are evaluated and inserted in the population relative to their respective fitness. Note that no crossover is used in EP. In the domain of BN structure learning, operators may include for example insertion or deletion of an edge, redirection of an edge or creation of v-structures. This is not an exhaustive list and more examples can be seen in (Cotta and Muruzábal, 2004; Wong et al., 1999).

As explained previously, K2 introduces the notion of variable orderings, which reduces considerably the search space. Instead of having to look through all candidate structures, the search focuses on finding a good ordering before setting it as an input to K2. However, this is not totally advantageous since such strategies will not consider all structures and hence miss some parts of the BN structure space. We recall that this can represent a drawback as the final output of BN structure learning is to find a structure rather than an ordering of nodes. Because of this properties, it is possible to split all methods that have been implemented into two families, those searching through the space of structures and those searching the space of orderings (Larrañaga, 2010).

One of the earliest work on searching the space of BN structures by means of EAs is described in (Larrañaga, Poza, Yurramendi, Murga and Kuijpers, 1996) where solutions are represented by a $n \times n$ connectivity matrix. This matrix defines the presence or absence of edges between the n variables of a problem. Two approaches are investigated. First, no restriction is made on edges that can be set. With no restrictions, genetic operators generate solutions that may not be acyclic and thus that are outside the problem domain. Through the introduction of a repairing step, edges that create cycles are removed until the structure becomes a DAG. In the second strategy, an ordering of nodes is considered and operators do not generate infeasible solutions, discarding the repairing step previously proposed. The space of structures was explored using a GA and solutions evaluated using CH score. Because the space of structures is too large, efforts have been made to reduce its dimension. These include combining both CI-based methods and EAs. In (Van Dijk et al., 2003), CI tests are run such as in the PC algorithm. This step leads to a skeleton graph, representing the edges that are most likely to be found in the optimal structure. This is also the search space on which a GA can run, where each gene of a solution represents a distinct edge of the skeleton. Note that the authors quantify the time dedicated to building the skeleton to 45 % of the total algorithm runtime.

Hybridization of CI and EA is also the topic of (Wong and Leung, 2004). Chi-square tests are performed for every pair of variables, based on the data. P-values representing

the degree of relationships between tested variables are recorded, but no skeleton graph is built as it could omit important edges. In the step that follows, a GA is used to generate structures. Before being scored, these are compared to the matrix of χ^2 p-values and discarded if under a certain threshold. This threshold varies as the search goes on and becomes more discriminative.

On the other hand lie EAs that search through the space of node orderings. A first insight of using an EA with K2 is the work presented in (Hsu et al., 2002). In this work, the K2 algorithm is part of the fitness calculation but the final CH score on which it is based is not the fitness value returned to the GA. Instead, K2 determines the best BN structure given an ordering. This structure is assessed by running a holdout validation where evidences are set on selected nodes in the BN and the Most Probable Explanation (MPE) (Sy, 1992) computed on the remaining nodes. The results of the MPE step can be compared with the holdout set and a fitness value derived from it. In actual fact, the approach described in (Hsu et al., 2002) is based on the earlier work by Larrañaga et al. (Larrañaga, Kuijpers, Murga and Yurramendi, 1996), in which the validation process is not performed. Instead, the CH score corresponding to the best BN found by K2 is used to guide the GA search. This algorithm is called K2GA and is presented in Algorithm 2 for a population *pop*, a dataset D, crossover and mutation rates δ_{cr} and δ_{mu} and a maximum of $genMax$ generations. Because the K2 process implies the computation of many local CH scores, it can be an expensive process on large data. This problem is addressed by Kabli et al.(Kabli et al., 2007) by investigating a surrogate to the K2 process. Kabli et al. makes the assumption that a chain structure gives sufficient information about the quality of an ordering to efficiently guide the GA search. A chain structure is a BN structure in which each node is the parent of its successor. Once the GA has converged, a proportion of the best orderings is fed into K2 in order to get their actual structures. This algorithm is referred to as ChainK2GA. A similar approach has been taken in (Wu et al., 2010) where K2 is associated to an implementation of ACO. In addition, a chain surrogate approach is also implemented. Experimental results on a suite of benchmarks show

that both ChainK2 and K2 methods have advantages, although benefits of using the chain surrogate are only seen on some specific problems (Wu et al., 2011). It was suggested that the structure of the original BNs may have an influence which led to the implementation of a hyperheuristic strategy based on problem characteristics that can help further understand what approach should be used (Wu et al., 2012). Although most of the aforementioned studies used benchmark problems, K2-based algorithms have also been successful at modelling real-world data (Cano et al., 2004; Oteniya, 2008; Wiggins et al., 2008; Kabli et al., 2008; Fournier et al., 2010).

Algorithm 2: *K2GA*

```
Initialize population pop
for each solution sol_i ∈ pop, i ∈ [1; |pop|] do
    Run K2 on sol_i and get structure S_i
    Set fitness of sol_i, f_i = CH(S_i, D)
end for
Initialize number of generations gen = 0
repeat
    Sort pop by fitness in descending order
    Select 2 parents from pop using selection method σ
    Generate offspring sol_c by applying crossover operator at rate δ_cr
    Apply mutation operator to sol_c at rate δ_mu
    Run K2 on sol_c and get structure S_c
    Set fitness of sol_c, f_c = CH(S_c, D)
    if f_c > f_|pop| then
        sol_|pop| = sol_c & f_|pop| = f_c
    end if
    gen++
until gen = genMax
```

Other approaches have investigated the use of hybrid strategies that search both spaces of ordering and structures. This is the case in (Lee et al., 2008), which proposes a dual representation GA to search the space of all possible network structures. A solution is represented by an ordering and a connectivity matrix that states the presence of edges between every pair of nodes. At each generation, both ordering and connectivity matrix are evolved and solutions assessed by means of CH score.

2.3.3 Bayesian Networks and Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs), first introduced in (Baluja, 1994), are another type of search and score strategy. Their principle is to model the distribution of the genes from selected solutions, often the best ones. Similarly to a GA, an EDA starts by generating solutions at random that are scored against a fitness function. Once evaluated, the population contains information about the genes that are correlated to high fitnesses. Yet, this is not simple to observe and modeling techniques can be used at that stage. For example, the top 20% of the ranked population can be used to build a model. This model then contains information about the best solutions encountered so far. Thus, sampling such models increases the chances to produce relatively good solutions. Generated solutions are evaluated and inserted in the population. The model can be updated and new solutions created until reaching a stopping criterion. A simple illustration is to study the Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein and Paaß, 1996) that only considers univariate distributions, that is no relationship among genes is modeled. On binary problems, UMDA inherent model assigns a probability to each gene of the solution to be true, based on the frequencies of true statements observed for each gene in the selected solutions. To generate a new solution, a number is drawn in $[0; 1]$ for each gene and compared to the relevant probability value. Of course there exist many other approaches and depending on the optimization problem, modeling interactions between variables may be beneficial. In this specific task, PGMs have proved to be successful, including the use of Markov Networks (Shakya and McCall, 2007) and BNs. Among BN-based implementations of EDA, the Bayesian Optimization Algorithm (BOA) (Pelikan, 2005) and the Estimation of BN Algorithm (EBNA) (Etxeberria and Larranaga, 1999) are the most commonly applied (Santana, 2011). For both, learning the BN model structure from the selected solutions is an essential task. In BOA, a greedy algorithm starts from an empty structure and add edges based on their effect on the structure's associated CH score. In EBNA, the difference lies in the fact that at each generation, the greedy search does

not start from an empty structure but from the structure obtained at the previous generation, since it is known that local search performs better when starting from solutions that are already good (Heckerman, 2008). Note that in the original implementation, the local search in EBNA is guided by calculation of the BIC score for all proposed structures. In both BOA and EBNA, solutions are sampled using PLS.

In addition to using BNs to model the gene distributions, EDAs have also been applied to the actual problem of BN structure learning. In (Romero et al., 2004), two EDAs, UMDA and Mutual Information Maximization for Input Clustering (MIMIC) (De Bonet et al., 1996) are used in a similar fashion than K2GA, to sample node orderings that are fed into K2 and assessed by means of CH score. Since UMDA and MIMIC are not designed to handle combinatorial problems, the encoding is altered and solutions are represented as vector of integers, each representing a factor of $n!$, the number of possible permutations for a problem of size n.

To our knowledge, EDAs for combinatorial optimization have been developed but have not been applied to BN structure learning. These are reviewed in (Ceberio et al., 2012).

2.4 Solution Quality

In search and score strategies, solutions can be compared based on their fitness values. In some problems, this can represent objectively the actual quality of solutions. In the particular case of BN structure learning, the fitness value obtained from the metrics described in 2.3.2.1 is often an approximation of the actual quality of the BN. On the general picture, improvement in terms of fitness generally means that the quality of the BN is also improved. Yet, at smaller scale, fitness and structure quality do not vary commensurately. In this section, we define similarity measures that can be used to assess the actual quality of a solution. This selection of measures acts on both spaces of orderings and BN structures. Note that all these methods compare a given solutions to another one. Typically, when using benchmark problems, a given solution would

be compared to a known optimal or to the true BN structure. For more permutation measures, we refer the reader to (Sevaux et al., 2005; Schiavinotto and Stützle, 2007).

2.4.1 Exact match distance

The exact match distance between two orderings is very simple to compute as it counts the number of elements that are at similar positions i in two permutations p and q of size n. Hence, the higher the exact match distance is, the more similar the two solutions are. Formally, the exact match distance is expressed as:

$$exactmatch = \sum_{i=1}^{n} x_i, \text{ with } x_i = \begin{cases} 0 \text{ if } p_i = q_i \\ 1 \text{ otherwise} \end{cases} \tag{2.20}$$

2.4.2 Edit distance

The edit distance gives a similar kind of information as the exact match distance. Edit distance represents the number of editing steps that need to be performed to obtain the permutation q from p. Since editing steps can differ between problems, these need to be stated before being able to compute the edit distance. In the present work, three editing operators are considered in the specific order: swap, remove and insert. Hence, to compute the edit distance between p and q, the number of elements that can be swapped is first counted. Next, other elements whose positions need to be changed are first removed and then inserted at the targeted position. Although, the edit distance allows discrimination between editing operators, the three selected operators are set with the same weight and thus considered as equal in the following experiments.

2.4.3 Kendall-Tau distance

Pairwise Kendall-Tau Distance (KTD) between two permutations p and q is calculated following Equation (2.21) where the elements i and j belong to the set P of unordered

pairs of elements obtained from p and q. $K_{ij}(p,q)$ equals either 0 if i and j are in the same order, or 1 if i and j are in a different order in p and q.

$$KTD(p,q) = \sum_{\{i,j\} \in P} K_{ij}(p,q) \tag{2.21}$$

In this work, KTD is also used to assess the diversity in a population of node orderings. All pairwise KTDs are calculated from every pair of solutions in the population. These are then averaged. Population KTD has been used for this purpose in (Dos Santos et al., 2010).

2.4.4 Structural Quality and Structural Hamming Distance

The similarity measures presented up to now operate in the space of orderings. However, as mentioned previously, it is essential to also assess the BN structure that is learnt. A straightforward approach is to compare the learnt structure to the original structure used to sample the data. By doing so, edges that are learnt can be qualified in different way. A learnt edge is *correct* if it is set in the same direction as in the true structure. It is *reversed* if it is set in the true structure with the opposite direction. An edge is *added* if not present in the true structure. Finally, an *omitted* edge refers to an edge present in the true structure but not learnt.

These four properties can be used as similarity measures themselves. For instance, counting the number of correct edges C is an indicator of BN structure quality. However, it can be argued that C is not an objective similarity measure since the number of reversed R, added A and omitted O edges also need to be considered. This is the reason why several metrics combining C, R, A and O have been developed (De Jongh and Druzdzel, 2009). In this work, we use three metrics. This may help anayzing results with respect to different contexts. Reversing an edge in a BN may still represent the same probability distribution. This is because the associated CPTs would also be affected by the change. Hence, in some contexts, reversed edges can be considered

as correct, although this is a problem dependent property. The number of relevant edges, rel, represents all learnt edges that are also found in the true structure, without consideration of the edges direction. It is computed as the sum of C and R. On the other hand, added edges and omitted edges represent approximations of the original structure. Thus, an objective of search techniques is to minimize their number. We call the sum of A and O, the number of erroneous edges err. Finally, the Structural Hamming Distance (SHD) is a measure more commonly used (Brown et al., 2005; De Jongh and Druzdzel, 2009). SHD represents the total number of differences between two structures, that is SHD=R+A+O.

2.5 Summary

- BNs are probabilistic graphical models in which variables of a dataset can be represented as nodes and conditional dependencies as directed edges. BN is a powerful tool to model data in presence of uncertainty. Typical uses of BNs include problem understanding, classification or decision support. One of the major tasks surrounding the use of BNs is to learn the structure from data, a NP-hard problem.

- In order to assess different approaches to learn the structure, a common strategy is to use known benchmark BNs to sample data. This data can be used by the proposed search algorithms to produce structures that can be compared to the original ones. Here, six benchmark BNs are considered that present different characteristics: *asia*, *tank*, *credit*, *car*, *boerlage* and *alarm*.

- There exist two main families of methods used for BN structure learning. These are either based on conditional independence tests or are search and score approaches.

- Search and score methods are based on producing solutions according to various mechanisms before assessing them using a so-called fitness function. In the case of

BN structure learning, a common fitness function is the CH score that is derived from a more general function called the Bayesian Dirichlet (BD). Search and score strategies differ from each other in the way they make use of the information given by the fitness of a solution.

- Greedy search is a particular type of search and score strategy in which the solution that is produced at each generation gives the best improvement of fitness over all neighbours of the current solution. One such technique is called K2. In K2, an ordering of nodes is used as an input and edges determined. Starting from an empty network, K2 adds at each iteration the edge that maximizes the CH score until no improvement can be obtained.

- K2 introduces a reduction of the search space. It makes it possible to search for an optimal ordering that can be used as an input in K2, rather than searching through the entire space of structures. K2GA is a GA that search for an optimal node ordering. At each generation, an ordering is produced and used in K2 to determine a set of edges and return an associated CH score. Other approaches using EAs or EDAs have been used to learn BN structures.

- Since many implementations are being explored in this work, several similarity measures are presented that help assessing the quality of the solutions produced. This includes measures that refer to the quality of a node ordering such as *exact match*, *edit* or *Kendall-Tau* distances, but also measures such as SHD that evaluate the quality of the actual BN resulting from an ordering.

Chapter 3

Difficulty of Node Ordering Permutation Optimization

3.1 Search Space Analysis

In this section, the fitness landscapes associated with the different benchmark datasets are assessed. This is aimed at highlighting and understanding the difficulties that can be faced by search and score algorithms. In addition, analyzing the fitness landscape can give indication on efficiency of algorithms and assist decision making regarding algorithm design (Marmion et al., 2012; Wu et al., 2012).

3.1.1 Correlation between fitness and distance

A common approach to studying fitness landscape is to calculate a search space's *Fitness Distance Correlation (FDC)* (Jones and Forrest, 1995). FDC gives an indication on how deceptive a problem is for optimization methods. FDC is usually computed following a *random walk*, that is a sequence of solution generations and evaluations, in which each solution is generated from the previous one using a chosen operator. In the present setting, the swap mutation is used as operator. The swap mutation generates

new permutations by swapping randomly two items from a given permutation. More details on this operator are given in Chapter 5. At each step of the walk, the fitness of the solution is calculated and its distance to the best known solution is stored. A FDC graph is obtained by plotting the distance to the optimum against the fitness of a solution. Thus, if the general curve generated has a negative gradient, the landscape is adapted to optimization methods. A negative gradient means that as a solution is drifting away from the optimum in terms of distance, its fitness also decreases. On the other hand, some search spaces exhibit positive FDC gradients, meaning that the problems are deceptive. Such problems are difficult to solve using search and score approaches because the fitnesses of solutions increase when their respective distances to the optimum increase, a counter-intuitive property. More generally, landscapes are often compared with respect to their correlation coefficients r that quantifies the aforementioned gradient, following (3.1) and obtained from the $\{f_i, d_i\}$ pairs representing correlation between fitness and distance of the solution obtained at step i in the random walk. Note that $\bar{f}$ and $\bar{d}$ represent the mean fitness and distance, while s_F and s_D their associated standard deviations. Using r for comparisons, landscapes are classified into three categories: straightforward if $r \leq -0.15$, difficult if $-0.15 < r < 0.15$ and deceptive or misleading for $r \geq 0.15$ (Jones and Forrest, 1995).

$$r = \frac{C_{FD}}{s_F s_D}, \text{ with } C_{FD} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d}) \tag{3.1}$$

In order to compute FDC from the random walk, it is needed to know the best solution. In the case of BN structure learning using a permutation representation, it is possible to define the global optimum by means of structure, but not in terms of variable orderings. This is due to the fact that the optimal structure can be created from several distinct variable orderings. For this reason and in order to define reference solutions in the space of orderings, it is important to introduce a few concepts from topological sorting. We consider two approaches, namely the *Kahn* (Kahn, 1962) and *Tarjan* (Tarjan, 1976) algorithms. A Kahn topological sort is sometimes referred to as a *width-first* topological

sort. Given a BN structure $\mathcal{S}$, each node X_i whose parents belong to the set $Pa(X_i)$ is assigned a level $\mathcal{L}_j$ such that $X_i \in \mathcal{L}_j$ if $Pa(X_i) \subseteq \mathcal{L}_{j-\alpha}$, where $\alpha > 0$ and if $Pa(X_i) \cap \mathcal{L}_{j-1} \neq \varnothing$. $\mathcal{L}_0$ is defined as the set of nodes for which $Pa(X_i) = \varnothing$. Based on these levels, it is possible to generate orderings of nodes that are topologically sorted in a width-first manner. The so-called Kahn optimal orderings are created by taking nodes from each level $\mathcal{L}_j$, until all nodes from $\mathcal{L}_j$ have been picked. Nodes from $\mathcal{L}_{j+1}$ are then selected and set in the ordering. Note that the process starts by sampling nodes from $\mathcal{L}_0$. The second type of topological sort, Tarjan sort, has a *depth-first* strategy where nodes are sampled starting from $\mathcal{L}_0$ and following directed edges to chose the next node to select. A node can only be picked if all his parents have already been picked. Both methods provide orderings that could theoretically lead to the true structure after running the K2 greedy search. Of course, it is important to recall that although these orderings are considered as optimal, K2 may not retrieve the true structure for all of them since some parameters influence the K2 results, such as the maximum number of parents a node can have or the size of the data. Consequently, in order to create k optimal orderings, $\frac{k}{2}$ orderings are generated based on each topological sort. This ensures that the choice of topological sort does not bias the study (Wu et al., 2012).

FDC was calculated for each selected benchmark dataset, containing 3000 instances. 30 random walks were run, starting from optimal solutions. Each random walk involved 100 steps and distance was computed by means of KTD to an optimal ordering. Final FDC values were obtained by averaging over the 30 result sets and are available from Table 3.1, with standard deviations in brackets. Overall results suggest that the different problems are fairly straightforward to solve by search and score methods. Yet, this is only considering FDC and further analysis needs to be done such as studying the ruggedness of the landscape. *Autocorrelation* (Weinberger, 1990) is commonly used for this purpose. A landscape is rugged when the difference in fitness of neighboring solutions is important. The autocorrelation AC_k is computed following a random walk of length L and depends on the number of steps k that is chosen to define neighbor solutions. For example, AC_3 measures the fitness difference between two solutions distant

by 3 random walk steps. AC_k is calculated following (3.2). Note that each AC_k needs to be compared with a threshold value $\theta = \frac{2}{\sqrt{L}}$ (Weinberger, 1990). By definition and to avoid approximations when L is small, AC_k cannot be objectively analyzed when $AC_k < \theta$. In our experimental setting, $\theta = 0.2$.

$$AC_k = \frac{\sum_{i=1}^{L-k}(f_i - \bar{f})(f_{i+k} - \bar{f})}{\sum_{i=1}^{L}(f_i - \bar{f})^2} \tag{3.2}$$

The closer to 1 an AC_k value is, the less rugged the fitness landscape is. Consequently, the results presented in Table 3.2 indicate fairly smooth fitness landscapes for all problems when $k = 1$ since $0.85 < AC_1 < 0.93$. From Figure 3.1, although it might seem that under increase of the distance k, the AC_k decreases gently, comparisons with other correlation analysis studies suggest the contrary. For comparison purposes, one can refer to (Hordijk and Kauffman, 2005), where the autocorrelation is computed in a similar manner for various NK landscapes. NK landscapes (Kauffman, 1989) represent a meaningful application to compare with as the degree of ruggedness involved with them can be tuned by varying the K value. In NK models, the K value represents the epistasis in the landscape associated with a problem of size N, that is the degree of correlation between solution genes with respect to the fitness. A NK landscape with K=0 means that the fitness can be computed by assessing independently each gene of a solution and by summing these independent contributions to the fitness. More generally, for $K = \kappa$, the fitness of a solution is obtained by summing the fitness contributions of groups of κ neighboring genes. Thus, the highest the value of K is, the more rugged the landscape is. Based on the set of figures given in (Hordijk and Kauffman, 2005), the autocorrelation curves of Figure 3.1 fit between those obtained by the NK landscapes with respective K values equal to 10 and 20 and $N = 100$.

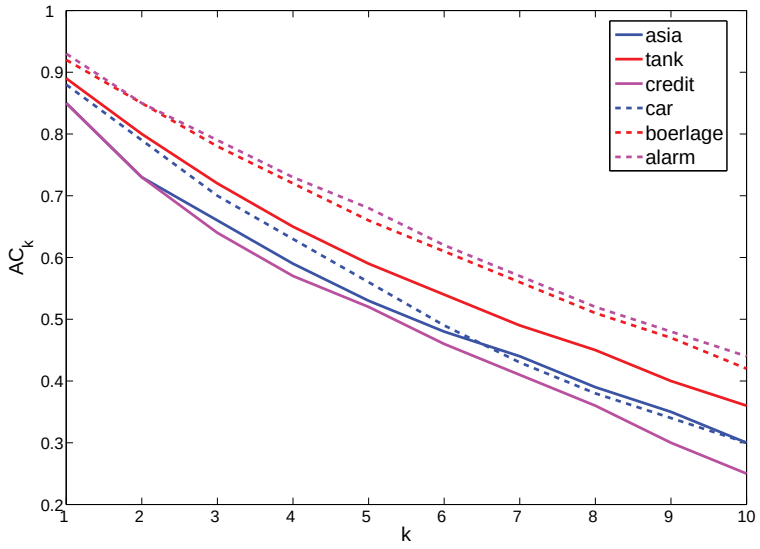


Figure 3.1: Variation of autocorrelation AC_k for all datasets under change of k

In addition, FDC was also computed using SHD to the true structure as a distance metric. As expected and as shown in Table 3.3, the coefficients are generally close to -1, meaning that the fitness attributed to a solution by means of CH score is a good approximation. *Boerlage* and *alarm*, the two largest problems of the test suite exhibit correlation coefficients lower than -0.90. The two smaller problems, *car* and *credit* have higher coefficients, but the correlation remains important (< -0.80). Yet, on the two smaller benchmarks, the correlation is less pronounced. This is particularly apparent for *tank* which has a fitness-SHD correlation coefficient of -0.25 and a large standard deviation (0.59). More generally, this shows that on the specific *tank* problem, the CH score does not reflect the actual quality of the structure. However, the correlation coefficient obtained using SHD has a different meaning than the r coefficient obtained using KTD. Because, the search operates in the space of orderings, SHD is not used to assess the relation between fitness and solution distances and KTD is preferred for this purpose.

asia	*tank*	*credit*	*car*	*boerlage*	*alarm*
-0.24	0.03	-0.34	-0.24	-0.48	-0.53
(0.30)	(0.39)	(0.27)	(0.33)	(0.26)	(0.23)

Table 3.1: Correlation coefficients r obtained on the different benchmark problems using KTD as distance metric, with standard deviations in brackets

3.1.2 Plateaux Analysis

In K2GA, solutions are represented by orderings of nodes. Each ordering is processed by the K2 algorithm which determines a set of edges. This means that different orderings can lead to the same structure and hence be set with similar fitness values. Because of this specific property of the K2 ordering representation, it is important to study the landscape with respect to the number of plateaux that it introduces. We define a plateau as a succession of operations that do not bring any change in terms of fitness value of an ordering. In this section, we use the results from the previously described random walk and count each step as an operation. Hence, a plateau is defined as a succession of adjacent swap mutations that generate orderings of similar fitness. From the random walk results, landscapes are evaluated on the number of plateaux they exhibit, the average size of a plateau and the size of the largest plateau. Note that plateau sizes are given in number of successive swaps and that plateaux of size 0 are not counted in the statistics. All measurements are averaged over the 30 repeat runs and presented in Table 3.4. In general and regardless of the benchmark, it can be said that the landscape associated with the optimization problem of BN structure learning exhibits many plateaux. The mean number of plateaux ranges from 15 (*car*) up to 23 (*boerlage*), which is an important number when considering that only 100 steps were allowed during the random walk. The average size of the largest plateau is also large ranging approximately between 8 and 19 steps. To put this number back into the context, this means that on *car*, up to 19 alterations can be performed to the ordering without affecting the fitness of the ordering. Of course, because of the experimental

	k									
	1	2	3	4	5	6	7	8	9	10
asia	0.85	0.73	0.66	0.59	0.53	0.48	0.44	0.39	0.35	0.30
	(0.11)	(0.18)	(0.20)	(0.21)	(0.23)	(0.24)	(0.25)	(0.25)	(0.26)	(0.27)
tank	0.89	0.80	0.72	0.65	0.59	0.54	0.49	0.45	0.40	0.36
	(0.07)	(0.11)	(0.14)	(0.19)	(0.21)	(0.23)	(0.24)	(0.25)	(0.27)	(0.26)
credit	0.85	0.73	0.64	0.57	0.52	0.46	0.41	0.36	0.30	0.25
	(0.09)	(0.17)	(0.18)	(0.19)	(0.19)	(0.19)	(0.19)	(0.20)	(0.21)	(0.22)
car	0.88	0.79	0.70	0.63	0.56	0.49	0.43	0.38	0.34	0.30
	(0.07)	(0.11)	(0.13)	(0.16)	(0.18)	(0.19)	(0.21)	(0.21)	(0.21)	(0.22)
boerlage	0.92	0.85	0.78	0.72	0.66	0.61	0.56	0.51	0.47	0.42
	(0.03)	(0.06)	(0.09)	(0.10)	(0.12)	(0.13)	(0.15)	(0.16)	(0.18)	(0.19)
alarm	0.93	0.85	0.79	0.73	0.68	0.62	0.57	0.52	0.48	0.44
	(0.04)	(0.07)	(0.10)	(0.12)	(0.13)	(0.15)	(0.16)	(0.17)	(0.18)	(0.19)

Table 3.2: Autocorrelation AU_k obtained on the different benchmark problems for different values of k, with standard deviations in brackets

asia	*tank*	*credit*	*car*	*boerlage*	*alarm*
-0.72	-0.25	-0.84	-0.83	-0.92	-0.94
(0.18)	(0.59)	(0.15)	(0.09)	(0.07)	(0.05)

Table 3.3: Correlation coefficients obtained on the different benchmark problems using SHD as distance metric, with standard deviations in brackets

design, the same ordering can be represented more than once in a plateau, although this happening is less likely as the number of nodes in an ordering increases. Consequently and despite being substantial (16), the largest plateau found on *alarm* is very unlikely to exhibit twice the same ordering. On another note, the mean size of a plateau is comprised between 2.25 and 5.48 depending on the problem. Some correlations can be observed from all results. Problems that exhibit lower number of plateaux, that is *car* and *alarm* also exhibits the highest value for largest plateau size and mean plateau size. This can be explained by the fact that when large plateaux are observed through a random walk, there are less room for extra plateaux. Similarly, *asia*, *tank* and *boerlage* all have relative low mean plateau sizes, but have many plateaux. It is possible that noise could lead to a large number of plateaux, that is the presence of single solutions that can lead the random walk away from a given plateau for a few steps only. However, since this is not essential to the understanding of the fitness landscape associated with BN structure learning, this was not investigated further. To summarize and according to the chosen benchmarks, BN structure learning using K2 is an optimization problem that presents plateaued landscapes. This property suggests that algorithms may experience unexpected convergence since different solutions can have similar fitnesses. This finding represents another challenge for search and score algorithms on top of the rugged landscape described by means of autocorrelation.

	asia	*tank*	*credit*	*car*	*boer-lage*	*alarm*
Num. of pl.	21.40 (3.28)	21.63 (4.00)	18.77 (3.42)	15.50 (3.85)	23.27 (2.56)	17.80 (3.80)
Largest pl.	8.43 (4.83)	11.87 (5.63)	14.20 (5.59)	18.73 (7.37)	8.07 (2.32)	16.00 (7.81)
Mean pl.	2.25 (0.69)	2.95 (1.17)	4.06 (1.16)	5.48 (1.86)	2.56 (0.46)	4.56 (1.71)

Table 3.4: Plateau characteristics of the different benchmark problems, with standard deviations in brackets

3.2 Influence of fitness properties on a population-based algorithm

In this section, we propose to study the presence of local optima in the fitness landscape, which is a known property of BN structure learning (Nielsen et al., 2003). In order to do so, K2GA is run on some tests problems and results are gathered regarding the quality of the solutions produced. In addition, following findings from the plateau analysis, the convergence is investigated.

K2GA is run with different settings on the benchmark problems. Settings are decided in accordance with literature and set in such ways that they help observing their impact on K2GA. A limit of 1000 generations is set. Fitness and SHD of the best solution at each generation is recorded and plotted. Tournament selection and cycle crossover are used to obtain these results. Since the tournament size is known to affect the selection pressure (Back, 1994), different tournament sizes were assessed. We recall that the tournament selection works by pre-selecting from the population a number of solutions defined by the tournament size. These solutions entered a tournament from which they are compared to each other relative to their fitness values. The solution with the highest fitness (in the case of a maximization problem) is selected and added to the breeding pool. For experimental purposes, a tournament size of 1 was used in specific

runs. This is similar to random selection. In addition to the tournament size, it has also been proved that the population size considered in isolation affects the convergence of a GA (Goldberg et al., 1991). Thus, several population sizes were considered, ranging from 10 to 100. Since BN structure learning is a problem computationally expensive to evaluate, populations sizes larger than 100 were not considered. Although it may increase algorithm performance, using large populations also means that less generations can be performed within the experimental bounds. In this study, the focus is set on the ability of search and score approaches to reach good solutions by means of evolution rather than by increasing the chances to get good individuals from the initial population. This choice of population sizes is also consistent with choices made in previous work on K2GA (Larrañaga, Kuijpers, Murga and Yurramendi, 1996; Kabli et al., 2007; Fournier et al., 2010). Figures 3.2, B.1, B.2, B.3, B.4 and B.5 show how K2GA behaves when varying the population and tournament sizes on all benchmark problems. Results are averaged over 10 runs and the size of the data used to learnt the BN structures is set to 3000 records.

When small population sizes are used such as 10 or 20 individuals, K2GA exhibits a very fast convergence and only a few solutions are actually evaluated. For example, when the population size is set to 10, less than 50 distinct fitness evaluations are performed. This is because a solution is only evaluated if it differs from both parents. Thus, in our implementation, such a low number of fitness evaluations is another sign of early convergence. As a direct consequence, K2GA does not have enough time to generate good solutions and it converges while the fitness of the best individual in its population is low. On *car*, a population size less than 50 seems to be insufficient to reach good solutions. Beyond this size, K2GA appears to always produce a final best solution of comparable fitness. These observations can be confirmed by analyzing the evolution of the KTD in the population over time. With small populations, the gradient of the curve is very high and the drop in KTD is sudden. Although this may be seen as a good behavior, the search does not evaluate enough solutions and does not reach solutions of good fitness, which reflects a premature convergence problem. This is smoothed by

increasing the population size.

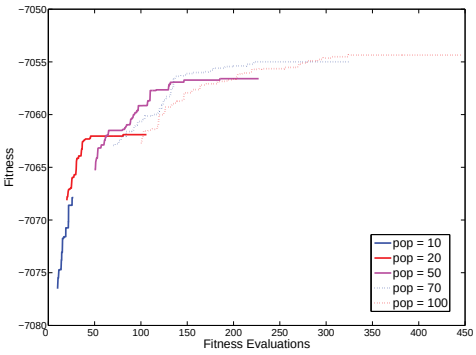
Similar observations can be seen with respect to the tournament size. Since the evolutions of fitness and KTD are correlated with population size, the impact of tournament size was studied under two population sizes, chosen such that they allow enough time for K2GA to converge to relatively good solutions, that is population sizes of 50 and 100. The largest considered tournament sizes that favor the best solutions for selection lead to faster convergence and similarly to the case of population size, lead to final solutions of lower quality, as it is the case with a tournament size of 4 for example. On the other hand, using a tournament size of 1 leads to the slowest convergence and the best final solutions. In this case, a tournament of size 1 is equivalent to randomly selecting a solution from the population, hence introducing the lowest selection pressure possible with tournament selection.

Overall, regardless of the settings on the population and tournament sizes, convergence happens very quickly. This is also true when setting the smallest tournament size along with large population sizes, where diversity between solutions decreases at a fast rate.

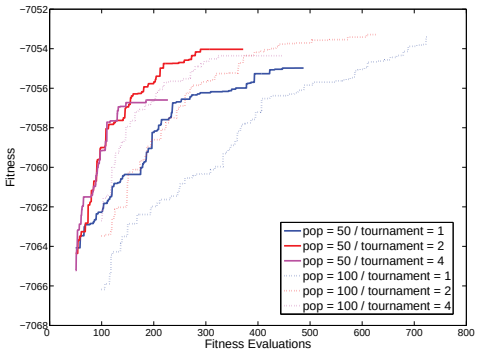
Since it has been shown that the different implementations of K2GA converge at a fast rate, it is important to study the quality of the solutions that each of them reach. One can claim that a slow convergence speed may not be a drawback if an algorithm reaches the global optimum over repeated run. In figures 3.3, C.1, C.2, C.3, C.4 and C.5, orderings that are obtained at the end of the 10 runs are compared with each other for four implementations of K2GA, namely, with population size of 50 or 100 individuals and tournament size of 1 or 4. In order to do so, orderings are considered to have similar quality if their fitnesses are equal, that is the BN structures that can be derived from them are the same. The first observation that can be made concerns the number of distinct fitnesses obtained over 10 runs. For example, on the *car* benchmark, K2GA has reached between 7 and 8 different solutions at the end of the runs, illustrating the presence of many local optima, but also of many instances of global optima. The presence of many instances of global optima is made possible by the fact that several

distinct variable orderings can lead to the same BN structure following the K2 process and hence have the same fitness value. In addition, by focusing only on the best ordering found over the 10 runs, it is noticeable that its frequency of occurrence is very low (between 10% and 20%) with tournament size of 1. When the tournament size is increased to 4, it seems that some of the orderings occur more often at the end of the runs, up to 30% of the time. However, this does not always represent the best ordering over all. For example, with a population of 50 individuals and a tournament size of 4, the final ordering that occurs the most (30% of the time) is only the fourth one in terms of fitness value over all ranked orderings obtained after 10 runs. Note that in this particular experimental setting, the best ordering was only found once over all runs. The last two observations support the hypothesis that the problem of learning BN structure exhibits many local optima. The number of local optima also appears correlated with the dimension of the problem. For instance, on *asia*, the most frequent final best solution produced by K2GA with tournament size of 1 and a population of 100 individuals occurred 7 times out of the 10 runs. Using the same settings, the final best solution only occurred 5 times on *credit* and twice on *car*. Finally on the two largest datasets, *boerlage* and *alarm*, all runs reached distinct solutions. Note that K2GA produced at maximum 2 similar solutions on the particular problem of *tank*.

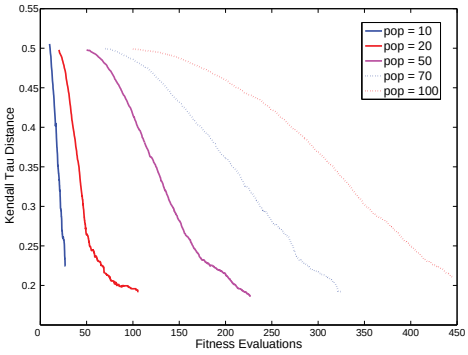
The set of experiments on K2GA helps to identify a few issues. First of all, it is obvious that despite trying different settings regarding some parameters known to affect diversity, K2GA exhibits early convergence. This is likely to be correlated with a convergence to solutions of low fitness, caused by an uneffective exploration of the search space. Evaluation of the solutions obtained by K2GA at the end of multiple runs shows that it converges to many different solutions. These can be defined as local optima. As the dimension of the problem increases, their number also increases and the reliability of the approach can be questioned, since two successive runs will lead to different solutions of different quality. In the next sections, a particular focus is put on managing diversity in the population in order to avoid early convergence. Several approaches are considered and compared to K2GA.



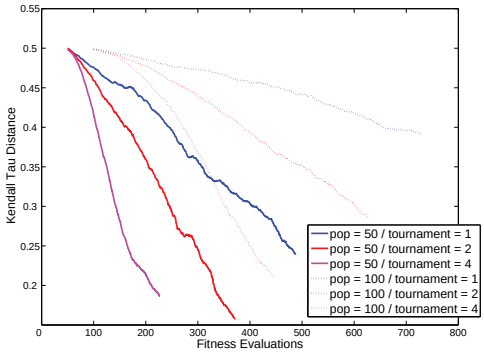
(a) Effect of population size on fitness



(b) Effect of tournament size on fitness

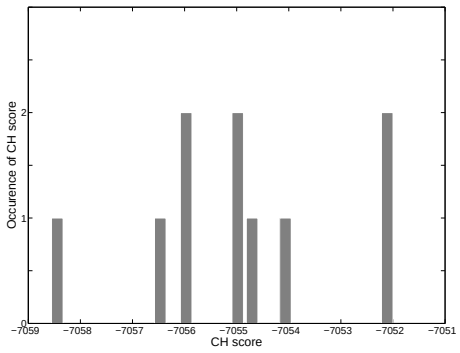


(c) Effect of population size on KTD

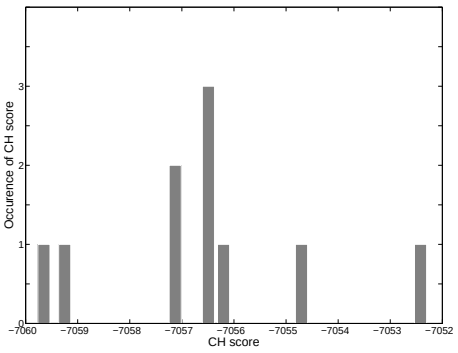


(d) Effect of tournament size on KTD

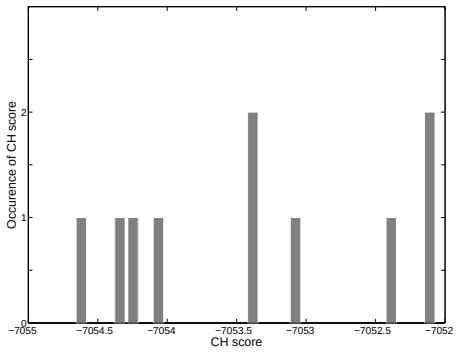
Figure 3.2: Convergence speed of K2GA on *car*



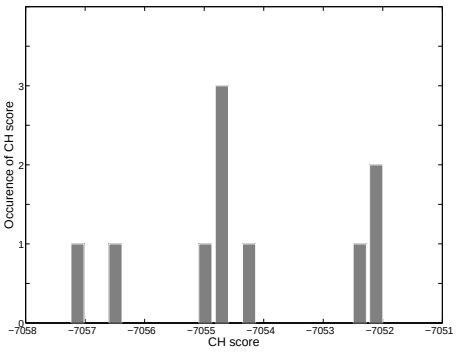
(a) Population size = 50, tournament size = 1



(b) Population size = 50, tournament size = 4



(c) Population size = 100, tournament size = 1



(d) Population size = 100, tournament size = 4

Figure 3.3: Frequency of best orderings obtained over 10 runs of K2GA on *car*

3.3 Summary

- Fitness Distance Correlation measures how correlated a fitness function is to the quality of solutions. Using KTD as a distance metric, FDC measures how different to an optimal ordering a solution is when its fitness decreases. FDC can be interpreted as a measure of optimization difficulty. In the case of BN structure learning, FDC using KTD was calculated on all benchmark problems. Results show that these benchmark problems can be considered as straightforward to solve for search and score methods.

- Although fitness and distance to optimum are correlated, the fitness landscape may present other challenges that can affect search and score algorithm performances. Autocorrelation measures the ruggedness of a landscape. It is found that the landscape is smoothed when considering solely neighbor solutions, that is distant by only a few step in the random walk. Yet, increasing the number of steps leads to a decrease in autocorrelation showing that the landscapes associated to all benchmark problems are rugged.

- Correlation between CH score and actual BN structural quality is also assessed. FDC was computed using SHD as a distance measure. It shows an overall correlation between CH score and SHD. However, this is not true for *tank*, showing that on this particular problem, there is a discrepancy between quality of node ordering and quality of structure.

- Since many orderings can produce the same BN structure in K2, plateaux can be observed in the landscape. Measurements following a random walk show that all landscapes exhibit many plateaux, including some that are large. This is prone to provoke early convergence in search and score algorithms.

- Such landscape properties are considered when running K2GA, an established algorithm for BN structure learning. Early convergence is found correlated to

population and tournament sizes. When small population or high tournament sizes are used, K2GA converges very fast but to sub-optimal solutions. This can be explained by the selective pressure being too high in such settings.

- Overall, the convergence of K2GA can be slowed down through careful setting of tournament and population sizes, but early convergence remains that can affect the quality of the final results.

- In addition to presenting early convergence, K2GA shows that the quality of the solutions that it finds is not optimal. It is found to converge to solutions of different quality over repeated runs. This helps confirming that the fitness landscape exhibits many local optima.

Chapter 4

Combining Local Optima by Means of Island Models

The previous chapter points out some difficulties related to learning BNs from data. One of the main challenges that were highlighted was the loss of diversity that can be seen when using population-based algorithms, leading to premature convergence of the search. In this section, we focus on this issue and investigate how convergence can be slowed down in K2GA by means of IM. In order to do so, the effect of the crossover operation on local optima is first studied. This motivates the use of IM within the K2GA approach which is implemented, tested on the benchmark suite and compared with its serial counterpart. The concept is put to the test in the context of distributed data, a common situation on which the inherent mechanism of IM is naturally adapted.

4.1 Effect of Crossover on Local Optima and Island Models

It has been shown in Chapter 3 that K2GA suffers from early convergence and does converge but to solutions of different fitnesses. Hence, it was considered that K2GA

population size / tournament size							
50/1	50/4	100/1	100/4	50/1	50/4	100/1	100/4
Asia				**Tank**			
4%	7%	0%	4%	13%	24%	4%	20%
Credit				**Car**			
29%	24%	2%	4%	22%	20%	9%	20%
Boerlage				**Alarm**			
31%	58%	36%	22%	33%	40%	40%	40%

Table 4.1: Frequency of beneficial crossovers over all possible combinations of local optima

converges to many local optimal solutions. In this section, we investigate when and how often the crossover of local optima can benefit the search. In order to do so, the final solutions obtained in Section 3.2 following 10 runs on each benchmark with different settings are considered as local optima. There are many ways to approach the question of measuring the benefits of crossover operations. In this work, we consider that one of the objectives of crossover is to produce a solution that has a better fitness than at least one of its parents, while being different from both of them. The latter part of this statement is particularly true in the context of BN structure learning when aiming at maintaining a certain level of diversity in the population. It is important to state that the aforementioned definition of a beneficial crossover is only true in the present context, where the search can easily be trapped in local optima. Such rule is defined and applied on solutions generated following crossover between every possible pair of local optima. Since 10 runs were completed for each problem, a total of 45 crossover operations were performed (there are $\frac{n(n-1)}{2}$ pairs of optima for n distinct optima, that is $\frac{10*9}{2}$ in the present case) for each configuration of population and tournament sizes. The results presented in Table 4.1 are averaged over the 45 operations. We recall that some of these node orderings display the same CH score although the node orderings that they represent are different. The cycle crossover operator used within K2GA was applied here. Since, it produces two distinct offspring solutions as explained in Figure 2.9, each of them was compared to both parents to decide whether or not crossover was beneficial.

Results are very dependent to the benchmark problem. For example, on *asia*, crossover of the local optima brings improvement to the search on maximum 7% of the occasions it is applied. In some settings such as with population and tournament sizes respectively set to 100 and 1, no advantage is seen in using crossover. However, on the larger problems, crossover produces improved solutions in up to 58% of the time, as seen on *boerlage*. These results need to be analyzed while considering the number of different optima that were reached by K2GA. On *asia*, despite presenting several final solutions, some local optima were of similar quality. Hence, using crossover on these is likely to produce solutions of similar fitness, which means that the crossover is not beneficial. Such analysis also gives an idea on how good the solutions previously obtained are. Since a simple crossover can improve them in 40% of the cases on *alarm*, their absolute quality can be questioned.

Experiments also display the importance of the settings regarding population and tournament selection. It has been previously observed that using a population size of 100 and a tournament size of 1 helps with maintaining the diversity. Running K2GA with similar settings on the four smallest problems also suggests that the quality of the solutions is high, since applying crossover on the final solutions only leads to improved solutions on few occasions (9% at maximum). Similarly, the settings that show the highest correlation with early convergence, that is using populations of 50 individuals and tournament size of 4, display the best rates of improvement on all of the benchmark problems but one, *car*. On *boerlage* and *alarm*, the differences between the configurations of K2GA is less obvious and crossover operations exhibit benefits at rates greater than 20%.

The overall message that arises from these results lies in the fact that crossover, which is a simple operation, can offer some benefits and produce improved solutions by bringing together local optima. Naturally, these findings need to be analyzed with respect to the problems. Yet, it seems that crossover of local optima represents a promising way to enhance the quality of the solutions found by K2GA on larger problems. Since

new solutions can be produced from crossing over individuals across different GA populations that have converged, it seems natural to further apply the idea and consider the use of IM.

In IM (Cantu-Paz, 1995), several evolutionary searches are run in parallel subpopulations, or *islands* and exchange solutions at given intervals. This has two advantages. First, the algorithm can easily be split between the searches and thus be set to run on different processors. Second, the independent searches combined with the exchange of solutions makes IM-based algorithms behave differently than their serial counterparts. IM have been widely applied in order to cope with limitations some serial EA implementations may have on specific problems, such as convergence to local optima, certainly the most frequent issue faced by EAs. It is known that IM can help with this matter by using the results from many searches at different islands. If a local optimum is reached at a specific island, this is likely to be fixed by integrating a solution found by another search at a different island. In (Skolicki and De Jong, 2004), the idea is extended in order to allow multi-representation between islands. In other words, the search is performed according to different criteria at each island in order to minimize the likelihood of falling in local optima. Thus, this approach seems particularly appropriate for searching through multi-optimal spaces such as those associated with BN structure learning (Nielsen et al., 2003).

To our knowledge, although IM has been applied to reduce computation time, it has not been used to improve the performance of EAs for BN structure learning. However, the need for parallel BN structure learning and more generally for parallel PGMs has been expressed within the EDA community in recent years (Iclănzan and Dumitrescu, 2010; Santana, 2011). In some EDAs such as the Bayesian Optimization Algorithm (BOA)(Ocenasek and Schwarz, 2001) and the Estimation of Bayesian Network Algorithm (EBNA)(Mendiburu et al., 2005), BNs are used to model the distribution of solution genes in order to sample new individuals. In serial versions of BN-based EDAs, it is accounted that 50 to 99% of the total runtime is dedicated to model building (Mendiburu et al., 2005). Work has been done in (Ocenasek and Schwarz, 2001)

in reducing the time allocated to the construction of the BN model in BOA using parallel implementation. By decomposing and distributing the scoring function to a pool of workstations, the distributed BOA exhibited a short runtime, but also a loss of efficiency.

More generally, IM has been applied in (De la Ossa et al., 2004) to parallelize EDAs. Two approaches were assessed in which either individual solutions or underlying probabilistic models were exchanged during migration. Findings were two-fold, showing that IM implementations outperform the serial versions and that in addition to being possible, the exchange of probabilistic models from which new solutions can be sampled generally leads to better results than the exchange of individuals.

4.2 Island Model K2 Genetic Algorithm

In this section, IMK2GA, an IM implementation of K2GA is presented. Further, experiments regarding the use of IM with K2GA are run in two steps. First, it is investigated if IM can help K2GA in maintaining diversity. Hence comparisons between several configurations of IMK2GA are performed in an experimental environment that exhibit characteristics prone to lead to early convergence. More explicitly, this was done by interacting with the population and tournament sizes. In the second part of the experiment, head-to-head performance comparisons are run in an optimal environment for K2GA. This second step is essential to assess the performance of IMK2GA relative to a state-of-the-art method, K2GA, when tunned with near-optimal settings.

4.2.1 Implementation

The proposed implementation is based on the same ordering representation as the one used in K2GA. More precisely, IMK2GA is composed of many instances of K2GA running in parallel. The principle is to pause the search when migration of individuals should occur in order to exchange solutions between islands. Once the migration step

is completed, the search can be resumed. Many strategies can thus be considered which relates to the migration topology, the migration rate, the migration size and the migration policy. In this section, we present a proof of concept and aim at studying the possible use of IMK2GA. Therefore, we only consider one particular strategy that uses a best-worst migration policy on a ring topology (De la Ossa et al., 2004) with two solutions being exchanged at each migration. In a ring topology, islands are thought of as forming a circle in which each island sends solutions to its right neighbor while receiving solutions from its left one at migration. Algorithm 3 summarizes the different steps of IMK2GA. The number of islands, number of migrations and migration intervals are respectively set to a, b and $migIntervals$. At each island isl_i, the population pop_i is evolved to search the space of solutions. The search is split in evolution stages stg_j according to the number of generation gen_i already performed. At the end of each evolution stage but the last, a subset mig_i is created for migration with the λ best individuals from pop_i. Subsets at different islands are then exchanged following a ring topology and the λ worse individuals replaced in each pop_i. Note that this is also the case when migrants present a lower fitness than the worse individuals.

In order to assess many configurations of the algorithm and to help the understanding, we will refer to different configurations of IMK2GA as $IMK2GA_{ab}$ in the rest of the chapter, where a and b represent the number of islands and the number of migrations respectively [1]. The traditional version of K2GA corresponds to $IMK2GA_{10}$.

4.2.2 Island Models and Loss of Diversity in K2GA

In the first experiment, small population sizes were used for the different benchmark problems. These were respectively set to 100, 50, 50, 20 and 20 for *asia*, *tank*, *credit*, *car* and *boerlage*. In addition, selective pressure was kept high through using tournament selection of size 4. Note that this part of the experiment was not assessed on *alarm* due to time and computational cost constraints. IMK2GA is later evaluated

[1] Only one-digit values are considered throughout this thesis for the number of islands and the number of migrations.

Algorithm 3: $IMK2GA_{ab}$

```
for each island isl_i, i ∈ [0, a − 1] do
  Initialize population pop_i
  Evaluate pop_i using K2 algorithm
  Initialize number of generation gen_i = 0
  for each evolution stage stg_j, j ∈ [0, b] do
    repeat
      Select orderings in pop_i to reproduce
      Apply crossover operator
      Apply mutation operator
      gen_i + +
    until gen_i = migInterval ∗ (j + 1)
    if j ≠ b then
      Select n best orderings to create subpopulation mig_i
      if i ≠ 0 then
        Replace λ worse orderings in pop_i by mig_{i−1}
      else
        Replace λ worse orderings in pop_i by mig_{a−1}
      end if
    end if
  end for
end for
```

on *alarm*, with respect to its performance. In this set of experiments and on a given dataset, each island is set with the same population size. This allows us to observe the behavior of all algorithms when the inherent evolution is based on populations with comparable settings. In addition, this strategy helps understanding the mechanism at a single island as $IMK2GA_{10}$ can be considered as such in this case. Different approaches were considered for the IM strategies in terms of number of islands (2 or 4) and number of migrations (1 or 3). Algorithm behavior was studied on 1000 individual fitness evaluations (FEs) with all migrations occurring during this period, when at least two islands were used. The migration interval was therefore set manually for each $IMK2GA_{ab}$. Migration interval for $IMK2GA_{10}$ refers to the total number of generations that were run. These values are given in Table 4.2. A maximum of 10000 generations per migration interval was allowed for all algorithms to reach at least 1000 FEs. Algorithms which did not reach that number of FEs were considered as having converged at the end of the run. Finally, the samples from which BNs were learnt were

	restricting diversity				enhancing diversity
Dataset	$IMGA_{21}$	$IMGA_{23}$	$IMGA_{41}$	$IMGA_{43}$	$IMGA_{43}$
Asia	1000	500	300	150	100
Tank	5000	1500	600	250	100
Credit	4000	1500	600	300	100
Car	8000	3000	4000	1300	100
Boerlage	8000	3500	3500	1300	75
Alarm	n/a	n/a	n/a	n/a	75

Table 4.2: Migration intervals for $IMK2GA_{ab}$ (in generations)

composed of 5000 records.

Tables 4.3 and 4.4 summarize the mean characteristics of the best BN obtained by each algorithm on each dataset after 1000 FEs. Best values for fitness and C appear in bold. Values that are not significantly different from the best (p-value>0.005 after Bonferroni correction (Dunn, 1961)) have been marked with a * symbol. Overall, $IMK2GA_{43}$ produces the ordering with the highest fitness for all datasets but *tank*. Yet, in terms of C, the highest numbers are obtained by $IMK2GA_{10}$ on *asia*, $IMK2GA_{23}$ on *tank* and by $IMK2GA_{43}$ on *credit*, *car* and *boerlage*. Despite being significantly better in some comparisons, differences are not as important among IM-based methods than as between them and the traditional K2GA although this statement is not true for comparisons on *asia*, the smallest test problem.

In addition to the sole results, it is important to evaluate how the quality of the solutions changes over time. Figures D.1, D.2, D.3, D.4 and D.5 show the evolution of the best solution at each generation for all $IMK2GA_{ab}$ configurations and all benchmark problems on which they were run. Solution quality is represented by means of fitness, relevant and erroneous edges and SHD. In addition, this set of figures also displays the evolution of the KTD in the population.

These figures also allow us to investigate the importance of the number of islands. First of all, there is a difference in terms of level of diversity that is reached following migration. Migration always has a positive influence on the KTD. However, when 4 islands are used, the level of KTD is considerably higher than when only 2 islands are

used. This is particularly apparent on car and credit as seen in Figures D.3 and D.4. The number of islands also has an impact on the evolution of the fitness. Although fitness is generally improved at migration, a lack of evolution can be observed successive to migration when using 2 islands. This behavior can be seen on all benchmarks and is most likely linked to to the aforementioned level of diversity that is much lower in these configurations.

The introduction of IM also involves another parameter of high importance, the number of migrations. In terms of KTD, when 3 migrations are used, the drop in diversity is slowed down. This observation is clear on *credit* and is represented by a gradient much smaller in the KTD curve of Figure E.3c, once the first migration has been performed. Although the diversity in the population still decreases, it does so at a very small pace after migration. Figure E.3c also displays how the KTD of the population of $IMK2GA_{41}$ is maintained after migration. However, using simply one migration seems less efficient than using 3. The time at which the first migration occurs represents another parameter that should be studied since it would be beneficial to slow down the diversity loss at an earlier stage in the process. However, this was not supported in the implementation used in this work and thus is not investigated.

More general comments can be made on the behavior of all implementations using IM. As expected following the experiments on bringing together local optima in Section 4.1, the fitness is always improved following the first migration. Subsequent migrations do not show signs of decrease in fitness although they do not always lead to improvement. This is probably caused by the fact that solutions of high quality have been reached already, making improvements less frequent. This hypothesis is supported by the fact that inefficient migrations are mostly observed on smaller problems. On the other hand, some surprising observations can be made. For example, $IMK2GA_{23}$ shows signs of decrease in KTD when migrations are performed on the *boerlage* problem. Although this behavior is unique to this algorithm and problem, it remains counterintuitive and hard to explain. Experiments on *asia* also generate interesting cases such as the high SHD that is associated to $IMK2GA_{43}$ after the first migration. However, on this

particular and easy problem, the sole GA evolution ($IMK2GA_{10}$) is able to reach good solutions (low SHD and high number of relevant edges). IM may have a negative effect on such problem when evolving solutions that are already of high quality. We note that the same observation is valid for $IMK2GA_{21}$ and $IMK2GA_{23}$, but to a lesser extent. $IMK2GA_{43}$ is also the only implementation that presents degradation relative to the number of relevant edges of the best solution throughout the search. Of course, these latter points are to be considered moderately since learning *asia* is not a very challenging task. Finally, the two implementations using 4 islands do never converge on *tank* in the given time limit. Fitness is constantly increased and their populations' SHD maintained to a steady level after the first migration. Yet, this good behavior relative to a GA performance corresponds to a decrease of the BN structure quality represented by an increasing SHD and number of erroneous edges. This can be explained by recalling the lack of correlation between CH score and BN quality on the *tank* benchmark as seen in Section 3.1.

We investigate in more depth the evolution of the edge quality in the best structures over time, through studying distinctly correct, reversed, added and omitted edges. This is done for two configurations of $IMK2GA_{ab}$. Figure D.6 illustrates how these edges change on $IMK2GA_{21}$, a configuration in which it is easy to study the sole impact of a single migration between two islands. Figure D.7 focuses on $IMK2GA_{43}$, which is the implementation exhibiting the best results. One of the effects of migration is a decrease of reversed edges and an increase in correct edges. This is simply the redirection of an edge and can be seen on the figures for *tank* and *car*. However, migration sometimes leads to the redirection of edges that were correct. Although this is not an observation that one wants when seeking for a BN structure, the evolution that follows mutation results in the edge being redirected to its correct state. Such behavior is observed on *asia*, *credit* and *boerlage* and helps stating that from the results on $IMK2GA_{21}$, a single mutation may not always be beneficial to the structure but it does not lead to irrevocable negative alterations. $IMK2GA_{43}$ follows this trend. It shows that migrations on *boerlage* can actually benefit the search and redirect edges,

although not seen with $IMK2GA_{21}$. This difference relates once again to the diversity in the population at the time of migration. But, we also notice that some negative alterations on *asia* are not fixed by evolution. Finally, we point out the difficulty that exists in discerning the effect of migrations. The first migration always affects the search noticeably, but it is not always the case with the subsequent migrations. We propose that this is due to the state of the population at each individual island. With all populations more likely to have reached the same best individuals after a few migrations, it becomes more difficult to further improve the search. It may suggest that an important parameter to study is the time of the first migration as mentioned already.

4.2.3 Performance Evaluation of IMK2GA

Most of the behavior inherent to $IMK2GA_{ab}$ has been investigated through the previous experiments. It was noted that overall, using 4 islands and 3 migrations leads to the best results, both by helping in maintaining diversity in the population, but also by producing solutions of high fitness. Yet, these findings need to be handled with care since the environment in which the algorithms were runs were not well adapted to a serial GA, with small population sizes and high selection pressure. Thus, we focus now on studying purely the performance of one configuration of $IMK2GA_{ab}$ and the serial $K2GA$. Naturally, $IMK2GA_{43}$ is selected as part of the experiments for the promising results it previously displayed.

In order to reduce the bias towards IMK2GA, the population sizes were increased to 100 for all benchmark problems and the tournament size set to 1, introducing a low selective pressure, and thus slowing down the loss of diversity in the population. 10 runs were performed for the two methods and the 6 benchmark problems were considered. Empirical results are shown in Tables 4.5 and 4.6. Statistical significance was tested over 30 runs following the same approach as for the previous experiments in Section 4.2.2.

There are two main points to take out of these results. On the four smallest problems,

		Asia	Tank	Credit
$IMGA_{10}$	C	**6.07 (0.36)**	14.07 (2.02)*	8.6 (1.11)*
	R	1.93 (0.36)	5.83 (1.92)	3.4 (1.11)
	A	0.03 (0.18)	6.0 (1.51)	1.9 (0.75)
	O	0.0 (0.0)	0.1 (0.30)	0.0 (0.0)
	Fitness	-11234.9313 (0.56)*	-16678.0470 (7.22)	-52834.2905 (7.72)
$IMGA_{21}$	C	5.97 (0.18)*	15.0 (1.39)*	8.83 (1.07)*
	R	2.03 (0.18)	4.9 (1.42)	3.17 (1.07)
	A	0.03 (0.18)	5.6 (1.08)	1.47 (0.56)
	O	0.0 (0.0)	0.1 (0.30)	0.0 (0.0)
	Fitness	**-11234.7114 ($\mathbf{2.49.10^{-12}}$)**	-16672.4934 (2.79)	-52828.5094 (5.86)
$IMGA_{23}$	C	6.0 (0.0)*	**15.1 (1.64)**	8.86 (0.85)*
	R	2.0 (0.0)	4.9 (1.64)	3.13 (0.85)
	A	0.0 (0.0)	5.3 (0.97)	1.37 (0.48)
	O	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
	Fitness	**-11234.7114 ($\mathbf{1.99.10^{-12}}$)**	-16672.3447 (2.03)*	-52827.2595 (6.11)*
$IMGA_{41}$	C	**6.07 (0.25)**	14.6 (1.62)*	9.06 (0.73)*
	R	1.93 (0.25)	5.37 (1.58)	2.93 (0.73)
	A	0.0 (0.0)	5.6 (0.84)	1.27 (0.44)
	O	0.0 (0.0)	0.03 (0.18)	0.0 (0.0)
	Fitness	**-11234.7114 ($\mathbf{1.76.10^{-12}}$)**	**-16670.8904 (1.41)**	-52824.9420 (3.75)*
$IMGA_{43}$	C	5.83 (0.45)*	14.17 (1.81)*	**9.1 (0.79)**
	R	2.17 (0.45)	5.83 (1.81)	2.9 (0.79)
	A	0.17 (0.45)	6.03 (1.14)	1.1 (0.30)
	O	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
	Fitness	**-11234.7114 ($\mathbf{1.76.10^{-12}}$)**	-16670.9070 (1.14)	**-52823.9331 (3.22)**

Table 4.3: Characteristics of best BNs found by each algorithm after 1000 individual FEs on *asia*, *tank* and *credit*

		Car	Boerlage	Alarm
$IMGA_{10}$	C	10.47 (1.69)	19.2 (2.33)*	n/a
	R	3.63 (1.43)	8.9 (2.04)	
	A	7.7 (2.27)	5.23 (1.52)	
	O	2.9 (1.04)	7.90 (0.83)	
	Fitness	-11659.9074 (4.89)	-51301.7066 (11.06)	
$IMGA_{21}$	C	11.63 (1.17)*	19.13 (2.32)*	n/a
	R	2.9 (1.35)	9.23 (2.03)	
	A	7.03 (1.62)	4.63 (1.25)	
	O	2.47 (0.76)	7.63 (0.75)	
	Fitness	-11654.2428 (3.32)*	-51294.3238 (9.36)	
$IMGA_{23}$	C	11.17 (1.57)	19.17 (2.58)*	n/a
	R	3.03 (1.72)	9.17 (2.18)	
	A	7.47 (1.50)	4.67 (1.94)	
	O	2.8 (0.75)	7.67 (0.70)	
	Fitness	-11654.8275 (3.88)*	-51289.7254 (9.92)*	
$IMGA_{41}$	C	12.0 (0.93)*	19.43 (2.59)*	n/a
	R	2.57 (0.99)	9.03 (2.24)	
	A	7.17 (1.13)	4.2 (1.28)	
	O	1.13 (2.43)	7.53 (0.76)	
	Fitness	-11653.0616 (2.06)*	-51285.2221 (7.96)*	
$IMGA_{43}$	C	**12.2 (0.98)**	**20.13 (2.20)**	n/a
	R	2.23 (1.02)	8.37 (2.09)	
	A	6.7 (1.32)	3.87 (1.45)	
	O	2.57 (0.62)	7.5 (0.67)	
	Fitness	**-11652.4001 (2.46)**	**-51285.0276 (9.88)**	

Table 4.4: Characteristics of best BNs found by each algorithm after 1000 individual FEs on *car*, *boerlage* and *alarm*

$IMK2GA_{43}$ exhibits a better fitness than the serial $IMK2GA_{10}$. This also corresponds to better values of C and SHD. The use of IM supports and enhances the search on these benchmarks. However, as the dimension of the problem increases, the performance of $IMK2GA_{43}$ becomes worst than the one of $IMK2GA_{10}$. On *boerlage* and *alarm*, the serial GA significantly outperforms the IM implementation on all markers, relative to both fitness and solution quality. Table 4.7 gives an idea of the state of the search at the end of the runs for both methods by presenting the population KTD after 1000 fitness evaluations. It appears that $IMK2GA_{43}$ has a high diversity, while the one of $IMK2GA_{10}$ is much closer to 0, reducing the chances to produce further improvements.

In order to understand these results, it is important to look at the state of each method when the maximum number of fitness evaluations is reached and measurements taken. Figure 4.1 shows the evolution of the best fitness over time throughout the search. It is interesting to note that the shape of the curves is very different between the two methods. While $IMK2GA_{10}$ confirms the trend overseen in Chapter 3 by displaying a fast improvement in fitness, followed by stagnation, $IMK2GA_{43}$ shows a gentle improvement over time and does not exhibit any sign of convergence. On *boerlage* and *alarm*, where $IMK2GA_{10}$ outperforms the $IMK2GA_{43}$, the latter has not converged when the experiment is paused. This suggests that there is room for further performance improvement in $IMK2GA_{43}$. Similarly, at the end of the runs on *tank*, *credit* and *car*, $IMK2GA_{43}$ is stopped while improving its best solution. Yet, it outperforms $IMK2GA_{10}$ on these three benchmarks.

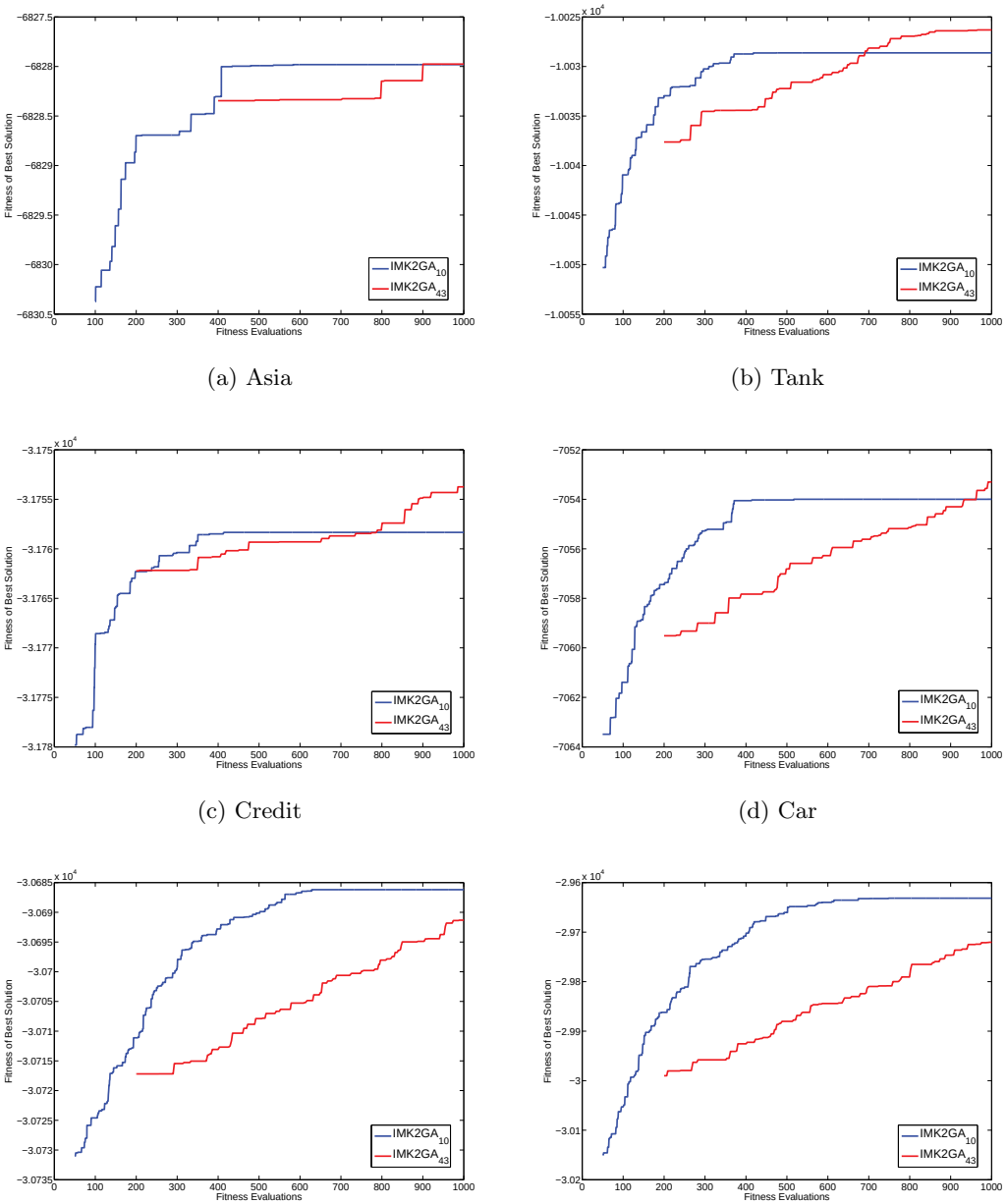
These findings highlight the difficulty in correctly setting the tradeoff between selective pressure and quality of final solutions. Of course, it is important to consider the applications and how many fitness evaluations can be performed. With time, the use of several islands and migration is likely to lead to better solutions at the price to a slow improvement in solution quality. On problems that cannot allow a long time for evolution, high selection pressure can be beneficial and serial GA implementations may be better adapted.

		Asia	**Tank**	**Credit**
$IMGA_{10}$	C	**6.73 (0.44)**	14.33 (2.09)	8.67 (1.14)
	R	1.27 (0.44)*	5.57 (1.96)	3.33 (1.14)
	A	0.0 (0.0)	5.33 (0.94)	1.40 (0.55)
	O	0.0 (0.0)	0.10 (0.30)	0.0 (0.0)
	SHD	**1.27 (0.44)**	11.00 (2.92)	4.73 (1.63)
	Rel.	**8.00 (0.0)**	19.90 (0.30)*	**12.00 (0.0)**
	Err.	**0.0 (0.0)**	5.43 (1.05)*	1.40 (0.55)
	Fitness	**-6827.9744 (0.02)**	-10027.5908 (2.27)	-31757.4589 (5.06)
$IMGA_{43}$	C	6.63 (0.48)*	**15.33 (1.62)**	**9.17 (0.52)**
	R	1.37 (0.48)	4.60 (1.43)	2.83 (0.52)
	A	0.0 (0.0)	5.03 (0.55)	1.03 (0.18)
	O	0.0 (0.0)	0.07 (0.25)	0.0 (0.0)
	SHD	1.37 (0.48)*	**9.70 (1.99)**	**3.87 (0.62)**
	Rel.	**8.00 (0.0)**	**19.93 (0.25)**	**12.00 (0.0)**
	Err.	**0.0 (0.0)**	**5.10 (0.70)**	**1.03 (0.18)**
	Fitness	-6827.9796 (0.03)*	**-10026.5528 (1.06)**	**-31753.7064 (1.74)**

Table 4.5: Characteristics of best BNs found by each algorithm after 1000 individual FEs on *asia*, *tank* and *credit* with parameters enhancing diversity.

		Car	**Boerlage**	**Alarm**
$IMGA_{10}$	C	12.97 (0.75)*	**20.50 (1.80)**	**26.53 (2.26)**
	R	1.77 (0.72)	7.60 (1.62)	15.33 (2.33)
	A	5.97 (0.84)	3.30 (1.44)	18.50 (2.09)
	O	2.27 (0.51)	7.90 (0.54)	4.13 (0.34)
	SHD	**10.00 (1.51)**	**18.80 (2.54)**	**37.97 (3.46)**
	Rel.	**14.73 (0.51)**	**28.10 (0.54)**	**41.87 (0.34)**
	Err.	**8.23 (1.27)**	**11.20 (1.72)**	**22.63 (2.14)**
	Fitness	-7054.3047 (1.75)*	**-30687.8064 (5.73)**	**-29647.0882 (34.39)**
$IMGA_{43}$	C	**13.03 (0.75)**	19.87 (1.65)*	25.10 (2.29)
	R	1.23 (0.50)	8.07 (1.57)	16.77 (2.22)
	A	6.30 (1.32)	4.10 (1.72)	20.97 (1.97)
	O	2.73 (0.73)	8.07 (0.44)	4.13 (0.43)
	SHD	10.27 (2.00)*	20.23 (2.14)	41.87 (3.33)
	Rel.	14.27 (0.73)	27.93 (0.44)*	41.87 (0.43)*
	Err.	9.03 (1.99)*	12.17 (1.77)	25.10 (2.07)
	Fitness	**-7053.7041 (1.14)**	-30692.8968 (4.83)	-29718.4400 (30.02)

Table 4.6: Characteristics of best BNs found by each algorithm after 1000 individual FEs on *car*, *boerlage* and *alarm* with parameters enhancing diversity



(a) Asia

(b) Tank

(c) Credit

(d) Car

(e) Boerlage

(f) Alarm

Figure 4.1: Evolution of the fitness of the best solution in $IMK2GA_{10}$ and $IMK2GA_{43}$ on all benchmark problems

	Asia	**Tank**	**Credit**	**Car**	**Boerlage**	**Alarm**
$IMK2GA_{10}$	0.25	0.11	0.17	0.19	0.12	0.08
	(0.05)	(0.07)	(0.06)	(0.08)	(0.08)	(0.03)
$IMK2GA_{43}$	0.46	0.40	0.39	0.41	0.43	0.43
	(0.01)	(0.01)	(0.02)	(0.01)	(0.02)	(0.01)

Table 4.7: Population KTD after 1000 fitness evaluations in $IMK2GA_{10}$ and $IMK2GA_{43}$ for each benchmark problem

4.3 Learning Bayesian Networks from Distributed Data

Distributed data mining (DDM) appeared as a new field of research when issues on classical centralized data mining were raised. Such issues encompass the long response time needed to gather data to a central site, the lack of proper use of distributed systems that could potentially improve the performance of data mining, or at least facilitate its completion. Other issues relate to the cost of communication involved with transferring data and finally the need to preserve privacy (Park et al., 2002).

From the literature, there is a distinction between two types of DDM, namely DDM which deals with homogeneous data, that is where the set of variables is identical across distributed data sites and DDM that handles heterogeneous data where variables are different between sites. Naturally, from this distinction, arise different approaches.

In this section, the use of IM is investigated in the context of homogeneously distributed data. This study is motivated by the solution migration process inherent to IM that could offer potential when performing data modelling tasks from distributed data. IM allows models to be learnt locally, that is based on the data at a single site, before exchanging these models through migration. Each island gets information about the data present at distant sites through the model it receives, but no data is transfered at any moment. The advantages of the approach are two-fold. First, it is a way to allow data modelling from distributed data, while preserving privacy, that is without exchanging or merging any data. Second, using IM could also represent a way to divide the data in order to mine separately subsets of the data, with different characteristics,

or smaller size, which can be beneficial in terms of computational power needed. We recall that the time associated with learning a BN from data is related to the number of instances in the data, among other elements. The use of IM is only treated here with the objective to allow privacy-preserving BN structure learning from distributed data, thus the issue of computational cost is not investigated.

Although several studies have addressed the problem of parallelization of BN structure learning (Lam and Segre, 1997; Ocenasek and Schwarz, 2001; Mendiburu et al., 2005; Santana, 2011), there are fewer of them that treat the issue of handling distributed data. In (Chen et al., 2004), local BNs are constructed at each site before being recombined to produce a global model. Despite retrieving similar structures as centralized methods, such approach presents flaws with respect to data privacy, as some selected instances are shared to compute the final parameters. Focus on privacy preservation is higher in (Yang and Wright, 2006), where a method is presented that computes CH score and the BN parameters from distant sites without sharing data. Other approaches including cryptography or noise addition have also been proposed for different kinds of data mining techniques but no method has yet been implemented which focuses on the exchange of BN structures (Yang and Wright, 2006).

4.3.1 Generating Distributed Data

Although generating distributed data could be straightforward, time was spent designing a strategy to produce different sets of data, corresponding to different contexts. Distributed real-world data may exhibit some differences between sites. For example, these can be the effects of differences in demographics among people represented by the data at distant locations. More generally, the data is likely to show dissimilarities in terms of its distribution across sites. In order to model these differences, data was generated with different distributions. Sampling simply the original BN as was done to generate the data used for the previous sets of experiments does not help introducing changes in the distribution. Hence, the parameters from the original BN were modified

to different extent to produce several altered BNs that can be sampled in turn. The process to alter the CPT entries of a given BN follows (Wu et al., 2012) and is detailed in Algorithm 4. It takes into consideration two parameters that are the alteration rate τ and the alteration power α. The process goes through all CPT rows r_{ij} for each node X_i and decide whether to bring some changes to the row or not. This decision is based on the value of τ, ranging from 0, where no changes are performed to 1, where each row is modified. The second parameter, α manages the degree of alteration that is brought to the row. When a CPT row is altered, each of its elements P_{ijk} is replaced by a value drawn in the range $[P_{ijk} - \alpha * P_{ijk}; P_{ijk} + \alpha * P_{ijk}]$. In this way, large values of α can result in new values that are very different from the original ones. Note that small alterations can also happen when α is large. Once all P_{ijk} of a given row have been changed, their values are normalized in order for them to sum up to 1.

Algorithm 4: Altering parameters in a given BN, at rate τ and power α

```
for each node X_i of BN do
  for each row r_ij of X_i's CPT do
    sum = 0
    Generate at random rand_1 ∈ [0; 1]
    if rand_1 ≤ τ then
      for each parameterP_ijk of r_ij do
        lowerbound = P_ijk − α ∗ P_ijk
        upperbound = P_ijk + α ∗ P_ijk
        Generate at random rand_2 ∈ [lowerbound; upperbound]
        P_ijk = rand_2
        sum = sum + P_ijk
      end for
      for each parameterP_ijk of r_ij do
        P_ijk = P_ijk/sum
      end for
    end if
  end for
end for
```

In order to evaluate IMK2GA on distributed data, three data configurations are considered. These essentially differ in the level of alteration brought to the original data distribution. The first set of BNs is obtained by using an alteration power α of 0.1. This represents very slight changes for the CPTs. For the two other sets of BNs, α was

respectively set to 0.5 and 1. These values were defined arbitrarily in order to give a broad overview of the performance of IMK2GA on different kinds of data whether it underwent very small ($\alpha = 0.1$) or important ($\alpha = 1$) changes. For all configurations, τ was set to 1, implying that all parameters from the BNs were altered.

For experiment purposes, 4 BNs were derived from each configuration. Thus, each of these BNs has a distinct distribution, although their distance to the original distribution is of the same order (similar α used for alteration). From each of the BNs, a dataset of 5000 instances was sampled that can be used at a single island.

4.3.2 Performance of IMK2GA on Distributed Data

The main objective of this set of experiments is to investigate whether IMK2GA can allow BN modeling from different datasets. Thus, it is important to study the quality of the learnt BNs and to compare them to the original structures. Quality of structures should be compared between the contexts of centralized data, in which all individual datasets are merged together and the one of distributed data, where the data is different at each island. Two algorithms are put to the test on each benchmark BN, and under the three configurations previously described. The two algorithms are K2GA to model the centralized data and $IMK2GA_{43}$ to handle the distributed data. Using 4 islands in IMK2GA allows modeling of 4 distinct datasets, set at each island.

30 runs were performed for each algorithm on all experimental configurations. It was decided to use a maximum of 3000 instances for modeling. Hence, in order to create a centralized dataset, all four 5000-instance datasets were brought together to create a 20,000-instance dataset. To avoid bias due to the data sampling process, the data underwent a preparation step in which it was randomized, leading to the use of different datasets at each run. Once randomized, the data can be sampled, by selecting 3000 records. Since, the maximum number of instances allowed was set to 3000 and 4 islands were defined, only 750 instances could be used at each island. Thus, the same randomization process was performed prior to start each run of IMK2GA. Each

algorithm was allowed 5000 generations to evolve solutions.

Tables 4.8 presents the results obtained on the four benchmark problems, *asia*, *tank*, *credit* and *car*.

The impact of the alteration power α can be seen on the numbers. When α is increased, that is when the data distribution gets more different than the original one, the results of the two algorithms are affected in different manner. K2GA running on the centralized data shows an overall decrease in terms of correct edges, while its associated SHD and the number of erroneous edges are increased. Erroneous edges are particularly affected by the increase of α and since the number of relevant edges is not changed to a significant extent, it can be said than increasing α leads to the addition of a lot of spurious edges in the network. This is obvious when comparing results obtained with α respectively set to 0.5 and 1. On the other hand, IMK2GA using distributed data is less affected by changes in α. Although the overall quality of the solutions is not as good as with solutions obtained by K2GA on data whose distribution is close to the original's ($\alpha = 0.1$), the various quality metrics do not vary to a large extent. This is particularly apparent on *asia* and *car*. Of course results remain problem dependent and variations can be observed as seen on tank between $\alpha = 0.5$ and $\alpha = 1$, but these are much less important than those observed on the centralized data. Note that at time, the structures are actually improved by increasing α. This is the case on *credit*, where the average number of correct edges grows from 8.66 ($\alpha = 0.1$) to 10.18 (α=1) and SHD is corrected from 4.58 to 3.13.

These results suggest potential benefits that could be found in the use of IM to model distributed data. IM seems able to make use of the different distributions to improve the search and reach better solutions than a serial GA. This may be related to the fact that introducing discrepancies in distributions leads the different islands to search different part of the search space. By using different datasets, the fitness function is consequently changed and solutions that are proposed by other islands at migrations may not remain as good solutions in a specific island population. This is a factor that

			Asia	Tank	Credit	Car
1 / 0.1	**centralized**	C	6.03 (0.41)	14.20 (2.24)	8.80 (0.83)	12.03 (1.11)
		SHD	2.60 (0.99)	11.63 (3.04)	5.10 (1.33)	9.73 (2.59)
		Rel.	7.23 (0.42)	19.93 (0.25)	12.00 (0.0)	14.13 (0.88)
		Err.	1.40 (0.88)	5.90 (1.27)	1.90 (0.70)	7.63 (2.09)
	distributed	C	5.63 (0.40)	11.83 (0.68)	8.66 (0.42)	9.67 (0.98)
		SHD	4.08 (1.04)	14.79 (0.84)	4.58 (0.53)	12.04 (1.67)
		Rel.	6.87 (0.31)	19.06 (0.22)	11.70 (0.20)	11.53 (0.75)
		Err.	2.84 (0.92)	7.56 (0.78)	1.54 (0.33)	10.18 (1.47)
1 / 0.5	**centralized**	C	5.77 (0.50)	14.50 (2.14)	8.80 (0.90)	11.27 (1.06)
		SHD	3.20 (1.30)	11.70 (3.06)	4.87 (1.63)	10.10 (2.53)
		Rel.	7.87 (0.34)	19.87 (0.34)	11.83 (0.37)	13.47 (0.72)
		Err.	1.10 (1.22)	6.33 (1.47)	1.83 (0.90)	7.90 (2.09)
	distributed	C	5.41 (0.46)	12.30 (1.03)	9.14 (0.34)	10.17 (0.82)
		SHD	4.53 (0.88)	14.75 (1.32)	4.03 (0.51)	11.36 (1.24)
		Rel.	7.05 (0.36)	19.07 (0.24)	11.52 (0.23)	11.68 (0.60)
		Err.	2.89 (0.77)	7.98 (0.80)	1.66 (0.28)	9.85 (1.12)
1 / 1	**centralized**	C	4.70 (0.59)	15.10 (2.02)	8.87 (1.06)	9.60 (1.50)
		SHD	8.97 (1.91)	13.33 (3.18)	11.37 (1.62)	19.10 (3.08)
		Rel.	7.10 (0.87)	19.90 (0.30)	12.00 (0.0)	13.50 (0.76)
		Err.	6.57 (1.98)	8.53 (1.65)	8.23 (0.84)	15.20 (2.73)
	distributed	C	5.49 (0.56)	9.93 (0.86)	10.18 (0.23)	9.54 (1.00)
		SHD	4.18 (0.91)	16.19 (1.48)	3.13 (0.39)	12.03 (1.45)
		Rel.	6.78 (0.37)	16.40 (0.54)	11.98 (0.06)	11.42 (0.84)
		Err.	2.90 (0.78)	9.72 (1.27)	1.33 (0.24)	10.15 (1.27)

Table 4.8: Characteristics of best BNs found on centralized and distributed datasets after 5000 generations on *asia*, *tank*, *credit* and *car*

can also increase the diversity in the population.

4.4 Summary

- Landscapes associated with BNs are generally multi-optimal. Using crossover on the different local optima has shown that it can be an operation beneficial for the search, generating offsprings that are better and different than their parents. According to empirical tests, this statement becomes more valid when the dimension of the problem increases as seen with *boerlage* and *alarm*.

- An IM version of K2GA is implemented to handle the many local optima it is likely to reach. IMK2GA shows improvements in terms of fitness of the final solutions obtained on all benchmark datasets over the serial K2GA in environment restricting diversity, that is when the selection pressure is high. However, this does not always correlate with better BN structures. Using 4 islands and 3 migrations leads to the best results from the configurations that were considered.

- In terms of best performance, that is the quality of the solutions and their associated fitness values in environments more likely to reduce the loss of diversity, IMK2GA shows benefits on small datasets over K2GA. On the largest problems, K2GA outperformed IMK2GA. Yet, this is to be put in perspective by the fact that IMK2GA exhibits a slow convergence rate and has not reached its full potential within the experimental bounds. Acting on the number of islands and migrations would affect these findings.

- Modelling distributed data is of growing interest in the machine learning community. By the inherent mechanism of IM, IM-based EAs are good candidates to perform such tasks and in particular BN structure learning while preserving privacy. IMK2GA is put to the test on several datasets that display different distributions.

- Empirical analysis shows that the use of IM makes K2GA less sensitive to differences in the respective distributions of distinct datasets. However, when the differences between these datasets are small, K2GA offers better results.

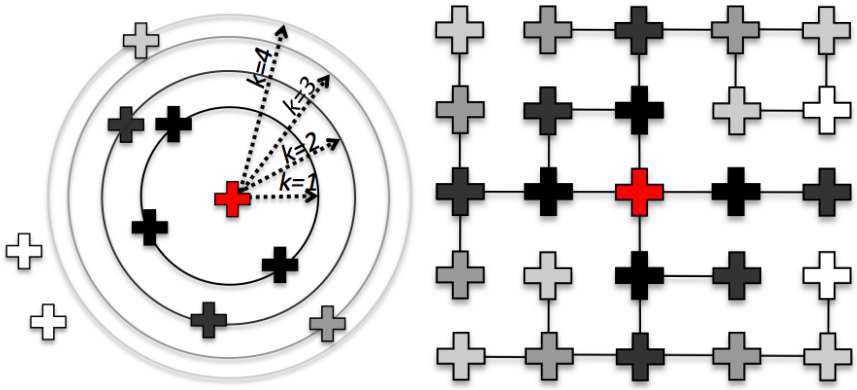
Chapter 5

Size of Neighborhood and Population Diversity

5.1 Neighborhood in the Space of Orderings

The notion of neighborhood is important with respect to EAs. The *neighborhood* of a given solution represents the space of solutions that are close to it in terms of gene characteristics. Some heuristics such as local search or best neighbor strongly depend on the expression of the neighborhood. In local search, a solution is mutated at each iteration, making sure that the offspring belongs to the set of neighboring solutions. In best neighbor, the set of all neighboring solutions is generated, the one presenting the highest fitness value becomes the offspring. The neighborhood of a solution needs to be expressed with respect to a parameter defining the degree of neighborhood. To understand this concept, it is more straightforward to study distances between solutions in Euclidean spaces as illustrated in Figure 5.1a. In Euclidean spaces, where a metric is defined, the neighborhood $N_{X,k}$ of a solution X is defined with respect to a *radius* k. Solutions distant by 1 unit from X belongs to the neighborhood $N_{X,1}$ of X. In order to increase the size of the neighborhood, k can be increased. Accordingly, in Figure 5.1a, excluding X, 3 solutions belong to $N_{X,1}$, 5 solutions belong to $N_{X,2}$, 6

solutions belong to $N_{X,3}$ and 8 solutions belong to $N_{X,4}$. The radius k would need to be further increased to include the two remaining solutions to the neighborhood of X. In Figure 5.1b, an example of neighborhood in discrete space is given. Each point represents a solution in the space. An edge is traced between two solutions X and Y if they belong to each other's neighborhood of radius 1, that is if $X \in N_{Y,1}$ and $Y \in N_{X,1}$. Considering the central point in Figure 5.1b as the solution X, it is now possible to define neighborhoods $N_{X,k}$ of different radius k where k represents the shortest path length in the graph. In other words, a solution Y is in $N_{X,k}$ iff there is a path of length k from X to Y. Excluding X, the sizes of the neighborhoods $N_{X,k}$ are $|N_{X,1}| = 4$, $|N_{X,2}| = 10$, $|N_{X,3}| = 16$ and $|N_{X,4}| = 22$.



(a) Euclidean spaces (b) Discrete spaces

Figure 5.1: Use of radius to define neighborhood of a solution in Euclidean and discrete spaces

The same idea can be extended to different types of search space once a metric is defined to quantify the distance between solutions. In discrete spaces such as permutation spaces, a metric can be expressed by means of minimum number of adjacent transpositions also referred to as minimum number of adjacent gene swaps $\rho(X, Y)$ needed to transform a solution X into another solution Y. In this chapter, we refer to $\rho(X, Y)$ as the *ordering similarity distance*. This choice of distance as a metric respects the fundamentals of metric spaces (Sutherland, 2009), which are non negativity and identity, symmetry and triangle inequality respectively summarized in (5.1), (5.2) and (5.3).

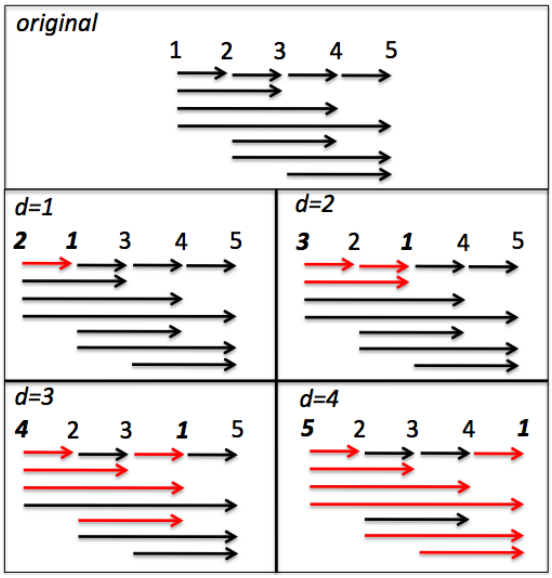


Figure 5.2: Impact of the distance d between two genes when using swap mutation. Red arrows represent edges that differ from the original structure.

$$\begin{cases} \rho\left(X,Y\right) \geq 0 \\ \rho\left(X,X\right) = 0, \forall X, Y \end{cases} \tag{5.1}$$

$$\rho(X,Y) = \rho(Y,X), \forall X, Y \tag{5.2}$$

$$\rho(X,Z) \leq \rho(X,Y) + \rho(Y,Z), \forall X, Y, Z \tag{5.3}$$

In (Jerrum, 1985), it is shown that the minimum number of adjacent transpositions $\rho(X,Y)$ between two permutations X and Y can be calculated by moving each individual element of a permutation towards its appropriate position in sequence. It is also proved that the order in which elements are being chosen does not affect the final value of $\rho(X,Y)$. Thus, the set of permutations is a metric space under $\rho(X,Y)$.

For many permutation representations, the ordering similarity distance ρ is sufficient to express differences between solutions. However, in K2GA, generating an ordering of nodes is not the final objective and although orderings are evaluated, it is their inherent

BN structures resulting from the run of the K2 deterministic process that are scored. Hence, when altering an ordering, the final BN structure that can be constructed from the ordering should also be considered. For example, considering the swap operator as mutation operator, swapping two adjacent genes from a solution X is likely to produce an ordering Y with a corresponding BN structure close to the the one of X. On the other hand, swapping two genes that are more distant will bring more changes in the resulting BN structures. Figure 5.2 illustrates how much alteration can result from swapping genes in an 5-node ordering with different distances between mutated genes. Note that this distance that is referred to as *mutation distance* has a different meaning than the ρ distance. From an original ordering $\{1-2-3-4-5\}$, four orderings are created by swapping the first gene with another one with mutation distances d of 1, 2, 3 and 4. To illustrate the importance of the mutation distance on the degree of alteration that can be observed, we consider for each ordering a BN structure with a maximum number of edges, that is 10 edges in total. Each mutated network is then compared with the original structure and the numbers of edges that differ highlighted. For example, when $d = 1$, only the edge $2 \rightarrow 1$ differs from the original structure where the edge between nodes 1 and 2 is $1 \rightarrow 2$. The four mutated structures exhibit different degrees of alteration, with an increase of the mutation distance leading to an increase of differences. This example describes how the locality breaks down when the distance between two genes increases, or in other words, how the concept of neighborhood needs to be considered with respect to the mutation distance. This illustration by means of number of altered edges is actually equivalent to calculating the KTD between each mutated ordering and the original one. It also confirms the relevance of KTD as a metric to assess distance between node orderings.

5.2 Distance-Based Mutation Operators

In this section, several mutation operators that use mutation distance are presented. In addition to describing their mechanisms, they are also compared with respect to

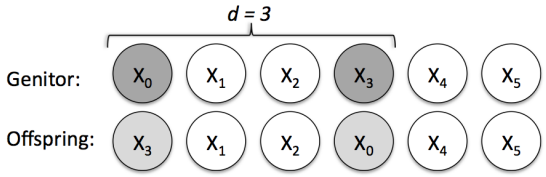


Figure 5.3: Example of swap mutation from a solution X of size $n = 6$, using mutation distance $d = 3$

their effect on the offspring orderings and corresponding BN structures when mutation distances are changed. The impact of these operators are observed through studying the relation between neighborhood radius k and the ordering similarity distance ρ and some structural and ordering measurements such as SHD and KTD.

5.2.1 Swap Mutation

The swap mutation operator is illustrated in Figure 5.3. A gene X_i is selected at random in the ordering and swapped with another gene selected randomly between X_{i-d} and X_{i+d}, unless restricted by the position of X_i. For example, if $i = 2$ and $d = 3$, the second gene to be mutated can only be X_5 since $i-d < 0$. Here, the mutation distance d represents the distance between the two genes to swap. The ordering similarity distance between genitor and offspring can be expressed as (5.4).

$$\rho = 2d - 1 \tag{5.4}$$

5.2.2 Insert Mutation

With insert mutation, a gene X_i is randomly selected and either inserted after X_{i+d} or before X_{i-d}. This position is chosen at random unless a constraint exists because of the position of X_i as explained for the swap operator. Figure 5.4 illustrates an example of insert mutation with a distance d set to 3 and X_0 as starting edge. Here, the distance d represents the distance between the starting gene X_i and the position it is inserted at. ρ can be expressed as (5.5).

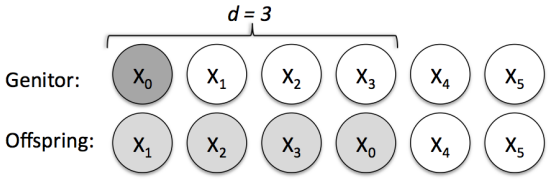


Figure 5.4: Example of insert mutation from a solution X of size $n = 6$, using mutation distance $d = 3$

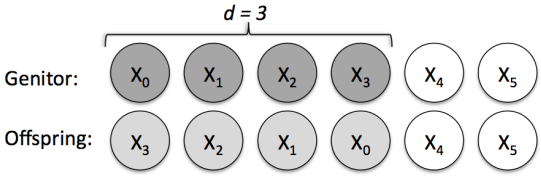


Figure 5.5: Example of invert mutation from a solution X of size $n = 6$, using mutation distance $d = 3$

$$\rho = d \tag{5.5}$$

5.2.3 Invert Mutation

Invert mutation acts on a subgroup of genes of size $d+1$. The first gene of the subgroup is selected at random. Figure 5.5 describes an example of invert mutation where the order in which genes of the subgroup appear in the genitor is reversed to generate the offspring. In this illustration, d is set to 3. ρ is expressed as (5.6).

$$\rho = \frac{d * (d + 1)}{2} \tag{5.6}$$

5.2.4 Scramble Mutation

In contrast with the three previous methods, the scramble mutation operator is not deterministic, that is given a genitor solution, the offspring can differ across repeated applications of the operator. This is due to a random step in the process. Similarly to the invert mutation operator, a subgroup of genes of size $d+1$ is selected. The order in

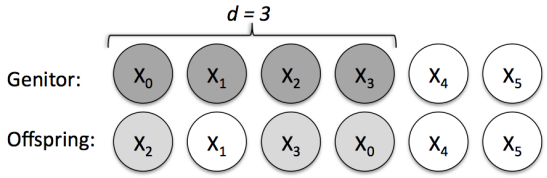


Figure 5.6: Example of scramble mutation from a solution X of size $n = 6$, using mutation distance $d = 3$

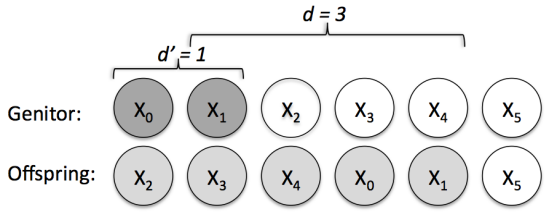


Figure 5.7: Example of displacement mutation from a solution X of size $n = 6$, using mutation distance $d = 3$, where d' takes the random value 2

which genes appear in the genitor is changed at random to create an offspring as shown in Figure 5.6, for $d = 3$. Due to the random re-ordering, the ordering similarity distance between genitor and offspring needs to be expressed as a range of values following (5.7).

$$\rho \in \left[1, \frac{d * (d+1)}{2}\right] \tag{5.7}$$

5.2.5 Displacement Mutation

The displacement mutation selects a subgroup of size d' to be moved. The position to which the subgroup is moved is defined by means of a distinct distance d as shown in Figure 5.7. d' is set randomly, while respecting position constraints. The order in which the genes appear in the subgroup is not modified throughout the process. Consequently, ρ is expressed as (5.8), where n refers to the problem size.

$$\rho \in [d, d * (n - d)] \tag{5.8}$$

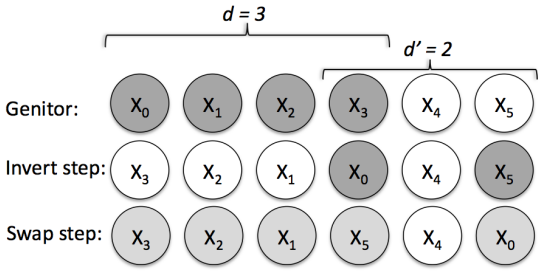


Figure 5.8: Example of invert+swap mutation from a solution X of size $n = 6$, using mutation distance $d = 3$, where d' takes the random value 2

5.2.6 Invert+Swap Mutation

The remaining two mutation operators combine some of the previous methods and were first defined in (Deep and Mebrahtu, 2011), where they show improvements over other mutation types on the Traveling Salesman Problem. The first of the two operators runs successively the invert and the swap mutation operators and is referred to as invert+swap. Because it is important not to increase consequently the number of parameters related to an algorithm, it was decided to only manage one distance parameter d. In this case, d is the mutation distance corresponding to the invert mutation, thus the number of genes to be part of the subgroup being inverted is equal to $d + 1$. During the second step of the process, the swap mutation uses a random distance d' that is related to d. The range of value for d' and the expression of ρ is described in (5.9). An example can be found in Figure 5.8 where the manageable distance d is set to 3, while a random value of 2 was drawn for d'. The offspring is the result of the swap step.

$$\rho \in [d + 1, 2n - d - 1] \tag{5.9}$$

5.2.7 Invert+Displacement Mutation

In order to generate a solution from a genitor X using the invert+displacement mutation operator, invert and displacement mutations are run consecutively. In the same

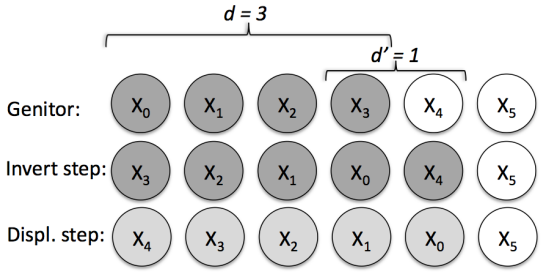


Figure 5.9: Example of invert+displacement mutation from a solution X of size $n = 6$, using mutation distance $d = 3$, where d' takes the random value 1

manner as the invert+swap mutation, two distances need to be defined but only one of them, d, needs to be manually set. In this approach, d is associated with the invert mutation step, while the displacement mutation step is characterized by d', set at random, accordingly to the value of d. In short and as summarized in Figure 5.9, $d+1$ represents the number of genes to form the subgroup whose positions are to be inverted, while d' represents the distance to which the previously defined subgroup needs to be displaced. The relation between d and d' is given in (5.10) along with the general expression for ρ.

$$\rho \in \left[\frac{(d+1) * (d+2)}{2}, \frac{-2d^2 + (2n-3)d + 2n - 1}{2} \right] \tag{5.10}$$

Because of the way both distances d and d' are correlated in invert+swap and invert+displacement mutations, setting small values of d means that the values for d' can be drawn from a larger range and thus be large. Consequently, dissimilarities can be observed between genitor and offspring solutions even if d is set with low values. It is understood that both invert+swap and invert+displacement mutations lead to larger dissimilarities between genitor and offspring solutions. Tables 5.1 and 5.2 present all values that ρ can take for each mutation type and different distances on two problems of respective sizes $n = 8$ and $n = 23$. Note that these problem characteristics match those of *asia* and *boerlage*. Based on these numbers, it is possible to foresee how the different operators will affect solutions and thus influence algorithm design choices. For example, the *insert* and *swap* operators can be qualified as conservative in comparison

	ρ			
Mutation operator	$d = 1$	$d = 2$	$d = 3$	$d = 4$
Swap	1	3	5	7
Insert	1	2	3	4
Invert	1	3	6	10
Scramble	1	[1;3]	[1;6]	[1;10]
Displacement	[1;7]	[2;12]	[3;15]	[4;16]
Invert+Swap	[2;14]	[3;13]	[4;12]	[5;11]
Invert+Displacement	[3;13]	[6;18]	[10;22]	[15;25]

Table 5.1: Influence of mutation operators and mutation distance d on ordering similarity distance ρ on a 8-variable problem ($n = 8$, such as *asia*)

with the rest of the methods, regardless of the mutation distance, that is offspring solutions should exhibit less differences than with other methods. On the other hand, $invert + displacement$ and *displacement* show high possible values of ρ, even for small d. This means that small distances are also able to lead to distant solutions. *Invert* and *scramble* mutations give the same maximum ρ values, but the use of *scramble* rather than *invert* introduces some randomness since ρ takes its value within a range of values. Finally, $invert + displacement$ displays another distribution of ρ in which its mean value does not vary despite changes in d. Only the size of the range varies, from large to smaller when d is increased.

5.2.8 Choice of Mutation Operator and Bayesian Network Structure Quality

In this section, the previous estimates on dissimilarities between genitor and offsprings presented in Tables 5.1 and 5.2 are compared with empirical results relevant to BN structure quality. The aim of this step is to assess to what extent ordering similarity distance as a measure of change in the space of orderings corresponds to change in structure and evaluation in the space of structures generated by K2.

In this set of experiments, 30 orderings are generated at random. Each of them is

	ρ											
	1	2	3	4	5	6	7	8	9	10	11
Mutation operator	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$	$d = \rho$
Swap	1	3	5	7	9	11	13	15	17	19	21
Insert	1	2	3	4	5	6	7	8	9	10	11
Invert	1	3	6	10	15	21	28	36	45	55	66
Scramble	1	[1;3]	[1;6]	[1;10]	[1;15]	[1;21]	[1;28]	[1;36]	[1;45]	[1;55]	[1;66]
Displacement	[1;22]	[2;42]	[3;60]	[4;76]	[5;90]	[6;102]	[7;112]	[8;120]	[9;126]	[10;130]	[11;132]
Invert+Swap	[2;44]	[3;43]	[4;42]	[5;41]	[6;40]	[7;39]	[8;38]	[9;37]	[10;36]	[11;35]	[12;34]
Invert+Displacement	[3;43]	[6;63]	[10;82]	[15;100]	[21;117]	[28;133]	[36;148]	[45;162]	[55;175]	[66;187]	[78;198]

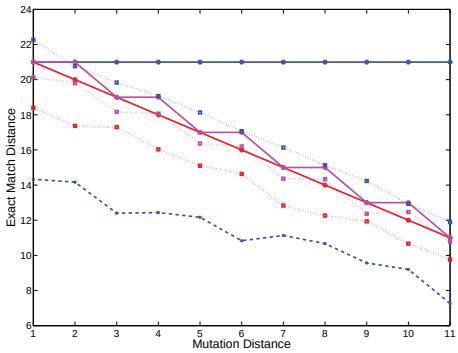
Table 5.2: Influence of mutation operator and mutation distance d on ordering similarity distance ρ on a 23-variable problem ($n = 23$, such as *boerlage*)

mutated once with mutation distance d, using each of the mutation types described in this chapter. The solutions that are produced are compared with their genitor. The process is repeated for every value d can take on each problem of size n, that is $d \in [1; \lfloor n/2 \rfloor]$. The upper bound of $\lfloor n/2 \rfloor$ on d ensures that any gene of a solution can be affected by the mutation. Greater d values would limit the number of possible mutations since genes cannot be moved outside the boundaries of the solution size n. In order to evaluate the effect of the different mutation types and mutation distance values, several measurements are performed. First, it is important to study the impact the changes have on the orderings as the mutation directly acts in this specific search space. Hence, the *exact match*, *edit* and *Kendall-Tau* distances are computed as described in Section 2.4. Second, solutions need to be compared with respect to the BN they represent. *SHD* between offspring and genitor is calculated since it evaluates the quality of edges that exist in a BN structure. Finally, the fitness difference between the two solutions is recorded since it is important to relate the changes that are brought to the scoring method that guides the search. These measurements are shown in Figure 5.10, for different mutation distances. This set of figures only shows results obtained on the *boerlage* dataset. Similar set of figures are given in E.1, E.2, E.3, E.4 and E.5 for the remaining five datasets.

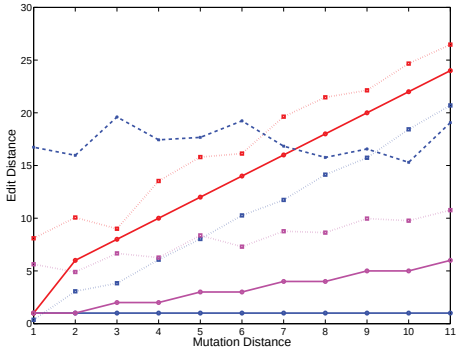
The first observation that can be made from all the figures is that we observe a discrepancy between the measurements that relate to the orderings and those that relate to the final BN structure. For instance, for all measurements following the swap mutation, the level of alteration in terms of orderings can be described as constant under increase of d. Yet, the impact of the variation of d on the structures that are obtained following the runs of K2 is different as illustrated by an increase of the SHD. This stresses the importance already mentioned to always consider both ordering and structure spaces when analysing solutions. In addition and by definition, edit and exact match distances cannot show variation for the swap mutation. On the other hand, KTD generally exhibits a linear increase for invert, insert, scramble and invert+swap. The remaining two mutation types, displacement and invert+displacement are overall more disruptive

operators. This observation is particularly valid when d is small, illustrating the fact that slight changes cannot be performed based on these two operators. Note that both involve a displacement step that appears as the single mutation type presenting the most ordering alterations on all datasets. Since our objective is to compare the different values of ρ calculated theoretically against empirical results on final solutions, SHD is the most meaningful metric here. As expected from Tables 5.1 and 5.2, the less disruptive method is insert. Swap is also less disruptive in comparison with the rest of the methods. Scramble and displacement are generally varying in the same range. This is probably an effect of the random process on which they rely. Comparable curves are obtained for invert and invert+swap when d is high. At lower d, invert+swap is more disruptive than invert. Finally the remaining invert+displacement shows a much higher SHD throughout the whole range of d and is the most disruptive mutation operator over all. Using these observations, it can be concluded that empirical results confirmed that ρ is a good approximation to the extend of disruption introduced in a BN ordering at mutation.

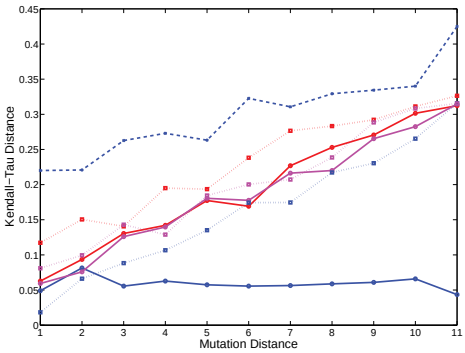
Since changes in mutation distance d bring changes in solutions, it is interesting to also assess how close to a random process each of the operator is when d is high. If using a mutation operator brings the same results than generating a new solution at random, it might be more judicious not to use such operator. Ideally, a mutation operator should generate offsprings close to random solutions when d is set to its maximum value, although there should remain a difference. Experiments were run in which two random orderings were compared on each of the benchmark problems by means of KTD and SHD. The process was repeated 30 times in order to match the experimental settings used to assess the mutation distance operators. Tables 5.3 and 5.4 present results obtained after 30 repeat runs along with respectively KTD and SHD obtained at $d = \lfloor n/2 \rfloor$ for comparison purposes. Values that appear to be not statistically significant from those of the random generation are annotated with a * symbol. For this, unpaired t-tests were performed. These comparisons show that there is a difference between using mutation operators and large distances and generating random orderings. This



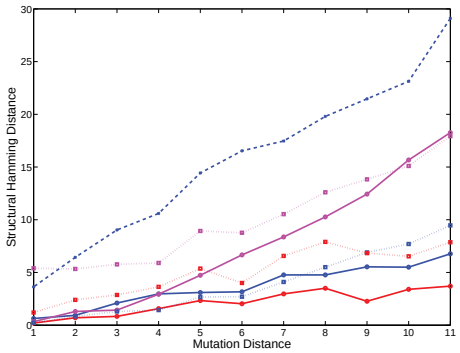
(a) Exact distance vs. mutation distance



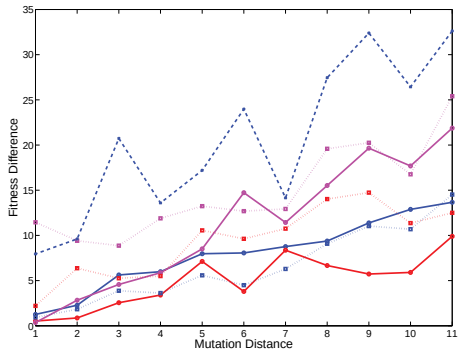
(b) Edit distance vs. mutation distance

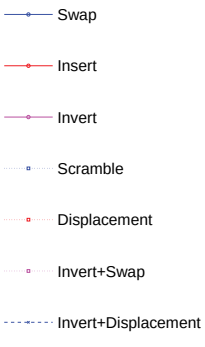


(c) KTD vs. mutation distance



(d) SHD vs. mutation distance





(e) Fitness difference vs. mutation distance

Figure 5.10: Effect of choice of mutation on produced solution on *boerlage*

Mutation operator	*asia*	*tank*	*credit*	*car*	*boerlage*	*alarm*
Swap	0.20 (0.12)	0.10 (0.05)	0.11 (0.09)	0.08 (0.05)	0.04 (0.03)	0.03 (0.02)
Insert	0.35 (0.11)	0.33 (0.08)	0.35 (0.10)	0.33 (0.07)	0.31 (0.07)	0.31 (0.07)
Invert	0.33 (0.14)	0.32 (0.13)	0.27 (0.12)	0.33 (0.12)	0.31 (0.07)	0.26 (0.05)
Scramble	0.32 (0.12)	0.29 (0.12)	0.31 (0.10)	0.29 (0.07)	0.32 (0.06)	0.30 (0.05)
Displacement	0.40 (0.16)	0.35 (0.08)	0.39 (0.12)	0.35 (0.08)	0.33 (0.09)	0.33 (0.07)
Invert+Swap	0.37 (0.19)	0.38 (0.12)	0.36 (0.18)	0.35 (0.07)	0.32 (0.08)	0.30 (0.06)
Invert+Displacement	0.48* (0.17)	0.43* (0.13)	0.39 (0.11)	0.44* (0.13)	0.43* (0.10)	0.34 (0.08)
Random Generation	0.51 (0.10)	0.50 (0.10)	0.49 (0.11)	0.48 (0.07)	0.48 (0.06)	0.49 (0.06)

Table 5.3: Comparison of KTD of solutions obtained by distance-based mutations and by random generation

is an important point since it means that even for large distances, mutation operators keeps some information from the genitor solutions. However, this is not always true. For example, invert exhibits no significant difference with random generation on *asia*. Since *asia* is a problem of very small dimension, this result is not significant. It is seen that the same invert operator shows differences on the rest of the benchmarks. On the other hand and at maximum distance d, invert+displacement, the operator defined as the most disruptive, reaches a similar level of disruption than if orderings were generated at random on all benchmarks except *alarm* by SHD. Considering KTD solely, invert+displacement is similar to a random generation on *asia*, *tank* and *boerlage* only.

Mutation operator	*asia*	*tank*	*credit*	*car*	*boerlage*	*alarm*
Swap	5.43	12.80	6.67	9.53	6.77	11.60
	(2.01)	(5.31)	(3.00)	(5.00)	(3.86)	(6.00)
Insert	3.10	9.93	3.70	7.30	3.70	8.03
	(2.89)	(5.34)	(3.36)	(6.81)	(3.33)	(4.59)
Invert	7.80*	24.77	9.80	20.77	18.27	44.10
	(3.20)	(7.10)	(2.93)	(6.58)	(3.50)	(5.00)
Scramble	4.87	14.33	5.67	11.43	9.47	25.50
	(2.92)	(6.63)	(2.83)	(6.69)	(3.42)	(7.54)
Displacement	4.77	10.30	5.90	7.87	7.87	14.73
	(3.62)	(6.52)	(3.90)	(7.46)	(5.49)	(14.13)
Invert+Swap	6.43	24.50	8.20	21.70	17.93	42.13
	(2.67)	(7.28)	(3.19)	(8.49)	(6.13)	(9.66)
Invert+Displacement	10.93*	33.67*	15.50*	28.83*	29.10*	57.00
	(3.36)	(8.43)	(4.22)	(8.07)	(8.18)	(15.38)
Random Generation	10.13	36.93	15.10	31.27	33.17	73.30
	(3.53)	(8.49)	(4.07)	(7.72)	(5.66)	(9.70)

Table 5.4: Comparison of SHD of solutions obtained by distance-based mutations and by random generation

5.3 Competing Mutating Agents

5.3.1 Implementation

Many EAs including the classic GA follow the principle that exploration of the search space should be first performed before the exploitation of particular regions. In terms of neighborhood, this often means producing offspring from the whole search space in explorative phases while increasing the focus on the neighborhood of the best solutions as the search progresses. Typically, in the last generations of a GA process, there are only a few differences between solutions in a population and the offsprings produced with respect to their genes. Other approaches such as Simulated Annealing (SA) (Kirkpatrick et al., 1983) depend on a temperature variable that manages how much degradation can be introduced into the search. In SA, a neighboring solution is produced and scored at each generation. This solution replaces the previous best with a probability depending on the score and the temperature. A high temperature will support replacement of the current best solution by a solution of lower fitness while a low temperature will favor solutions of better fitness. The temperature in SA decreases as the search progresses. This policy aims at allowing degradation while exploring the search space and refraining it when exploiting an area of the search space.

Runs of EAs involve some stochasticity. Hence, it is hard to anticipate how much exploration needs to be done. For instance, the basin of attraction of a global optimum can be reached early during the search, leading the algorithm to find an optimal solution early. On the other hand, many optimization problems exhibit multi-optimal landscapes. Such problems, such as the BN structure learning are likely to trap the search in a local optima, exploiting an area of the search space which is good but not optimal. Taking these two contrasting scenarios in consideration, a different direction is taken in this section that presents a new algorithm where exploration and exploitation are both performed in parallel throughout the search. The approach makes use of distance-based mutation operators on a population of agents in order to vary the size

of the neighborhood of solutions through varying the level of disruption ρ introduced by each agents. Thus, ρ is set accordingly to the agent's position in the population.

Varying the way mutation is performed during the search is not a novel idea. The use of adaptive mutation has been explored in previous works including (Cobb and Grefenstette, 1993) and (Thierens, 2002). In these works, several strategies were proposed around a GA framework. In (Cobb and Grefenstette, 1993), a so-called triggered hypermutation is introduced that increases the mutation rate under some circumstances such as a degradation of the GA performance over time. This strategy was found helpful since it preserves information on the population when used on changing environments, that is when the fitness landscape varies during the GA run. Some ideas presented in (Thierens, 2002) are closer to those of SA, where the mutation rate decreases as the search progresses to become more and more restrictive on the degree of random information introduced in the population at each generation.

Here, a population of agents is considered in which each agent aims at improving its assigned solution by means of distance mutation. Large distances are allowed for agents in low positions while best agents, that is those with the highest quality solutions, are constrained with smaller distances. Since each agent only uses mutation to improve its solution and because each agent aims at reaching the best positions in the population, we call this approach COMpeting Mutating Agents (COMMA). Algorithm 5 presents the outline of COMMA used for maximization optimization. For each position pos_j in the population *pop* sorted in ascending order, a mutation distance d_j is set such that for two agents at positions e and f, $d_e \leq d_f$ if $e < f$. Since it can be beneficial to allow degrading solutions to be accepted, as seen in SA, a probability p_j is also set for each pos_j. Each agent a_i is initially assigned a random solution s_i. The population is then sorted by fitness. At each generation, each agent mutates s_i using the distance $dist_i \in [1, d_r]$ defined according to its position r in the population. If the mutated solution s_{new} has a better fitness than s_i, a_i replaces s_i with s_{new}. If s_{new} has a poorer fitness than s_i, s_{new} only replaces s_i with probability p_r.

Mutation type	Rule 1			Rule 2		
	Fitness	C	SHD	Fitness	C	SHD
tank						
Swap	-10030.6	13.4	12.9	-10026.7	13.4	11.5
Insert	-10031.6	13.4	12.7	-10026.1	14.0	11.6
Invert	-10032.8	12.4	14.5	-10028.6	14.2	11.4
Scramble	-10039.9	12.9	15.0	-10026.9	15.0	10.2
car						
Swap	-7059.0	11.6	12.3	-7053.2	13.3	9.6
Insert	-7059.9	11.2	13.9	-7052.3	13.6	8.9
Invert	-7060.5	11.1	13.0	-7054.4	12.8	10.4
Scramble	-7059.7	11.4	13.0	-7052.8	13.6	8.9
boerlage						
Swap	-30733.1	16.8	26.6	-30679.7	19.4	20.4
Insert	-30733.9	15.4	27.4	-30681.4	18.4	20.8
Scramble	-30732.2	15.6	28.2	-30688.5	18.6	20.6

Table 5.5: Comparison of two rules to assign values to mutation distances

Parameter tuning can be a difficult task, especially when the number of parameters involved in an algorithms is large. The above description of COMMA implies several variables such as the size of the population $|pop|$, the number of bins $|\Delta|$ in which the population is divided or the mutation distance $\Delta_j$values for each bin j. Thus, following some preliminary comparisons, a rule was set in order to define distances for each bin in relation with $|pop|$, $|\Delta|$ and the number n of variable of the problem. Two rules were originally compared. In the first rule, the bins would each be set with small distance values $\Delta_j \in [1; |\Delta|]$. Because, we considered only 4 bins, $\Delta_j \in [1; 4]$. On the other hand, a second rule was set that covers a wider range of distances, that is the best agents will be set with the minimum distance $\Delta_0 = 1$, while the agents in the worse bin will be attributed a distance $\Delta_{|\Delta|-1} = \left\lfloor \frac{n}{2} \right\rfloor$. The rest of the distances are equidistant to each others. More generally the distance in each bin in the second rule can be expressed as (5.11).

$$\Delta_j = \begin{cases} 1 \text{ , if } j = 0 \\ \left\lfloor \frac{j*n}{2*(|\Delta|-1)} \right\rfloor \text{ , otherwise} \end{cases} \tag{5.11}$$

In order to assess the efficiency of these two approaches, some mutation types were selected and COMMA run on 3 datasets, namely, *tank*, *car* and *boerlage*. 10 runs were performed and mean fitness, correct edges and SHD of the best solutions compared. The maximum number of generations was set to 1000 FEs for *tank*, the smaller problem, while *car* and *boerlage* were left to run up to 5000 FEs. Note that only a sample of the mutation operators and datasets were used for this experiment, as more experiments were not justified. Results presented in Table 5.5 respectively suggest that the second rule which allows larger distances gives better results than the first rule on all three measurements, and this for all datasets. Thus, distances are set according to (5.11) for the rest of the chapter and the distances Δ_i are no longer considered as input parameters in COMMA. Once distances are set, each agent needs to be allocated to a bin. In the proposed design, the population is split into bins of equal size when this is possible, that is when the number of agents is divisible by the number of bins. In the other case, the q worse bins have an extra agent such as $q = |pop| - \frac{|pop|}{|\Delta|} * |\Delta|$. The distance Δ_j is allocated to all agents a_l, with $l \in \left[j * \left\lfloor \frac{|pop|}{|\Delta|} \right\rfloor ; j * \left\lfloor \frac{|pop|}{|\Delta|} \right\rfloor - 1\right]$ if $j < q$, $l \in \left[j * \left\lfloor \frac{|pop|}{|\Delta|} \right\rfloor + j - q; (j+1) * \left\lfloor \frac{|pop|}{|\Delta|} \right\rfloor + j - q\right]$ otherwise.

As seen in Chapter 4, the use of IM showed improvements over its serial counterparts when learning BN structures using GA by maintaining diversity. Hence, IM was also implemented for COMMA and is referred to as IM-COMMA. The outline of IM-COMMA is presented in Algorithm 6 for n islands, m migrations and migration intervals of size $migInterval$. The search is split into evolution stages stg_l where $migInterval$ generations are performed. At each stg_l, the COMMA process is performed and paused to allow migration. Solutions from the λ best agents from each island are sent to the neighbouring island at migration and assigned to the λ worse agents in its population following a ring topology with a best-worse policy.

Algorithm 5: $COMMA$

Initialize *pop* of σ agents with random solutions, distance vector d of size σ and probability vector p of size σ
repeat
  Sort *pop* by fitness in ascending order
  for each agent $a_i, i \in [0, \sigma - 1]$ **do**
    Get position r of a_i in *pop*
    Generate new solution s_{new} with fitness fit_{new} by mutating s_i with distance $dist_i$ selected with uniform probability from $[1, d_r]$
    if $fit_{new} > fit_i$ **then**
      Assign $s_i = s_{new}$
    else
      Assign $s_i = s_{new}$ with probability p_r
    end if
  end for
until Stopping condition met

5.3.2 Experiments

Experiments regarding COMMA are presented in two steps. First, different versions of COMMA and IM-COMMA with or without degradation are compared to GA-based methods. Second, and since several mutation types exist, different versions of COMMA are developed and compared against each other on the benchmark suite.

5.3.2.1 Comparison of COMMA and GA

Two versions of COMMA and IM-COMMA were implemented in order to observe the effect of degradation. We set one of the COMMA and one of the IM-COMMA with degradation probabilities p_j equal to zero while the two other versions were set with degradation probabilities of [0.2, 0.2, 0.4, 0.4, 0.6, 0.6, 0.6, 0.8, 0.8, 0.8], respectively stated from the best to the worse position in the population of agents. For the time of the experiments, we respectively call these methods $COMMA$, $IM - COMMA$, $COMMA_d$ and $IM - COMMA_d$ where d stands for degradation. As the population size for $COMMA$ and $COMMA_d$ was set to 10, using 4 distinct mutation distances and probabilities helped observing how agents evolve. Mutation distances were set relative

Algorithm 6: $IM - COMMA$

Initialize k populations of σ agents with random solutions, distance vector d of size σ and probability vector p of size σ
for each evolution stage $stg_l, l \in [0, m-1]$ **do**
  for each island $isl_k, k \in [0, n-1]$ **do**
    $gen_k = 0$
    repeat
      Sort pop_k by fitness in ascending order
      for each agent $a_i, i \in [0, \sigma - 1]$ **do**
        Get position r of a_i in pop_k
        Generate new solution s_{new} with fitness fit_{new} by mutating s_i with distance $dist_i$ selected with uniform probability from $[1, d_r]$
        if $fit_{new} > fit_i$ **then**
          Assign $s_i = s_{new}$
        else
          Assign $s_i = s_{new}$ with probability p_r
        end if
      end for
      gen_k++
    until $gen_k = migInterval$
  end for
  if $l \neq m$ **then**
    Select λ best orderings to create subpopulation mig_k
    if $i \neq 0$ **then**
      Replace λ worse orderings in pop_k by mig_{i-1}
    else
      Replace λ worse orderings in pop_k by mig_{n-1}
    end if
  end if
end for

Dataset	K2GApop	IMK2GAmigInterval
Asia	100	150
Tank	50	250
Credit	50	300
Car	20	1200
Boerlage	20	800
Alarm	20	600

Table 5.6: K2GA and IMK2GA settings

to the number of nodes in each benchmark. $IM-COMMA$ and $IM-COMMA_d$ were both set with 4 islands, 3 migrations and 7 generations were chosen as migration interval in order to reach 1000 FEs. K2GA and IMK2GA were set following Table 6.5, with tournament selection of size 4.

In this set of experiments, the number of cases in each dataset was set to 3000. Results regarding quality of the solutions obtained by the different methods are presented in Table 5.7. This encompasses the number of correct edges (C), SHD, number of relevant edges ($Rel.$) and number of erroneous edges ($Err.$). 30 runs were performed for each algorithm on each dataset, except *alarm* and unpaired t-tests carried out. For the case of *alarm*, 10 runs were performed due to the computational time that is involved. Best values over all methods appear in bold while those not statistically significant from the best (p-value>0.003 after Bonferroni correction for a 95% confidence level) are marked with a $*$ symbol.

Overall, $COMMA$ seems competitive with the GA-based methods although presenting a few exceptions such as a significantly higher SHD on *asia* and a higher number of erroneous edges on *credit*. On the other hand, $IM-COMMA$ seems less competitive when the problem size increases. For example, $IM-COMMA$ exhibits significantly higher SHD for the three largest problems, *car*, *boerlage* and *alarm*, while its serial counterpart does not. The benefit of using IM in conjunction with COMMA can be questioned. While IM were originally used to avoid local optima or to bring together the outcomes of different evolutions by means of crossover, it was applied here to an algorithm that does not converge, nor evolve its solutions using crossover. Thus, a

migration in $IM-COMMA$ can simply be considered as a successful mutation of the solutions from the worse agents in the island populations.

Degradation was introduced in COMMA in order to prevent the search to becoming on the same solutions for too many generations. Yet, according to the results of Table 5.7, degradation does not bring much improvement to the results of COMMA and IM-COMMA. Its use even leads to solutions of lower quality on problems such as *car* and *boerlage*, where $COMMA_d$ has a significantly lower C and higher SHD. We recall that the best agents are subject to degradation at a 0.2 rate in this implementation. In order to further understand how such results are obtained, it is possible to study how agent positions change in the population over time. In Figure 5.11, the positions of the agents is plotted for $COMMA$ (Figures 5.11a, 5.11b) and $COMMA_d$ (Figures 5.11c, 5.11d) on a typical run on *tank*. Each row from these figures represents the positions of the agents at a given generation. The plot is to be read from top to bottom where the first row represent the initial generation and the last row the final generation. Each agent is assigned a distinct color kept throughout the run. Thus, by reading two successive lines, the position changes can be observed. Note that agents are ordered in the population according to the fitness of their solutions. In the representation, the best agents are on the right hand side while the worse ones are on the left hand side. Knowing this, Figure 5.11 can be analyzed. On the run of $COMMA$, agents exchange positions less often than on the run of $COMMA_d$. This is particularly obvious among the bottom half of the population. The impact of introducing degradation can be seen as a succession of changes in 5.11c and 5.11d, and this even when the algorithm is approaching the end of its run, suggesting a constant exploration of the search space. In comparison, $COMMA$ has reached a state of stability after 75 generations where the solutions of the agents cannot be improved sufficiently to cause a position change. Although allowing degradation may have seemed a good idea at first, it can also be disruptive and not support the search. For instance, from Figures 5.11c, 5.11d, the best agent at generation 0 remains in the top two positions for more than 10 generations because the degradation probability is very small in these positions. As soon as another

agent takes its position, degradation starts playing a more important role, having for consequence the fast drop in the agent position. Half way through the run, the same agent, which was initially the best one, belongs to the two worse agents' positions in the population and remains in low positions until the end of the run. More generally, it seems that beyond a certain number of generations, agents that are in the lowest positions will remain there until the end of the run. Figures 5.11c, 5.11d also display a swap of colors between generations 0 and 100, which means that the worse agents at the beginning of the run ended being the best ones, while the initial best agents finished the run with the worse solutions. The last comment regarding agent's positions concerns Figures 5.11a and 5.11b where the contribution of the strategy of $COMMA$ can be observed. By looking closely at the final best agents and at how it reached the top position, we see that it occupied every possible position in the population. It also reached the second position after generation 52, but did not remains at this position until the end of the run unlike observed with $COMMA_d$.

Looking back at Table 5.7, it seems that in the context of allowing degradation, the use of IM offers better results than the non-parallel $COMMA_d$. Yet, the solutions obtained by $IM - COMMA_d$ remain overall below solutions found by the best methods, with the difference reaching statistical significance for C on *car* and for SHD on *boerlage*.

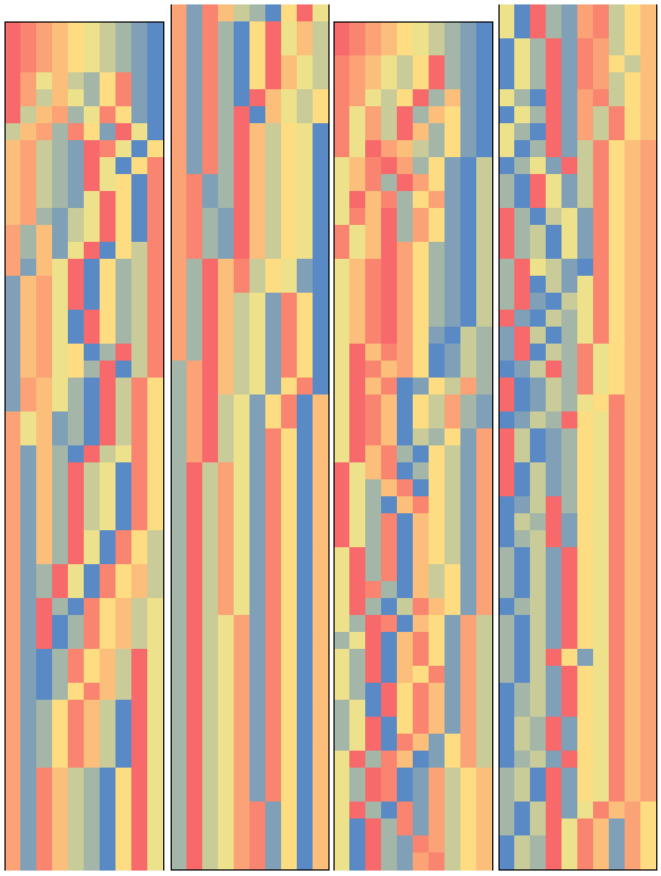
Across all methods run on *tank*, $K2GA$ exhibits the best C and SHD, which contrasts with the general trend observed over all other benchmark problems. In order to investigate this result, it is helpful to have an insight on how the fitness of the best solution varies over time with all methods and to compare it with the evolution of the solution quality, such as SHD for instance as done in Figures 5.12. As expected, $K2GA$ converges after around 200 fitness evaluations to a solution of lower fitness than the rest of the algorithms. The use of IM prevents this from happening and the state of convergence is not reached over the 1000 fitness evaluation boundary for $IMK2GA$. The simple $COMMA$ is the only COMMA-based algorithm to reach a higher fitness than K2GA. Yet, this does not correlate with a low SHD at the end of the run. Looking at the evolution of the SHD throughout the run illustrates a discrepancy between CH

score and BN structure quality. For example, the case of $IMK2GA$ shows that the SHD decreases with the first 400 fitness evaluations and is at this stage much lower than those of solutions obtained by other methods. However, beyond 400 fitness evaluation and despite observing an increase of the fitness of the best solution, the SHD increases and ends up being greater than the one of $K2GA$. This finding fits in what has been described in Chapter 3, where *tank* was defined as exhibiting a lack of correlation between fitness and SHD.

Finally, results on the two largest benchmark problems show that $IMK2GA$ performs best in comparison with $COMMA$ and $IM-COMMA$. This is maybe due to the difficulty of the problems and the fact that $IMK2GA$ converges faster than the COMMA-based methods that are still improving their fitnesses at the time the search is stopped as shown in Figure 5.13. This suggests that better results would be obtained if they were left to run for more generations. It may also be a hint that the current settings are not optimal and too much exploration is being performed. Assessing mutation operators other than swap may be a way to solve this problem and is treated in the next section. In addition, it is found that the use of degradation is beneficial on *alarm*, with $IM-COMMA_d$ exhibiting solutions that are better than those of $K2GA$ in every aspect. The use of degradation also affects the way the IM implementation performs. On *alarm*, $IM-COMMA$ shows the worst results, reaching statistical significance, while its counterpart allowing degradation is the best performing approach. This can be explained by the fact that each island in IM-COMMA methods is subject to fewer generations in order to reach a similar number of fitness evaluations than the serial versions of COMMA. Building on the previous idea that COMMA approaches may suffer from a too slow exploitation of the good areas of the search space, $IM-COMMA$ is more prone to this problem than its serial counterpart which is allowed more generations. In this context, degradation may have a positive influence in making the exploration faster, reaching better areas of the search space earlier in the search.

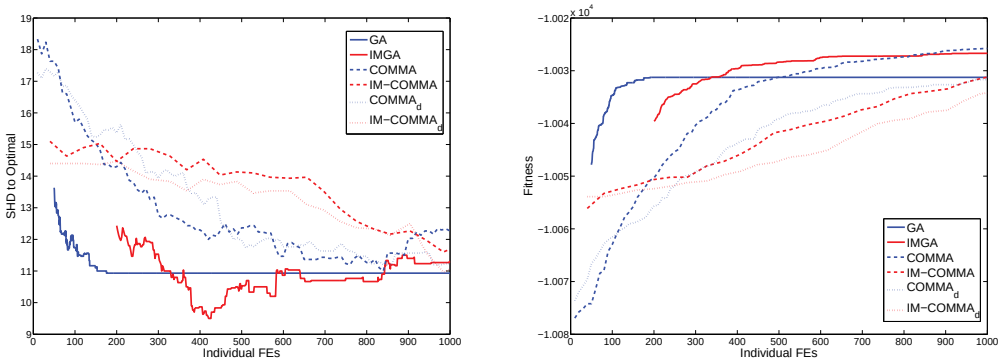
		Asia	Tank	Credit	Car	Boerlage	Alarm
K2GA	C	6.43 (0.56)	**14.77** (1.75)	8.87*(1.18)	10.83 (1.34)	17.43 (2.26)	23.50* (3.32)
	SHD	1.60 (0.66)	**10.93** (2.66)	4.70*(1.64)	13.33 (2.88)	24.80 (3.03)	48.60* (4.34)
	Rel.	**8.00** (0.00)	19.67*(0.54)	**12.00** (0.0)	14.27*(0.81)	28.07*(0.63)	41.30* (0.78)
	Err.	0.03*(0.18)	6.03*(1.52)	1.57 (0.56)	9.90*(2.71)	14.17*(2.44)	30.80* (3.87)
IMK2GA	C	6.57*(0.56)	14.20*(2.04)	9.23*(0.50)	12.13*(0.76)	**19.60** (2.14)	24.40* (3.41)
	SHD	1.47 (0.67)	11.30*(2.76)	3.80*(0.60)	11.57 (1.96)	**20.90** (3.23)	44.30* (5.16)
	Rel.	**8.00** (0.00)	19.83*(0.37)	**12.00** (0.00)	14.23*(0.76)	28.03*(0.48)	41.60* (0.80)
	Err.	0.03*(0.18)	**5.67** (1.07)	**1.03** (0.18)	9.47*(2.08)	**12.47** (2.16)	**27.10** (4.16)
COMMA	C	6.70*(0.46)	13.43*(1.87)	9.13*(1.23)	**12.47** (0.96)	18.70*(2.08)	24.20* (2.56)
	SHD	1.37 (0.66)	12.27*(2.64)	4.27*(1.71)	**10.93** (2.14)	22.87*(2.59)	46.90* (3.08)
	Rel.	**8.00** (0.00)	**19.87** (0.34)	**12.00** (0.00)	**14.33** (0.54)	**28.17** (0.64)	41.30* (0.46)
	Err.	0.07*(0.36)	5.83*(1.13)	1.40 (0.55)	**9.07** (1.67)	13.40*(1.65)	29.80* (1.66)
$IM-COMMA$	C	**6.90** (0.30)	14.17*(2.03)	**9.63**(1.20)	11.63*(1.17)	18.13*(2.32)	22.80* (3.37)
	SHD	**1.10** (0.30)	11.83*(2.78)	**3.60** (1.74)	12.80 (2.41)	23.53 (3.33)	52.10 (5.52)
	Rel.	**8.00** (0.00)	19.70*(0.46)	**12.00** (0.00)	13.87*(0.81)	28.07*(0.63)	41.00* (0.63)
	Err.	**0.00** (0.00)	6.30*(1.39)	1.23*(0.62)	10.57*(2.38)	13.60*(2.24)	33.90 (3.67)
$COMMA_d$	C	6.87*(0.34)	14.53*(2.06)	9.37*(1.28)	11.40 (1.11)	17.10 (1.78)	25.40* (3.07)
	SHD	1.13*(0.34)	11.23*(3.24)	4.10*(1.81)	12.80 (2.18)	24.93 (2.39)	44.80* (6.00)
	Rel.	**8.00** (0.00)	19.83*(0.37)	**12.00** (0.00)	14.20*(0.54)	27.83*(0.90)	**41.80** (0.60)
	Err.	**0.00** (0.00)	5.93*(1.59)	1.47 (0.62)	10.00*(1.73)	14.20*(2.27)	28.40* (3.98)
$IM-COMMA_d$	C	6.87*(0.34)	14.43*(2.46)	9.53*(1.33)	11.63 (1.08)	17.87*(2.31)	**26.10** (2.07)
	SHD	1.13*(0.34)	11.20*(3.69)	3.70*(1.93)	11.93*(2.62)	24.77 (2.92)	**43.70** (3.93)
	Rel.	**8.00** (0.00)	19.70*(0.46)	**12.00** (0.00)	14.17*(0.73)	28.03*(0.80)	41.60* (0.49)
	Err.	**0.00** (0.00)	5.93*(1.67)	1.23*(0.67)	9.40*(2.22)	14.60 (1.98)	28.20* (2.40)

Table 5.7: Characteristics of best BNs obtained by each algorithm after 1000 FEs



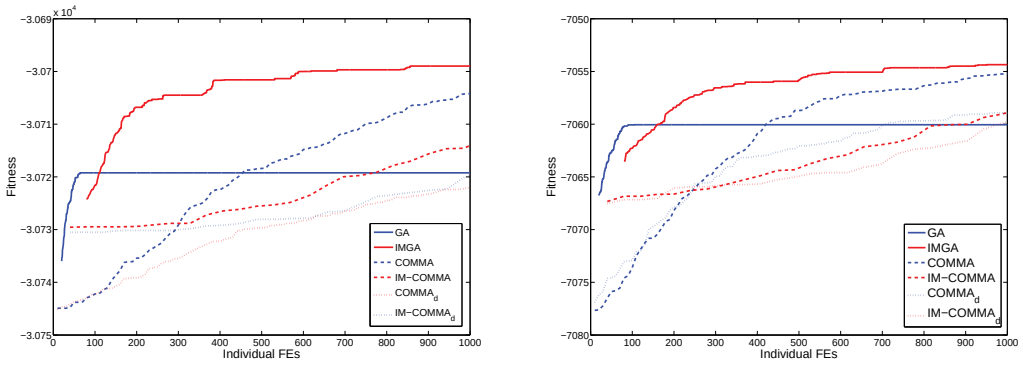
(a) $COMMA$, gen. 1 to 50 (b) $COMMA$, gen. 51 to 100 (c) $COMMA_d$, gen. 1 to 50 (d) $COMMA_d$, gen. 51 to 100

Figure 5.11: Evolution of agent's positions over time on *tank*



(a) SHD of best solution (b) Fitness of best solution

Figure 5.12: Evolution of SHD and fitness of the best solution over time on *tank*



(a) *boerlage* (b) *car*

Figure 5.13: Evolution of fitness of the best solution over time on *boerlage* and *alarm*

5.3.2.2 Comparison of distance-based mutation types within COMMA

In order to evaluate the performance of the different mutation types, they were used as operators within COMMA. 10 runs were performed on each dataset and the results averaged, including fitness, number of correct edges and SHD of the best solution found at the end of the runs. In the previous set of experiments, the COMMA-based implementations showed that they were still improving solutions when stopped. Thus, we set a higher limit on the total number of fitness evaluations and stopped the algorithms after 10000 of them. Because of the small sample size, Wilcoxon signed-rank test is used in order to evaluate differences observed in the results summarized in Table 5.8. Since, there are multiple comparisons, Bonferroni correction is applied. Thus, statistical significance was defined for p-values > 0.002.

Results show that the choice of mutation operator is important and affects the quality of the solutions that are obtained by COMMA. Since a large number of generations was used, we can consider that all implementations were given enough time to evolve good solutions.

Over all benchmarks, displacement and invert+swap mutations are the operator that lead to the best BNs. With respect to the suitability of the operators for optimization, that is considering only the fitness value, invert+swap is the best, exhibiting the best

fitness on all benchmark problems. On the other hand, insert mutation gives the worse results. The swap operator used in the previous section is mostly behind the best operators in terms of fitness. This is not always correlated with significant differences on BN quality.

In order to analyze these findings, it is important to refer to ρ, the expression of the level of disruption introduced by each operator as illustrated in Table 5.2. We recall that we can categorize the operators in two distinct kinds, those that introduce a fixed disruption for a given distance d, that is swap, insert and invert, and those whose disruption level takes its value according to some stochastic process, thus in a range of values. Out of the three operators that introduce a fixed level of disruption, two of them, swap and invert are clearly behind the best operators with respect to their results. The value of ρ for invert is the largest from all the operators with fixed ρ. On the other hand, insert, which produces good solutions with similar characteristics than those obtained by the best operators on *asia* and *car*, is the less disruptive operator. It seems that the invert operator is too disruptive to allow efficient optimization. According to Table 5.2, the scramble operator can be as disruptive as invert. However, since there is no fixed value for ρ, small ρ can be used, resulting in the generation of solutions of better quality than invert.

Invert+displacement, the most disruptive operator of all and the most disruptive of those that define ρ in a range of values exhibits results of lower quality than displacement and invert+swap, which may be caused by a level of disruption too high. This observation matches the earlier evidence that when large mutation distances are used, invert+displacement gets close to the results of random ordering generation.

The two best operators present different ρ values. Displacement usually gives a wide range of values to draw from and hence can sometimes be disruptive, while it may also not be. Invert+swap offers a range of values that is always centered on the same mean value, despite varying d. Increase of d affects the bounds and forces the operator to a ρ closer to the mean. Hence, this may result in having more disruption when d is small

than when d is high. Although showing very dissimilar strategies, both displacement and invert+swap mutation operators perform similarly and are the most adapted operators to be used in COMMA. Note that both invert+swap and invert+displacement use the invert operator as part of their mutation process. However, the resulting ρ values are very different and invert can be simply seen as a mechanism to get to these values.

5.4 Summary

- The neighborhood of a solution S defines the set of solutions that are close to S in terms of gene characteristics. On Euclidean spaces, neighborhood size is defined by the neighborhood radius. This idea can be extended to other representation spaces such as permutations. We defined a metric ρ that defines the level of dissimilarities between a solution S_1 and a solution S_2, obtained by mutating S_1. ρ represents the number of adjacent gene swaps that are needed to transform S_1 into S_2. Thus, ρ can be considered as a metric to define the extent of the neighborhood of a solution, given a mutation operator.

- Seven mutation operators are presented that uses a mutation distance d to control the degree of alteration that they each introduce to a solution. These are *swap*, *insert*, *invert*, *scramble*, *displacement*, *invert+swap* and *invert+displacement*. Each of them presents different values for ρ, that help concluding that *insert* is the less disruptive operator, while *invert + displacement* leads to the most disruptions.

- An algorithm is proposed that makes use of the distance mutation d to perform both exploration and exploitation of the search space throughout the search. COMMA uses a population of agents that are each assigned a solution. Agents are ranked according to the fitness of their respective solutions and mutation distances set accordingly. Agents with the best solutions are assigned small d,

	Swap	*Insert*	*Invert*	*Scramble*	*Displacement*	*Invert+Swap*	*Invert+Disp.*
				asia			
Fit.	-6827.97*	**-6827.96**	-6828.26*	**-6827.96**	**-6827.96**	**-6827.96**	**-6827.96**
C	6.9*	**7.0**	6.9*	**7.0**	**7.0**	**7.0**	**7.0**
SHD	1.1*	**1.0**	1.2*	**1.0**	**1.0**	**1.0**	**1.0**
				tank			
Fit.	-10024.92*	-10024.80*	-10026.35*	-10024.48*	-10024.81*	**-10024.27**	-10025.81*
C	13.2*	13.2*	13.9*	12.5*	13.5*	12.9*	**14.0**
SHD	12.5*	12.3*	12.0*	13.4*	11.5*	12.3*	**11.1**
				credit			
Fit.	-31762.57	-31758.16	-31760.18	-31755.55	**-31753.02**	**-31753.02**	-31753.33*
C	8.2	9.1*	9.1*	**9.7**	9.0*	9.0*	9.3*
SHD	5.8	4.1*	4.3*	**3.2**	4.0*	4.0*	3.7*
				car			
Fit.	-7053.35	**-7052.11**	-7054.77	-7052.17*	**-7052.11**	**-7052.11**	-7053.18
C	13.5*	**14.0**	12.9	13.8*	**14.0**	**14.0**	13.4
SHD	9.0*	**8.0**	10.2*	8.4*	**8.0**	**8.0**	9.1*
				boerlage			
Fit.	-30683.54	-30683.80	-30701.62	-30689.90	-30677.68*	**-30677.25**	-30680.80
C	18.7*	19.8*	19.0*	19.0*	**21.1**	20.7*	20.0*
SHD	20.8*	20.2*	22.4*	21.5*	**18.3**	19.2*	19.6*
				alarm			
Fit.	-29627.38	-29639.28	-29784.05	-29692.44	-29578.19*	**-29556.86**	-29634.64
C	29.5*	28.7*	23.9	28.0*	**30.4**	29.5*	28.4*
SHD	34.9*	34.7*	44.4	38.1*	**31.3**	32.1*	35.3*

Table 5.8: Performance comparison of COMMA implementations with different distance-based mutation operators

likely to bring less alteration to their solutions, while the worst agents are set with large d. As the search progresses, agents improve their solutions and compete for a place in the top of the population, where less alteration is introduced. The use of Island Model and the allowance of degradation in COMMA are also investigated.

- COMMA shows that it can produce solutions of similar quality than those obtained by K2GA and IMK2GA. However, the improvement of the solutions is slower with COMMA which has not converged by the end of the runs. This latter observation suggests that allowing more runtime may lead to better solutions. Studying the positions of agents and how they evolve over time shows that the competing strategy introduced in COMMA supports the search. The use of IM does not bring improvement to the quality of the solutions that are produced. On the other hand, introducing degradation can help in finding good solutions as seen on *alarm*.

- Several mutation operators are assessed within the COMMA framework. *Displacement* and $invert + swap$, a combination of two operators obtained from literature, exhibit the best results overall both in terms of CH score and BN structure quality, showing that they are the operators the best adapted to COMMA for BN structure learning.

Chapter 6

Applications to Prostate Cancer Staging

In the present chapter, the potential use of BNs is investigated in a medical environment [1]. More precisely, it considers learning BNs to model data related to prostate cancer staging, based on two real-world datasets respectively gathered by the British Association of Urological Surgeons (BAUS) and the urology unit from Aberdeen Royal Infirmary (ARI). Four main points are treated here. First, the process of prostate cancer staging is described, along with the related current clinical practice. Second, the study is justified by comparing BNs to other classifiers with respect to their ability in predicting cancer stages from patient data. In the third part of this chapter, we apply techniques defined in the previous chapters to the two medical datasets at hand. Finally, the use of IM is explored to make use of distributed medical data, which represents a barrier to many data modelling studies in the field.

[1] Large parts of this chapter are extracted from material published in O. Regnier-Coudert, J. McCall, R. Lothian, T. Lam, S. McClinton, J. N'Dow. Machine learning for improved pathological staging of prostate cancer: A performance comparison on a range of classifiers. Artificial Intelligence in Medicine, 55(1):25-35. 2011. It is used with permission of the publisher.

6.1 Prostate Cancer Staging

6.1.1 Medical Background

Cancer is a disease where malignant cells are developed and alter the function of their hosting organs or tissues. Typically, malignant cells reproduce and group together to form a tumor. Untreated tumors grow and affect surrounding healthy cells, leading to a spread of the cancer. Metastasis happens when the cancer reaches surrounding organs or tissues. The presence of cancer results in the deterioration of some body functions and can lead to death when vital organs are touched.

Whilst in the past prostate cancer was a disease which predominantly affected older men well into their seventies, the advent of Prostate Specific Antigen (PSA) testing over the past three decades has caused a shift in the age of presentation, such that men in their early fifties are increasingly being diagnosed. In addition, PSA testing has also resulted in a stage migration from late, symptomatic stages to early, asymptomatic stages of the disease. Men with raised PSA would then undergo prostate biopsies which will confirm the diagnosis and provide a grade of the disease expressed by means of the Gleason Sum score (GS), with the grade of the cancer reflecting its aggressiveness. Once the diagnosis is confirmed, a Digital Rectal Examination (DRE) is performed to assess the local Clinical Stage (CS). The stage is a means of indicating the spread of the disease, expressed by the TNM staging system (Sobin, 2009), whereby the T stage refers to the local extent of spread. The treatment options available for localised prostate cancer include surgery, external beam radiotherapy, brachytherapy, active monitoring and minimally invasive localised therapy such as cryotherapy (Kirby, 2002). Surgery by way of radical prostatectomy, where the prostate is surgically removed completely, is one of the leading options. Although most of the curative treatment options result in similar cure rates, surgery has the major advantage of removing the prostate completely as well as providing the actual pathological stage and grade of the disease, which in turn influence prognosis. The pathological stage is the most accurate determination of

the actual stage of the disease (as opposed to clinical stage which is an estimate), being determined by pathological examination of the entire prostate specimen. However, the major drawback of surgery is its associated adverse effects, such as intra-operative complications (e.g. bleeding), prolonged hospital stay, urinary incontinence and erectile dysfunction. The pathological stage of prostate cancer significantly influences the prognosis; the presence of extra-prostatic extension reduces the chance of cure and increases the risk of adverse effects. Consequently, surgery may not be appropriate for every man with prostate cancer, and those with more advanced disease should be offered other options instead. Such decision-making crucially relies on the prediction of the likely pathological stage. It is for this purpose that predictive staging tools were created.

6.1.2 Predictive Staging Tools

Partin tables (Partin et al., 1993) are the most commonly used tool for prostate cancer staging. The tables are a means of predicting the likely pathological stage of the cancer using the pre-treatment variables of PSA, GS and CS, with the result being expressed as probabilities. Based on a patient's PSA, GS and CS, probabilities are provided for each of four discrete pathological stage outcomes: Organ Confined (OC), Extra-Prostatic Extension (EPE), Seminal Vesicle Involvement (SVI) and Lymph Node Involvement (LNI). The predicted probabilities of pathological outcomes are displayed by means of look-up tables organised according to the three pre-treatment variables, which are in turn divided into sub-groups.

Partin tables were originally created using Logistic Regression (LR) (Hosmer and Lemeshow, 2000) on a database gathering records of patients that were treated with radical prostatectomy in a single US institution (Partin et al., 1993). Since then, the tables have been updated using different up-to-date datasets (Partin et al., 1997, 2001; Makarov et al., 2007). The revision takes into account changes in population demographics, advances in health technology and improved health care systems, but the

tables are still based on the same fundamental LR-based methodology.

Partin tables are the most well-established and most widely used pathological staging tool in the urological community worldwide. However, concerns have been raised regarding their validity on non-US populations as such populations may present different characteristics (Kattan et al., 1997; Blute et al., 2000; Penson et al., 2002; Graefen et al., 2003; Augustin et al., 2004; Eskicorapci et al., 2005; Karakiewicz et al., 2005; Song et al., 2005; Gao et al., 2008; Bhojani, Ahyai, Graefen, Capitanio, Suardi, Shariat, Jeldres, Erbersdobler, Schlomm, Haese et al., 2009; Bhojani, Salomon, Capitanio, Suardi, Shariat, Jeldres, Zini, Pharand, Péloquin, Arjane et al., 2009). In some instances, Partin tables were considered to be unsuitable for the target population because of limitations with respect to their predictive power (Kattan et al., 1997; Penson et al., 2002; Song et al., 2005; Gao et al., 2008; Bhojani, Ahyai, Graefen, Capitanio, Suardi, Shariat, Jeldres, Erbersdobler, Schlomm, Haese et al., 2009; Bhojani, Salomon, Capitanio, Suardi, Shariat, Jeldres, Zini, Pharand, Péloquin, Arjane et al., 2009). The appropriateness of the methodology behind Partin tables, especially in regard to the choice of predictive variables and classifier, was not addressed in those studies. In addition, it is widely recognized that prostate cancer staging is associated with a high level of uncertainty. All these considerations are compelling clinicians to explore alternative means of generating predictive tools, especially those which apply machine-learning techniques which have the potential of improving the quality and accuracy of predictive performance (Cruz and Wishart, 2006).

This issue was addressed in the UK by constructing a predictive table derived from a UK-wide surgical cohort (Bott et al., 2008). In that study, pre-operative clinical stage was not included in the final analysis, and percentage core biopsy was used as a surrogate instead.

Variable Name	Categories
PSA	0-2.5, 2.6-4.0, 4.1-6.0, 6.1-10.0, >10.0
GS	5-6, 3+4, 4+3, ≥8
CS	T1c, T2a, T2b/c
PS	OC, EPE, SVI, LNI

Table 6.1: Variable details of the BAUS-4 dataset

6.1.3 Data

6.1.3.1 BAUS Data

BAUS gathered clinical and pathological data on over 7500 patients that were received with prostate cancer and underwent radical prostatectomy in one of the 57 different centers of the study between 1999 and 2008. This accounts for approximately 20% of the total number of prostatectomies that were performed in the whole of UK over this period (Shaida and Malone, 2007). The BAUS dataset can be considered as large and representative of the British population and consequently, well suited for the assessment of Partin tables for use in the UK.

From the original BAUS dataset, two subsets were created to meet the different objectives. To construct the first one, we only kept the records where PSA, GS, CS and PS were set in order to match Partin tables variable settings. We call this dataset BAUS-4. Each variable in BAUS-4 was discretized following Partin method as described in Table 6.1. The final size of BAUS-4 was 1701 records, following the removal of cases where data was missing for any of the four variables and where input data was erroneous. The distribution of PS in the prepared data was compared with its distribution in the original BAUS dataset. No important differences were noticed and we assumed that the data remained unbiased after the preparation process.

The original BAUS dataset was used to create a second subset called BAUS-6. In BAUS-6, the two variables *age* and ASA are added to BAUS-4. ASA is a score which describes the severity of a patient's symptoms on a scale ranging from 1 to 5. No

Variable Name	Categories
PSA	0-2.5, 2.6-4.0, 4.1-6.0, 6.1-10.0, >10.0
GS	5-6, 3+4, 4+3, ≥8
CS	T1c, T2a, T2b/c
Age	<55, 55-59, 60-64, 65-69, ≥70
ASA	1, 2, 3
PS	OC, EPE, SVI, LNI

Table 6.2: Variable details of the BAUS-6 dataset

patient was received with an *ASA* of 4 or 5 and only three categories were kept for this variable. *Age* was discretized in five categories that were chosen as to ensure a balanced distribution between them, as described in Table 6.2. BAUS-6 contains 1535 records after preparation.

6.1.3.2 ARI Data

A further dataset, ARI-10, was prepared from data collected at the Aberdeen Royal Infirmary, UK. This data contains different variables and allows the exploration of variables that were not collected in the original BAUS dataset. Table 6.3 presents the variables that were selected for our study. In addition to the Partin variables, ARI-10 includes information on patients' age, erectile function, prostate size following transrectal ultrasound (*TRUS size*) and stage prediction following Magnetic Resonance Imaging (*MRI stage*). Two variables are also included that relate to the patient's well-being. They both result from the International Prostate Symptom Score (IPSS) which is composed of seven questions related to the effect of the symptoms on the patients and an additional question which reflects his overall Quality of Life (QoL). We respectively call these variables *IPSS symptoms* and *IPSS QoL*. In clinical practice, *IPSS symptoms* and *IPSS QoL* are used as a means of comparing the effect of treatment on a patient. However, they can also be an indicator of other variables such as prostate size for example which can lead to procedure complications and be taken into account when selecting the best treatment option. Being based on patients from a single institution, the size of ARI-10 is much smaller and contains 85 records. Such small size implies that

Variable Name	Categories
PSA	0-2.5, 2.6-4.0, 4.1-6.0, 6.1-10.0, >10.0
GS	5-6, 3+4, 4+3, ≥8
CS	T1c, T2a, T2b/c
Age	<55, 55-59, 60-64, 65-69, ≥70
Pre-op erection	Full function, Partial, Absent, Unknown
IPSS Symptoms	Mild, Moderate, Severe, Unknown
IPSS QoL	0, 1, 2, 3, 4, 5, 6, Unknown
TRUS size	0-30, 31-60, >61, Unknown
MRI stage	T0/T1, T2, T3a, None
PS	OC, EPE, SVI, LNI

Table 6.3: Variable details of the ARI-10 dataset

the variance of classifiers built from this data is likely to be important (Brain et al., 1999) and results should be considered as preliminary.

6.1.3.3 Data Distribution

One important characteristic of both BAUS and ARI data relies in the distribution of the class variable being very skewed towards milder pathological stages. By consequence, the number of SVI and LNI cases is low in each dataset as illustrated in Figure 6.1.

6.2 Bayesian Networks for Prostate Cancer Staging

This section introduces three main objectives that present interests for both medicine and machine learning communities. Overall these experiments help to establish the potential that BNs have with respect to applications on prostate cancer data.

First, we aim to critically assess the methodology which was used to construct Partin tables. This involves externally validating the version currently being used by practitioners, that is, studying how well it performs on a population that presents different characteristics. Here, the tool is evaluated on a large British cohort and results are

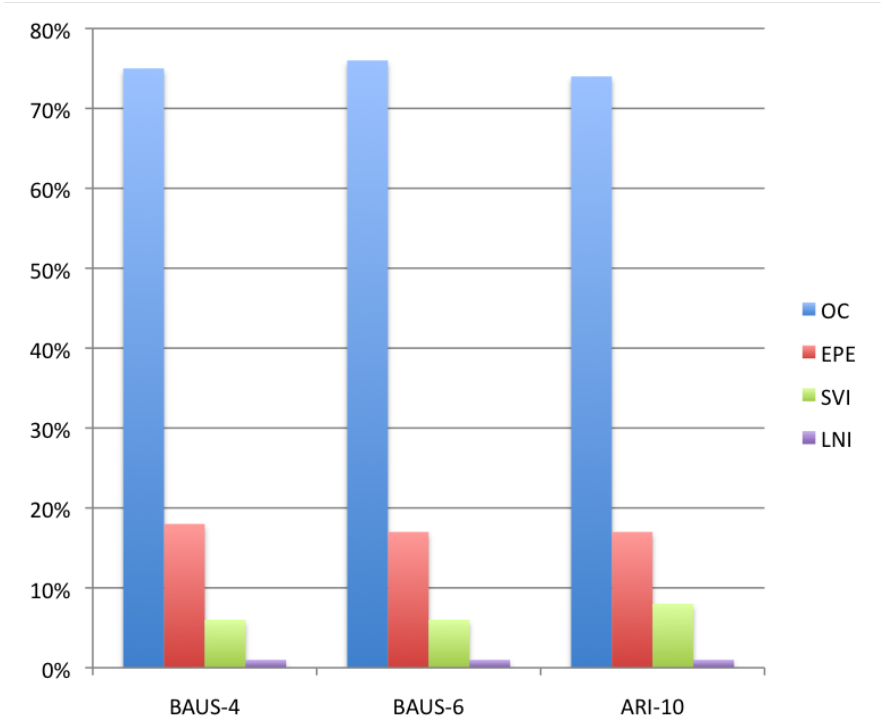


Figure 6.1: Distribution of the class variable pathological stage in the three datasets

compared to those of its internal validation given in (Makarov et al., 2007), where the original data was also used for testing. Using the British data and the approach described in (Makarov et al., 2007), we build new lookup tables and assess the methodology itself. The results are compared against the previous validation studies (Kattan et al., 1997; Blute et al., 2000; Penson et al., 2002; Graefen et al., 2003; Augustin et al., 2004; Eskicorapci et al., 2005; Karakiewicz et al., 2005; Song et al., 2005; Gao et al., 2008; Bhojani, Ahyai, Graefen, Capitanio, Suardi, Shariat, Jeldres, Erbersdobler, Schlomm, Haese et al., 2009; Bhojani, Salomon, Capitanio, Suardi, Shariat, Jeldres, Zini, Pharand, Péloquin, Arjane et al., 2009) and provide additional understanding on Partin tables performances.

Second, we propose alternative classifying techniques to build lookup tables for prostate cancer staging. We run many classifiers, including LR, on our data and study the performances of the models produced by each. We compare the different methods with respects to their predictive power and propose alternatives to LR.

Finally, we investigate the impact of new variables being introduced into the model. Two different datasets are used for this purpose each using distinct set of predicting variables. Among them, variables that were originally excluded when Partin tables were built are considered. In (Partin et al., 1993), patient's age was tested against other combination of variables and did not show statistically significant improvement to the LR-based model. A range of classifiers is considered and applied to different subsets of data in order to observe the impact that inclusion of elements can have.

6.2.1 Overview of Alternative Classifiers

In order to fairly compare LR to other methods, the selection of these techniques represents an important step of the study. A first set of runs was performed in order to select the best classifiers from the Weka platform (Hall et al., 2009). In this section, we present the methods that showed the best initial performances and that were applied to the different datasets for the complete study. In order to ensure the comparison covers a wide area of the machine learning landscape, we considered methods from the following machine learning families: decision tree learning, lazy learning, regression, Support Vector Machine (SVM), Artificial Neural Networks (ANNs) and BNs. Most of these classifiers have already been applied to cancer applications in the past as summarized in (Cruz and Wishart, 2006).

Random forest (RF) (Breiman, 2001) is an ensemble method in which several decision trees are constructed. Each tree is built using a subset of the input variables chosen at random. The final outcome of the classifier is the class which is predicted by the highest number of individual trees.

In k-nearest neighbours (k-NN) (Aha et al., 1991), a new instance is classified according to the class value of its k most similar neighbours. A majority vote is used to infer a classification outcome from the k retrieved values. The distance between the test instance and the training instances can be computed in several ways. The most popular method for numerical attributes uses Euclidean distance. This has also been adapted

to handle nominal variables, the distance between two instances corresponding to the number of attributes they have in common.

LR (Hosmer and Lemeshow, 2000) associates a weight to each of the predictors (for binary variables) or to each of the predictor states (for multinomial variables). Weighted predictor observations are summed and fitted to a logistic curve to produce a probability for the response variable. As previously mentioned, LR is the technique which was used to generate the Partin tables and represents therefore an important element of comparison.

In SVM, instances are represented as vectors and projected onto a n-dimension graph where n is the number of features in the dataset. Building such a classifier requires finding the optimal hyperplane that splits instances in clusters according to their class values. For the present study, the Sequential Minimal Optimization algorithm (SMO) (Platt, 1999) was used to train the SVM.

ANNs consist of two or more layers of artificial neurons which receive information signals via their respective inputs. The value of the input information is weighted and processed by a neuron according to the value of its activation threshold. Using many layers of linked neurons, complex decision process can be modeled. Different approaches have been developed to learn the neuron weights or to set the activation functions. From the wide range of available ANNs, and in order to cover more than one approach, Multilayer Perceptron (MLP) (Rumelhart et al., 1985) and Radial Basis Function (RBF) (Broomhead et al., 1988) were selected for the study. MLP was applied for prostate cancer staging in previous studies and presented better performances than LR (Han et al., 2001; Matsui et al., 2002; Veltri et al., 2002). However, it has also been proven that MLP does not always outperform LR (Borque et al., 2001; Anagnostou et al., 2003; Kawakami et al., 2008). RBF was compared against LR in (Borque et al., 2001) but no significant difference was observed between the methods.

Finally, although BNs do not need to be described in this section, restrictions can also be applied on them that result in simplified models often preferred to general BNs on

classification tasks. In a Naive Bayes (NB) (Langley et al., 1992), a class variable is set prior to build the model. The BN which is created afterwards defines the class variable as a parent of all other variables. In a NB, no edges are allowed between the predictor variables. The concept of NB has been extended as to consider that relations may exist between the predictors. Such BN is called a Tree Augmented Naive Bayes (TAN) (Friedman et al., 1997) and is built using a greedy search and the CH metric.

6.2.2 Experimental Design

The main objective of the study is to evaluate the performance of the different classifiers. Area under the ROC curve (AUC) is a standard method to assess a model's predictive power (Metz, 1978). AUC takes into consideration both sensitivity and specificity and represents an objective way to cope with data which is unbalanced between classes. An AUC value close to 1 describes a model with a good predictive power, while a value close to 0.5 shows that the model is no better than a random decision. An AUC of zero describes a model that classifies all instances with a wrong label.

On the other hand, many medical studies have assessed Partin tables by computing their concordance index (c-index) (Harrell Jr et al., 1982), as an alternative to AUC. C-index reflects how good a model is at accurately distinguishing between two randomly selected patients with different outcomes. Algorithm 7 outlines how the concordance index CI_{c_i,c_j} between two distinct classes c_i and c_j is computed for a given model M. Let s_p and s_q respectively denote the class label of subjects p and q. s_p and s_q belong to the ordered set C of n class labels such as $c_1 < c_2 < ... < c_n$. Let S_i denotes the set of subjects p with class label c_i such as $S_i = \{p : s_p = c_i\}$ and $m_i = |S_i|$.

$P(s_p = c_i)$ represents the probability of subject p to be classified as c_i while $P(s_p > c_i)$ denotes the probability of subject p to be classified with a class label better than c_i. Similarly, $P(s_p < c_i)$ denotes the probability of subject p to be classified with a class label worse than c_i.

Algorithm 7: Concordance Index CI_{c_i,c_j} between classes c_i and c_j

Initialize $correct = 0$
for each pair $(s_p, s_q) \in S_i \times S_j$, $(i < j)$ **do**
  from model M, compute $P(s_p < s_q) = \sum_{k=1}^{n-1} P(s_p = c_k) * P(s_q > c_k)$ and $P(s_q < s_p) = 1 - P(s_p < s_q)$
  if $P(s_p < s_q) > P(s_q < s_p)$ **then**
    $correct$ ++
  end if
end for
return $CI_{c_i,c_j} = correct \ / \ m_i * m_j$

In order to validate the Partin tables and their methodology, the approach that was used to build them (Makarov et al., 2007) was carefully studied and replicated. As a result, multinomial LR was applied to the data using bootstrap resampling with 1000 replications. The variables were discretized in the same way as in Partin tables and as given in Table 6.1. C-indices for each pathological stage against OC were computed as to ensure the results can be compared with previous external validations and with original findings.

For the rest of the study, 10-fold stratified cross validation was performed 1000 times for each classifier on the three datasets and AUC was calculated. The number of folds and the choice for stratification was decided following (Kohavi, 1995) to ensure the measure of accuracy reflects objectively the model's true abilities with respect to its variance and bias when data varies.

To assess RF, k-NN, LR, MLP, RBF and the SVM classifiers, we used the Weka suite (Hall et al., 2009), while BN-based models were built and analyzed using implementations of K2GA developed at the Robert Gordon University and presented in Chapter 2. To ensure the performance measures were consistent across the two tools, the random seeds for stratification and cross validation was set to the same value. We compared several BNs using both platforms and retrieved similar AUC values.

An initial 10-fold cross validation analysis was run for each of the Weka classifiers in order to ensure that comparisons are fair. The parameter settings were hand optimized

Classifier	Settings
k-NN	k = 3 (BAUS-4); k = 4 (BAUS-6, ARI-10) Search = linear based on Euclidean distance
RF	maxDepth = unlimited numFeatures = 3 (BAUS-4, BAUS-6); numFeatures = 4 (ARI-10) numTrees = 10
LR	maxIts = -1 ridge = 10^{-8}
MLP	Hidden layers = 4 (number of classes) Learning rate = 0.3 Momentum = 0.2 Training time = 100 epochs
RBF	maxIts = -1 minStdDev = 0.1 numClusters = 2 ridge = 10^{-8}
SVM	Filter type = Normalize training data Complexity parameter = 1 Kernel type = Puk ($\Omega = 1$; $\Sigma = 1$)

Table 6.4: Classifier settings in Weka

until no further improvement with respect to the AUC could be found. We regard these settings as optimum and kept them for the study. The final settings are presented in Table 6.4. K2GA, the algorithm used to search for an optimum BN was tuned with the settings presented in Table 6.5. K2GA and the greedy search for TAN were run on the complete datasets, providing an optimum structure for each. The parameters were calculated afterwards for each testing fold. We call the BN learned from K2GA, CHBN, as it is based on the CH metric. For BAUS-4, CHBN was found exhaustively because it was possible to test all possible structures, due to the small number of variables in the data.

6.2.3 Experimental Results

Results for this set of experiments were dealt with and discussed in relation with the three main themes that are the external validation of the Partin tables, the choice of the methodology and finally the importance of the data.

GA parameter	Value
Number of runs	20
Population size	100
Selection type	Rank selection
Crossover type	Cycle crossover
Crossover rate	0.9
Mutation rate	0.1

Table 6.5: Settings for K2GA

	Partin tables with US Data (Makarov et al., 2007)	Partin tables with BAUS Data	Multinomial LR with BAUS Data
EPE vs. OC	0.696	0.602	0.610
SVI vs. OC	0.830	0.709	0.713
LNI vs. OC	0.894	0.819	0.873

Table 6.6: Concordance index of the different LR models

6.2.3.1 External validation of Partin tables

Similarly to the internal validation of the Partin tables, c-index was calculated for the three non-OC pathological stages vs. OC. Results are presented in Table 6.6 and illustrate how good the different LR models are at distinguishing between patients with each combination of stages. Such values can be understood relative to the scale given in (Galfano et al., 2008). The scale defines three levels of predictive power for a model according to its c-index. A model has low, moderate or high prognostic accuracy if its c-index is respectively between 0.5 and 0.7; between 0.7 and 0.9; or greater than 0.9.

Referring to Table 6.6, any model built following Partin approach is found to have a low predictive power when distinguishing between OC and EPE cases, regardless of the dataset used for validation. Internal validation numbers given in (Makarov et al., 2007), show that the Partin tables predictive power is moderate for SVI vs. OC and LNI vs. OC cases. When applied to British data, we notice a drop in terms of c-index for every combination of outcomes. Building a new model using the same methodology improves slightly the c-index but both Partin tables and new LR model can be described with

a low predictive power for EPE vs. OC and a moderate predictive power for SVI vs. OC and LNI vs. OC cases.

As shown in Table 6.6, original Partin tables achieve lower c-indices when applied to the BAUS data. This implies that when used on a UK population, Partin tables have a lower predictive power than on the native data from which they were derived. With a c-index below 0.70 for OC vs. EPE, Partin tables can be considered as having poor predictive power for patients falling in these two categories. We recall that patients with OC and EPE pathological stages are the most frequent cases in the BAUS dataset, as they count for nearly 95% of the entire cohort. In addition, the decision making for radical prostatectomy is strongly guided by the probabilities of a patient to have OC or EPE conditions. Correct distinction between these two classes represents thus the most important feature of the Partin tables. Furthermore, the applicability and usefulness of predictive tools with a c-index lower than 0.70 has been questioned, although there is a lack of an accepted reference threshold beyond which the use of a predictive model becomes unacceptable (Kattan, 2006).

The AUC analysis also supports this assertion and is in keeping with the trend seen in other validation studies (Kattan et al., 1997; Blute et al., 2000; Penson et al., 2002; Graefen et al., 2003; Augustin et al., 2004; Eskicorapci et al., 2005; Karakiewicz et al., 2005; Song et al., 2005; Gao et al., 2008; Bhojani, Ahyai, Graefen, Capitanio, Suardi, Shariat, Jeldres, Erbersdobler, Schlomm, Haese et al., 2009; Bhojani, Salomon, Capitanio, Suardi, Shariat, Jeldres, Zini, Pharand, Péloquin, Arjane et al., 2009). In these studies, the AUC for OC varies between 0.604 (Gao et al., 2008) and 0.817 (Graefen et al., 2003).

There are several possible explanations for the reduced performance of Partin tables when applied to a non-US population. Firstly, the original c-indices for Partin tables were derived from internal validation, indicating that the same data was used to both generate and assess the tables. The resulting model should thus be very well suited for the population from which it was derived but not necessarily for different populations.

Secondly, the updated Partin tables (Makarov et al., 2007) were generated using a cohort of patients from a single institution. The tables reflect therefore the local population's demographics, genetic and ethnic mix, disease trends, environmental and social factors and health care system, and therefore may not perform as well on populations with different characteristics. The BAUS cohort presents some fundamental differences likely to affect the models predictive power. These differences may be due to significant discrepancies in health care system policies between the US and the UK. For instance, PSA screening is routinely practised in the US, while it is not the case in the UK, and clearly UK patients undergoing prostatectomy are presenting with a higher PSA, GS or clinical stage. Lymph node dissection is also a domain that differs between UK and US policies. While in the US, it is systematically performed on patients undergoing prostatectomy, in the UK, it is reserved for patients presenting high PSA and high GS. The lower rate of dissection in the UK may account for the differences in c-indices and AUC for LNI in the different models, as this pathological stage is only observed following dissection. Finally, the poor performance may also be due to the categorization of the pre-operative variables. As has already been shown, the disease characteristics and trends differ between the two populations and as such the original Partin sub-grouping of variables may not be entirely appropriate for the BAUS cohort.

Despite the overall low predictive power that can be associated with all models built following Partin approach, UK derived lookup tables show a better c-index. This observation supports our assertion that when applied locally, lookup tables generated from a UK population may have higher predictive power than those generated from a population with inherently different characteristics.

6.2.3.2 Use of alternative classifiers and addition of new variables

Using the BAUS-4 dataset, we evaluated a range of classifiers. Table 6.7 describes the AUC of the different classifiers for each pathological stage. Each of these values illustrates how good the model is at correctly classifying a new patient in the given

Classifier	AUC (OC)	AUC (EPE)	AUC (SVI)	AUC (LNI)
NB	**0.662** (0.002)	**0.604** (0.003)	**0.702** (0.004)	**0.827** (0.012)
TAN	0.654 (0.003)	0.588 (0.005)	0.701 (0.007)	0.794 (0.015)
CHBN	0.630 (0.003)	0.578 (0.006)	0.693 (0.005)	0.809 (0.014)
LR	0.660 (0.002)	0.601 (0.004)	0.694 (0.004)	0.717 (0.036)
MLP	0.645 (0.006)	0.587 (0.008)	0.693 (0.012)	0.792 (0.031)
RBF	0.649 (0.006)	0.591 (0.009)	0.686 (0.012)	0.767 (0.046)
k-NN	0.632 (0.005)	0.569 (0.008)	0.666 (0.012)	0.700 (0.014)
RF	0.633 (0.006)	0.569 (0.008)	0.660 (0.013)	0.465 (0.019)
SVM	0.525 (0.003)	0.492 (0.004)	0.585 (0.009)	0.491 (0.001)

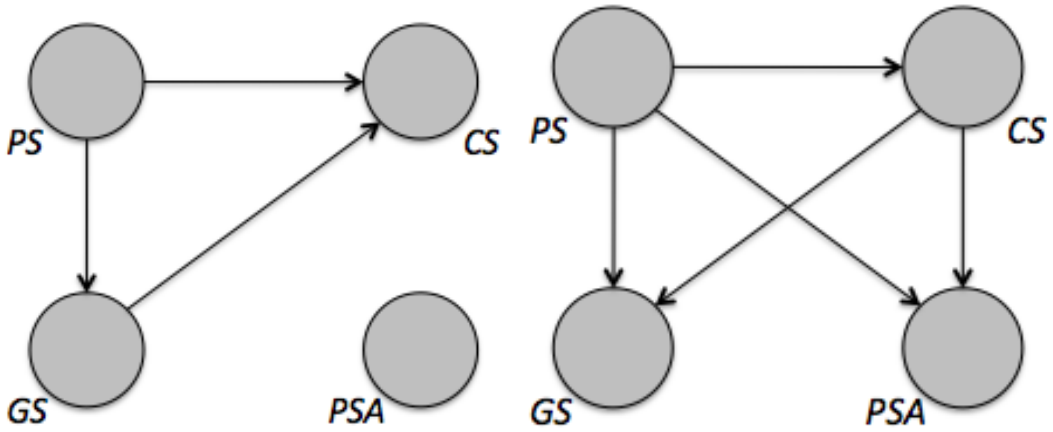
Table 6.7: BAUS-4 : Performances of the different classifiers

category. Overall, the use of different methods gives rise to the variety of AUCs that are calculated.

The model built using NB offers the best AUC in any pathological stage. Bonferroni correction was applied to ensure a fair comparison between methods and the difference between NB and the other classifiers is found to be statistically significant (p-value$<$0.005).

Among models based on BN, CHBN and TAN do not offer any advantage over the simpler NB. It is interesting to note that, despite its performance, the naive structure is not found by the exhaustive search used in CHBN nor by the greedy search for TAN. Figure 6.2 illustrates the structures that were learned by the GA and by the TAN search algorithm. These are based on the CH metric and reflect relationships within the data. The structure showing the highest CH score does model relationships between PS, GS and CS but considers PSA as conditionally independent from the other variables. In the TAN, in addition to the naive structure, PSA is considered as conditionally dependent on CS. The latter model outperforms CHBN to a statistically significant extent for the prediction of OC, EPE and SVI cases.

The BAUS-6 dataset introduces two extra variables that were not in BAUS-4. The same classifiers were run and AUC values are described in Table 6.8. With the probability distribution becoming harder to model, no classifier shows the best AUC on all four



(a) K2GA (b) TAN

Figure 6.2: BN structures learned from BAUS-4 dataset

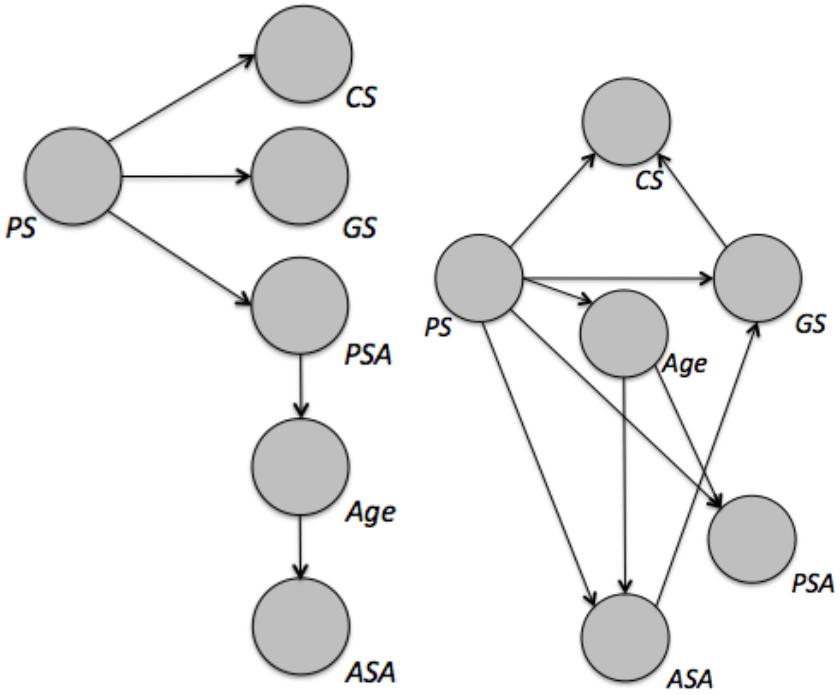
Classifier	AUC (OC)	AUC (EPE)	AUC (SVI)	AUC (LNI)
NB	**0.679** (0.002)	0.620 (0.004)	0.713 (0.005)	0.740 (0.007)
TAN	0.668 (0.004)	0.600 (0.006)	**0.735** (0.008)	0.627 (0.008)
CHBN	0.675 (0.002)	**0.622** (0.003)	0.724 (0.004)	**0.773** (0.006)
LR	0.675 (0.003)	0.615 (0.005)	0.699 (0.006)	0.731 (0.015)
MLP	0.650 (0.009)	0.597 (0.011)	0.694 (0.017)	0.746 (0.040)
RBF	0.656 (0.009)	0.599 (0.011)	0.692 (0.015)	0.648 (0.079)
k-NN	0.627 (0.007)	0.560 (0.009)	0.665 (0.012)	0.522 (0.049)
RF	0.604 (0.009)	0.540 (0.010)	0.616 (0.016)	0.470 (0.009)
SVM	0.516 (0.004)	0.496 (0.006)	0.538 (0.010)	0.493 (0.001)

Table 6.8: BAUS-6 : Performances of the different classifiers

values of PS, as was the case on BAUS-4. Despite this difference, all best models are built using different BN techniques.

With respect to the assessment of LR for prostate cancer staging, both NB and CHBN significantly outperform LR for the prediction of all pathological stages, except for OC classification where LR yields an AUC not significantly different to that of CHBN.

The BN structures learned using K2GA and TAN search are presented in Figure 6.3. K2GA retrieves the naive structure that exists between CS, GS, PSA and the class variable PS. In addition, relationships are discovered between PSA and *age*; and *age* and ASA. These two conditional dependencies are also found by the TAN.

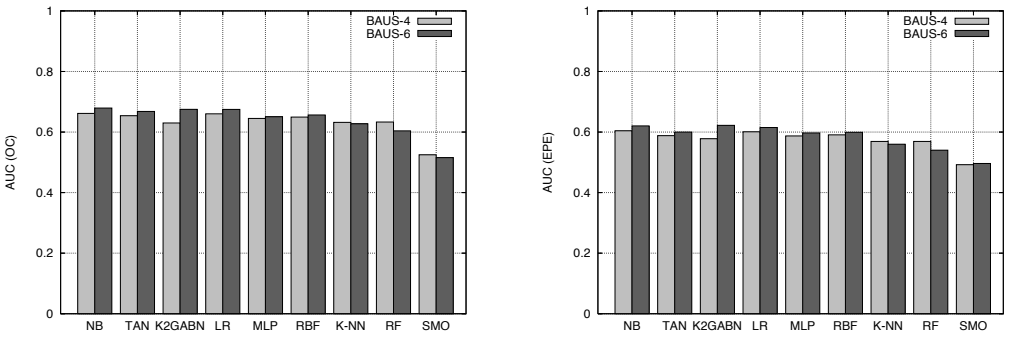


(a) K2GA (b) TAN

Figure 6.3: BN structures learned from BAUS-6 dataset

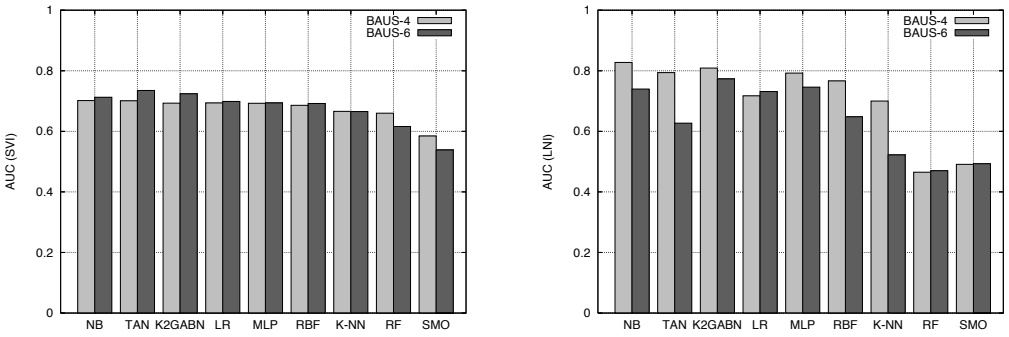
Along with Tables 6.8 and 6.7, Figure 6.4 illustrates the impact of adding features to the set of variables originally used in Partin tables. Performances of each classifier can be compared across the two datasets. LR-based models are significantly improved by addition of the *age* and *ASA* variables in all four categories. BN techniques and RBF present better AUC on BAUS-6 than on BAUS-4 for OC, EPE and SVI predictions, while MLP reaches a statistical significant level of difference for OC and EPE predictions. The k-NN and RF methods suffer from the addition of variables. As a consequence, their AUC values decrease for all of the *PS* categories but one, as LNI prediction is improved for RF. LNI prediction is, over all classifiers, altered negatively by the inclusion of *age* and *ASA* in the study, except if combined with the use of LR or RF.

The ARI-10 dataset, prepared from a smaller number of records, but with more variables is used to explore the behavior of the classifiers and possible relationships among variables. AUC results are shown in Table 6.9 for our selection of classifiers. The range



(a) OC

(b) EPE



(c) SVI

(d) LNI

Figure 6.4: Difference in AUCs between BAUS-4 and BAUS-6.

Classifier	AUC (OC)	AUC (EPE)	AUC (SVI)	AUC (LNI)
NB	0.523 (0.036)	0.410 (0.051)	0.528 (0.030)	0.008 (0.007)
TAN	0.592 (0.032)	**0.639 (0.046)**	0.488 (0.029)	0.011 (0.010)
CHBN	**0.668 (0.026)**	0.591 (0.045)	0.567 (0.029)	0.019 (0.008)
LR	0.534 (0.048)	0.342 (0.059)	0.582 (0.081)	0.380 (0.237)
MLP	0.500 (0.055)	0.379 (0.067)	**0.668 (0.075)**	**0.670 (0.232)**
RBF	0.490 (0.071)	0.487 (0.067)	0.475 (0.116)	0.546 (0.294)
k-NN	0.604 (0.041)	0.592 (0.047)	0.400 (0.067)	0.329 (0.172)
RF	0.512 (0.059)	0.455 (0.071)	0.545 (0.088)	0.500 (0.000)
SVM	0.500 (0.000)	0.500 (0.000)	0.500 (0.000)	0.500 (0.000)

Table 6.9: ARI-10 : Performances of the different classifiers

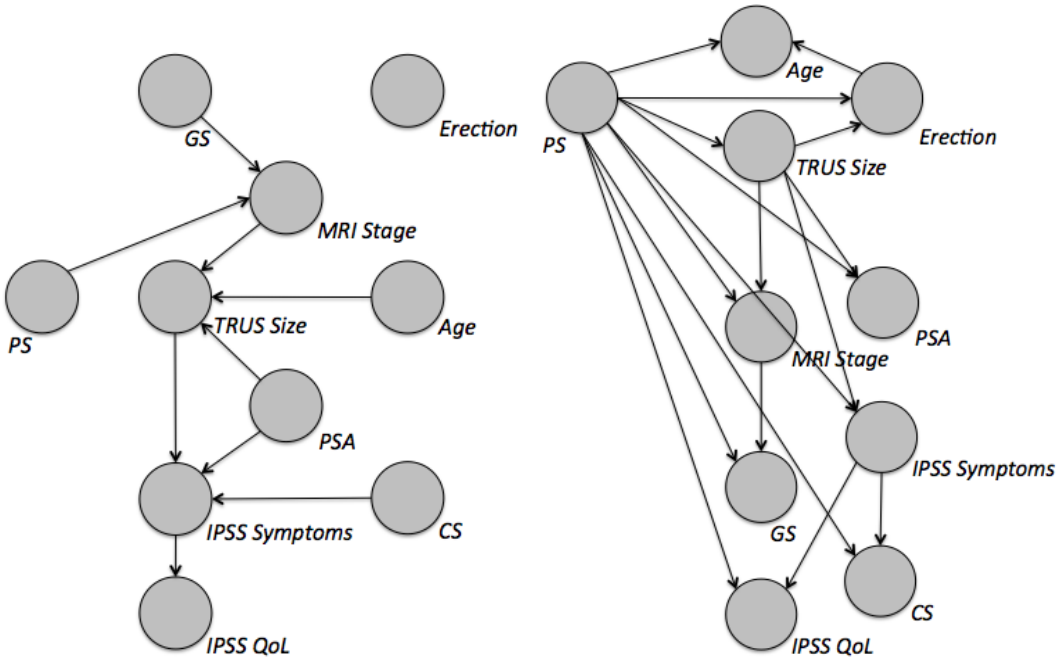
of AUC values is larger than previously observed with other datasets. The standard deviation is also larger for all methods. These two observations is likely to be linked with the small size of the dataset and the larger number of variables.

Similarly to their performances on the BAUS datasets, highest AUCs in three of the four pathological stages are obtained from BN techniques. The best model for prediction of OC and EPE are respectively built using CHBN and TAN. However, their AUC is much lower for EPE and close to zero for LNI prediction where MLP performs best.

The CHBN and TAN structures presented in Figure 6.5 both show similar patterns. For example, edges are modeled between the variable *TRUS size* and PSA, *MRI stage* and *IPSS symptoms*. *IPSS symptoms* also appears as an important node, with relationships with CS, *TRUS size* and *IPSS QoL*. We note that K2GA found that *MRI stage* is the only variable conditionally dependent on PS. Finally, the *pre-operative erection* variable is isolated from other variables.

Over all datasets, and among a pool of various classifiers, techniques using BN offers the best AUC for PS prediction in 10 comparisons out of 12. Only MLP outperforms other methods in two domains. This is observed when MLP is applied on ARI-10 dataset and AUCs for SVI and LNI are measured. k-NN, RF and SVM are generally clearly behind the other techniques in terms of AUC.

To build and assess a predictive model, three main elements need to be taken into



(a) K2GA (b) TAN

Figure 6.5: BN structures learned from ARI-10 dataset

consideration. The choice of the classifier is important but it can only lead to good prediction if the input variables have been chosen carefully. The quality of the data is also a key factor in the process of building a model.

Nine methods were applied to the different datasets. The AUCs of the resulting models vary significantly. For example the AUCs for OC prediction on BAUS-4 range between 0.525, using SVM, and 0.662, using NB. This results from the inherent characteristics and approaches of the different methods, along with the type of data being used. BN techniques have an overall higher predictive power than other methods. However, when measured on BAUS-4 and despite being statistically significant, the difference in mean AUC between NB and LR, the two best performing techniques, is small (0.002). When applied on a dataset containing a larger number of variables, such as BAUS-6, the difference between the two same techniques becomes more marked (0.004). Similarly on ARI-10, LR is clearly outperformed in mean AUC by the best BN model, CHBN (0.134). The dimension of the dataset impacts on the classifier's abilities to produce

high quality models. When only four variables are involved, it is expected to see small differences in performances between the methods as the joint probability distribution is easier to model for low-dimensional data. When new variables are included in the study, some classifiers can lead to complex models. For example, the Euclidean distance, on which the k-NN algorithm is based, loses discriminating power when applied on high dimensional data. This characteristic of k-NN is illustrated when the variables *age* and ASA are included in the BAUS data by a drop in terms of AUC for all class labels. The tree algorithm RF also has difficulties in correctly classifying PS on BAUS-6. However, the quality obtained on the different datasets matches findings from literature (Deng et al., 2011) in which RF is described as a method which is not suitable for problems where predictor variables have different numbers of values. The performance of RF here is also consistent with the drop in AUC observed when the variables *age* and ASA, with 5 and 3 states respectively, are introduced to the model. On the other hand SVM techniques are known for their good performances on high dimensions. We suggest that the poor predictive power presented by SVM on the BAUS datasets can be explained by the number of variables being too few for this method which is particularly adapted to high-dimensional problems (Platt, 1999).

As already mentioned, LR appears as a competitive solution for prostate cancer staging. However, it does not outperform any of the selected alternative methods on any domain. A potential drawback of LR resides in the fact that all predictors are considered as independent from each other. This assumption also is made in a NB. Although this assumption can be justified on BAUS-4 according to the performances of both LR and NB, it does not seem justified on BAUS-6 and ARI-10. On these two datasets, AUCs from TAN and CHBN are overall higher than those from NB and LR. Besides, the associated BN structures reveal relationships between some of the predictor variables. Independence between predictors can be assumed on datasets with a few variables. However, with addition of extra features, interactions are likely to appear that are beneficial for the model's quality. The current version of Partin tables, based on LR may not suffer from the previously discussed assumption, but experiments have shown

that LR is not the best method for PS classification. In addition, the performance gap with other approaches such as BN appears to increase with the dimension of the data to model.

When the first version of Partin tables was created (Partin et al., 1993), Chi-square tests were performed along with LR. The aim was to discover the combination of variables that best correlates with PS. Results led to the conclusion that PSA, GS and CS should be used together, and age of patients removed from the study. In the present study, experiments were run on two subsets of the BAUS dataset. The AUCs of the models built from BAUS-6, including the two new variables ASA and age, were higher than the ones obtained from BAUS-4 with the same methods. Among these methods, LR produced a better model when including age in the study, contradicting Partin's original assumption.

The quality of the data represents a key factor in the construction of a predictive model. BAUS dataset is the result of a large scale data collection, involving 57 different centers. Such data is extremely hard to gather as it involves collaboration between institutions, standardization of the data and ethical issues. As a consequence, some records had to be removed due to inconsistencies. Although the size of the datasets finally used was highly reduced, it remains large with respectively over 1700 and 1500 patient records for BAUS-4 and BAUS-6. The difficulties encountered in the data preparation process is to be taken into consideration as it reflects a current challenge of medical data mining (Ramakrishnan et al., 2010). Local data represents a good opportunity to explore ideas but can also suffer from too few records. Such difficulties were observed on ARI-10 and caution should be taken when analyzing resulting numbers. Another challenge resides in the skewed distribution of pathological stages in both BAUS and ARI datasets. Representing around 1% of all records, LNI condition is the most challenging stage to predict, and is illustrated by standard deviations higher than for other stages for most methods. These results should also be treated carefully as they are likely to vary with the data.

AUC measures show that the classifiers based on BN are better adapted to prostate cancer staging than other methods from our selection. On BAUS-4, NB outperforms TAN and CHBN. In other terms, setting the BN structure to one of its simplest form was beneficial over the use of heuristic search for optimum networks. One could argue on the efficiency of the heuristic employed in CHBN and TAN, but the metric on which the search is based describes how well a structure reflects the data. The CH score assesses a BN in a general way, thus without focusing on a particular variable. Scores of the different BNs presented in Table 8 show that CH values are always higher for CHBN than for TAN and NB as the search strategy aims at maximizing it. However, CH scores and AUCs for PS are not affected in the same manner. A low CH score does not ensure that the corresponding AUC for PS will be high. The development of NB and TAN was originally motivated by this limitation on BN. These two restricted BNs are biased toward a specific purpose, such as classification of a predefined variable. Their performance on BAUS-4 confirms their efficiency on small dimensional datasets over unrestricted networks.

However, as seen with NB, this is an efficient solution when the number of variables to model is low, but shows limitation in other contexts. On BAUS-6, the performances between NB and CHBN are close for OC prediction and NB is outperformed by CHBN for EPE and SVI classification. In a similar manner, experimental results on ARI-10 shows that NB has the worst performance of the three BN methods for AUCs on OC and EPE. In addition, CHBN presents with a AUC for OC of 0.668 against 0.604 for the second best performing technique, k-NN. This important difference reflects the ability of the GA to find a good solution from a large space of possible structures. Heuristic search appears like the right approach when more features are included. Search and score algorithms benefit from using datasets with a large number of variables. The larger search space associated offers more possible dependencies between variables that are likely to further improve the model's predictive power.

The structures found by K2GA represent the most relevant relationships associated with the data. On BAUS-4, the PSA variable is isolated from the rest of the features

		BAUS-6	**ARI-10**
$K2GA$	Fitness	-9464.5337 ($5.30.10^{-12}$)	-999.5378 (0.17)
	S.R	97%	60%
$IMK2GA_{43}$	Fitness	-9464.5337 ($5.46.10^{-12}$)	-999.4796 (0.14)
	S.R	93%	77%
$COMMA$	Fitness	-9464.5337 ($5.30.10^{-12}$)	-999.4454 (0.18)
	S.R	97%	93%

Table 6.10: Performance of three search and score approaches on two medical datasets

and it results in a model with a lower predictive power than the ones obtained from NB and TAN when setting PSA as dependent on PS. This reflects the importance of PSA and is consistent with medical understanding (Partin et al., 1993). From the experimental results on BAUS-6 using K2GA and TAN, PSA is also considered as conditionally dependent of PS. PSA and ASA appear to be correlated with *age*, a medically meaningful finding. In addition, the NB structure on BAUS-4 is retrieved when K2GA is run on BAUS-6. CS, GS and PSA are indeed linked with PS, illustrating that these variables are the most significant for PS. On ARI-10, only *MRI stage* and GS are contained in the Markov blanket of the class variable PS. In this latter model, only *MRI stage* and GS are needed to infer a patient's pathological stage. The presence of *MRI stage* in PS' Markov blanket while CS is not is consistent with the fact that MRI is a more accurate means of evaluating PS than DRE (Sanchez-Chapado et al., 1997). Other relationships were found to be relevant with medical expectations, such as the dependency between *TRUS size* and *age* and PSA (Collins et al., 1993), but others such as the relationship between *MRI stage* and *TRUS size* raised questions regarding their medical meaning (Anastasiadis et al., 2006). This may be due to the small size of the ARI dataset implying that these results should only be considered as exploratory.

	r	*Num. of pl.*	*Largest pl.*	*Mean pl.*
BAUS-6	-0.39 (0.23)	18.77 (2.45)	11.97 (4.98)	3.02 (0.68)
ARI-10	-0.27 (0.24)	19.23 (4.26)	15.20 (10.85)	3.99 (1.70)

Table 6.11: Fitness landscape characteristics of BAUS-6 and ARI-10

6.3 Comparative Study of Bayesian Network Structure Learning Algorithms on Prostate Cancer Data

Since the suitability of BN to model prostate cancer data was proved, the use of the three heuristics described in this thesis was investigated. This step is to ensure that the network structures used for classification purposes are the most representative of the data. Hence, 30 repeat runs of $K2GA$, $IMK2GA_{43}$ and $COMMA$ were performed on the two medical datasets BAUS-6 and ARI-10 and the mean fitness of the best solution found after 1000 fitness evaluations recorded. The GA-based algorithms were set following findings from Chapters 3 and 4, that is with a population of 100 individuals, tournament selection of size 1 and cycle crossover. Using 4 islands and 3 migrations was also justified since this configuration has previously shown best results over all experiments. Similarly, $COMMA$ was set with a population of 10 agents, in accordance with Chapter 5.

Empirical results on BAUS-6 that are presented in Table 6.10 show that all methods are able to find the best network, with a high frequency. This results in a very low standard deviation associated with the fitness of the best solution after 1000 evaluations. This shows a good reliability of the different learning algorithms in finding an optimal structure on this simple 6-variable problem. However, the small size of the problem does not allow further performance comparisons and a sole K2 process may be enough to produce optimal solutions.

With the use of ARI-10, differences can be seen in terms of mean fitness of the best solution. Standard deviations are also higher, as a result of a more diverse search. This variation can be explained by an increase of the search space in comparison with the one from BAUS-6, but also by the size of the data. With 85 records, running K2 is likely to lead to many approximations. The overall best results are obtained by $COMMA$ on this particular problem after allowing 1000 fitness evaluations to be performed. Yet, it is important to note that all methods managed to find the same best structure in at least half of the 30 runs as illustrated by success rates (S.R.) in Table 6.10. Thus, all approaches can be seen as reliable when considering the use of repeat runs on this problem. These best structures are also similar to the ones presented in Figures 6.3a and 6.5a

With lower standard deviations and higher success rates, ARI-10 seems to present less local optima than the other benchmark problems previously studied. However, it was noticed that a particular network structure was learnt by the GA-based algorithms when the best structure was not found. Overall, the structure with the CH score of -999.7363 was obtained 10 times (33% of the runs) by K2GA and 7 times (23% of the runs) by IMK2GA. In addition, this solution was found on the totality of the unsuccessful runs of IMK2GA and on 83% of all unsuccessful runs of K2GA. This solution whose structure is presented in Figure 6.6 can be considered as a local optima. It differs from the best structure by 4 added, 3 omitted and 2 reversed edges, but with 4 similar edges. Using IM helped improving the success rate of the GA, although not to the extent of avoiding the local optima. On the other hand, $COMMA$ showed a good behavior and exhibits a high success rate (93%).

Table 6.11 presents characteristics of the landscapes associated with both BAUS-6 and ARI-10. These includes the correlation coefficient r of the FDC analysis, the number of plateaus, the size of the largest plateau and the mean size of a plateau. Fitness landscape was evaluated following the approach described in Chapter 3, following repeated random walks. Since these random walks start from an optimal orderings, the best structure found over 30 runs on each dataset was used as a reference and Kahn and

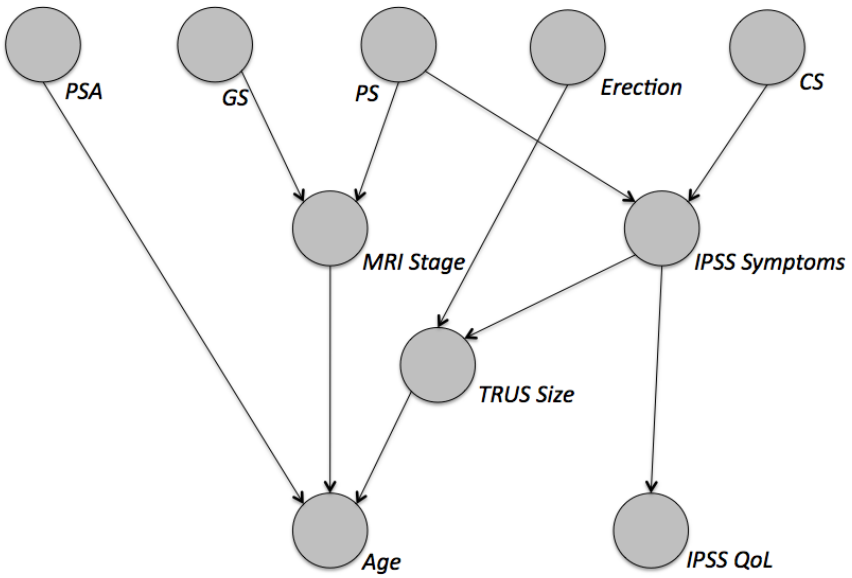


Figure 6.6: BN structure resulting from a locally optimal ordering on ARI-10 dataset

Tarjan orderings derived from it to generate orderings. Findings show that r values are in the same range as those of the benchmarks previously studied, to the exception of *tank*, and it can be extrapolated that finding a BN structure for BAUS-6 and ARI-10 is a problem that is straightforward for optimization methods, with respect to the correlation between fitness and distance to optimal. Numbers related to plateaux are also comparable to those of the benchmark problems.

6.4 Bayesian Network Structure Learning on Distributed Medical Data

6.4.1 Distributed Data in Medicine

Although the field of medical data mining involves many studies, there exist difficulties that relate to the nature of the data itself. These unique challenges offered by medical data are summarized in (Cios and William Moore, 2002) under five main arguments. These issues relate to the questions of data ownership, fear of lawsuits that could result

from mining practitioners' data, data privacy and security, administrative issues and finally potential benefits having to be proved.

In short, data ownership is an open debate that questions whether data regarding patients belongs to the patients themselves or to the institutions treating them. This has direct consequences on agreements that could be arrange to allow access to data to researchers, as it can sometimes be unclear who should be concerned by the problem.

Practitioners also sometimes fear lawsuits that could results from mining the data relevant to their patients as mining studies may reveal some errors relevant to the practitioners' choices regarding patients' treatments. Thus, asking for data may sometimes result in refusal.

The problem of privacy and security of the data covers many aspects such as the risks that are involved when data is handled and transfered between institutions. Sharing data may also influence the relationships between patients and practitioners that is based on confidence. Finally, data verification may be required on some studies. This step may be very difficult to realize if not impossible in cases such as when data is anonymized.

In order to get access to data, proposal often needs to be written and accepted. This suggests that there should be a strong motivation behind the project as well as a thorough project plan. These points may be hard to define in cases and small institutions or individual researchers may experience difficulties in gathering all the evidences required.

The last hurdle to data sharing concerns institutions' administration. Sharing data requires in some cases the implementation of legal contracts, new set of policies, user training, the definition of termination procedures and many other elements. These can represent barriers to potential data mining for both data host and requesting institutions, decreasing the chances for studies to arise.

Although data is highly needed, policies on data protection raise a barrier to large scale projects and many studies are performed using local data, gathered in single

centers. Conclusions of such studies are thus difficult to generalize. Computer science could address this problem by developing a framework to make use of this data while ensuring that patients cannot be tracked through their records. The approach proposed in Chapter 4 and based on the use of IM is investigated in a real-world context, based on the BAUS data.

6.4.2 BAUS Distributed Data

The BAUS-6 datasets was used as a starting point for this study. However, in order to increase the size of the search space, two extra variables were included in the dataset, potency and continence. Since many records exhibited missing values on these two fields, a missing state was included for these variables. This is to ensure the number of instances is similar to the one of BAUS-6 and large enough for modeling. Note that missing states were not added for the other 6 variables of the problem. We call this new dataset BAUS-8. Note that BAUS-8 is only created with the objective to increase the complexity of the modeling task. Results and model structures presented in this section are not to be considered as medically valid, but simply to support the hypothesis that IM can be used to model distributed data.

In order to simulate situations that could arise from the medical domain, the BAUS dataset need to be divided in some ways. This was made possible by consideration of the origin of the records. Thus, each record was mapped to a medical center, which allowed consistent and meaningful partition of the data. In total, 35 NHS research cancer networks were involved in the study. Note that cancer networks are different from single institutions that are part of the BAUS data collection procedure, networks usually covering many institutions.

Because of the large number of cancer networks, the distribution of the different instances of BAUS-8 on each cancer network shows important differences across networks. This includes some networks with a large number of instances (251 for Central South

Area	Size	Cancer networks
North	270	Greater Manchester & Cheshire, Humber & Yorkshire Coast, Lancashire & South Cumbria, Merseyside & Cheshire, Northern, Northern Ireland, Scotland, Yorkshire
Central	818	3 Counties, Arden, Black Country, Leicestershire & Northamptonshire, Mid-Anglia, Mid-Trent, Mount Vernon, Norfolk & Waveney, North East London, North London, North Trent, North West Midlands, South East London, South Essex, Thames Valley, Wales, West London
South	613	Avon & Somerset & Wiltshire, Central South Coast, Dorset, Peninsula, Surrey & West Sussex & Hampshire, Sussex

Table 6.12: Cancer network areas

Coast), but also some networks associated with a very few records (1 for Leicestershire-Northamptonshire and West London) as illustrated in Figure F.1. Such small sample size is not suitable for modelling and merging cancer networks is required. This lead to the creation of three distinct datasets representing the North, Center and South areas of the UK. Table 6.12 summarizes how the cancer networks are grouped under the three labels.

6.4.3 Island Model for Distributed Data

In the previous chapter, in order to assess the quality of learnt BNs, structures were compared to a known true structure from which data was sampled. In the current problem of modeling prostate cancer data, it is not possible to know the true structure of the problem. Hence, to understand the quality of the solutions learnt from distributed data, it was decided to compare them to the best network structure that was discovered on the centralized data, that is when the three individual datasets were merged together.

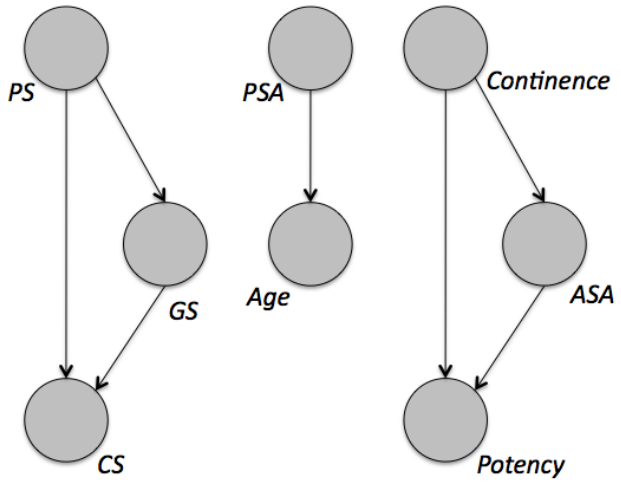


Figure 6.7: Best BN structure obtained by K2GA and IMK2GA on the centralized BAUS-8 dataset

Both K2GA and $IMK2GA_{43}$ were run in order to maximize the chances to obtain a good structure. 30 runs of each with population size 100 and tournament size 1 lead to similar results from both methods. A very small standard deviation was associated with these results ($> 10^{10}$) and it was found that the structure presented in Figure 6.7 exhibits the best CH score (-13860.6307). This BN structure is considered as a reference and is used for comparison when learning BNs using distributed data.

K2GA and $IMK2GA_{33}$ were run on the divided data for 5000 generations. Empirical results are shown in Table 6.13. Orderings obtained at the end of the runs are set as input in K2 to define their quality. This step is performed twice in this set of experiments, using the local dataset first and the centralized data. Note that all repeat runs ended with similar results, that is all islands were associated with an ordering of similar fitness at the end of each run. Thus, standard deviations (0.0) were not included in the tables.

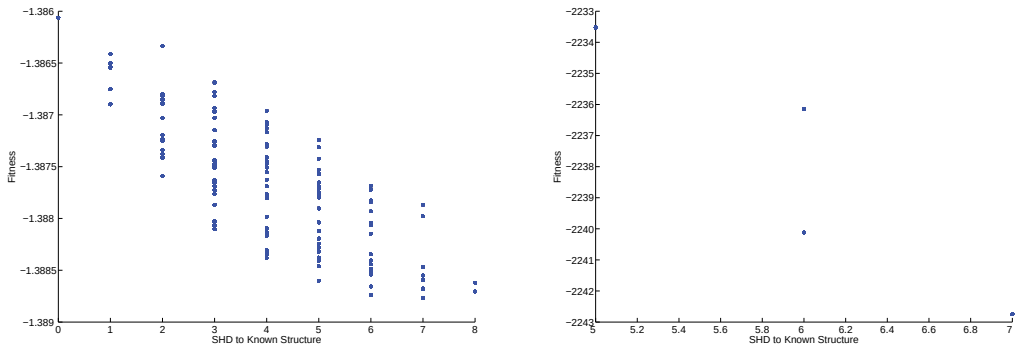
The first observation that can be made concerns the discrepancies that exist between the different islands, thus the different datasets. For instance, the island modeling the Central data ends up the runs with a structure that differs from the one obtained on centralized data by only one spurious edge. However, this is not the case for the

Island	K2 on local data					K2 on centralized data				
	C	R	A	O	SHD	C	R	A	O	SHD
***IMK2GA*$_{30}$**										
North	1	1	0	5	6	3	2	1	0	5
Central	7	0	1	0	1	7	0	0	0	0
South	4	0	1	3	4	5	1	0	1	2
***IMK2GA*$_{33}$**										
North	1	1	0	5	6	3	2	1	0	5
Central	7	0	1	0	1	7	0	0	0	0
South	4	0	1	3	4	5	1	0	1	2

Table 6.13: BN structure quality at each island after 5000 generations in $IMK2GA_{30}$ and $IMK2GA_{33}$ computed from local and centralized run of K2

remaining two islands or datasets that retrieve final BNs of relative poor quality. By looking closely, we can see a correlation between poor structure quality and size of the local dataset. For example, modelling the North dataset that only consists in 270 records leads to a network with only two edges. This can be explained by simply looking at the FDC plot shown in Figure 6.8. The reason why only four points are plotted is because despite generating many distinct orderings during the random walk, only four different structures were found by K2 and assessed by CH score. In short, it means that there are only four potential CH score values that can be attributed to any ordering and that only four BNs can be built from the North dataset. Similarly, there are less structures that could potentially be built from the South dataset than from the Central one. The results presented in Table 6.13 following the K2 run on the best ordering using the centralized data confirms the low quality of the orderings found from the North dataset, in comparison with the other two. Using the Central data leads to an optimal ordering.

The experimental settings are not a good ground for performance comparison between serial and IM-based evolutions. Thus $IMK2GA_{30}$, that is simply three K2GAs running in parallel, using different datasets do not show any difference with $IMK2GA_{33}$ in terms of structure quality. This illustrates a lack of influence of the migration process on the search that can be caused by the difficulties of K2 to use orderings on small

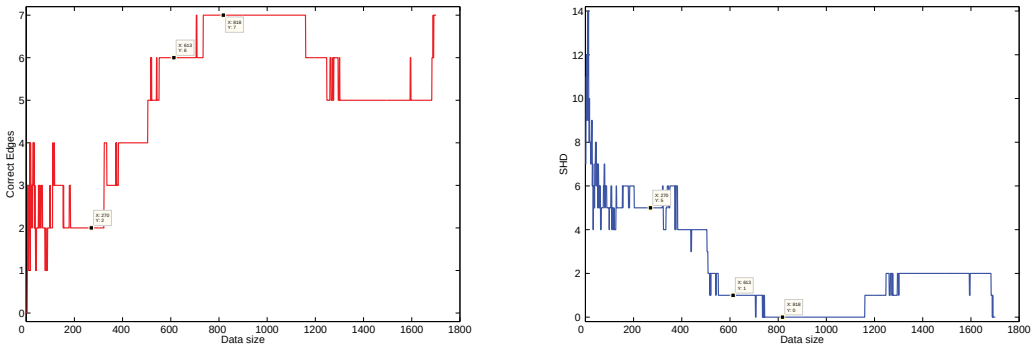


(a) 1701 instances (b) 270 instances

Figure 6.8: Correlation between fitness of an ordering and its related SHD to known structure on complete BAUS-8 data and a 270-instance subset

datasets. Hence, an ordering that can be optimal may be scored with a poor CH value at a specific island because the amount of data available is too small, inhibiting the effect of solution migration.

To illustrate the impact of the size of the data, figure 6.9 covers the variation in SHD in the network learnt by K2 from a similar variable ordering, under variation of the number of instances in the dataset. When datasets with less than 800 instances are used in K2, the number of correct edges starts decreasing as the number of instances is also decreased. The drop in structure quality reaches a critical point where almost no edge can be found. Similar pattern is observed with SHD, illustrating the presence of a minimum number of records needed to efficiently model a problem. These observations are in concordance with previous analyses on K2's abilities (Cooper and Herskovits, 1992). In Chapter 7, a few ideas on how this limitation could be overcome are given as part of potential further work. They should be investigated in order to allow modelling of the distributed BAUS data.



(a) C (b) SHD

Figure 6.9: Impact of dataset size on the quality (C and SHD) of BN structures learned by K2 from an optimal ordering. Labels show the expected quality for the sizes of the BAUS-8 North (270), Central (818) and South (613) datasets.

6.5 Summary

- UK-wide data was available from BAUS and gathered records from patients that were treated with radical prostatectomy in recent years. In addition data from ARI patients was also at hand for modelling purposes.

- One of the main challenge regarding prostate cancer treatment is to predict the pathological stage of new patients based on some markers. Prostate cancer staging is currently clinically done by using lookup tables called Partin tables, based on LR.

- New lookup tables are prepared following the same methodology as the one of the Partin tables, but using the BAUS data. Predictive power is evaluated by using concordance index and AUC. It is found that original Partin tables have a low predictive power on British cases. The new lookup tables lead to improvement but the quality of the prediction remains low.

- A range of classifiers is applied to the BAUS and ARI datasets. Although difference are small when only 4 variables are included in the problem, BNs show better predictive power than other techniques when the data dimension is increased.

- K2GA, IMK2GA and COMMA are assessed on the task of learning BNs from the prostate cancer data. On the BAUS data, all methods perform similarly, reaching the same best structures after 30 runs. On the other hand, COMMA exhibits the best results by both fitness and success rate when modelling the ARI data.

- The fitness landscapes associated with the modelling of both medical datasets present comparable characteristics than those of the benchmark problems, to the exception of none or fewer local optima.

- The importance of modelling techniques that could handle distributed medical data is detailed. NHS Cancer networks are included in the study and in the BAUS data. This enables the creation of three datasets that gather records from different geographical areas of the UK.

- However, the small size of some of the datasets restricts the performance of the K2 process and the search cannot lead to good structures. On the largest dataset, Central, that is composed of 818 records, K2GA is able to retrieve the same structure as the one obtained from centralized data. Resampling techniques need to be investigated to enable modelling of small datasets.

Chapter 7

Discussion and Directions for Further Research

In this chapter, we present the main contributions of the work presented in this thesis. In addition, many limitations were overseen in the experiments conducted throughout this thesis. These should not be disregarded as they help introducing further lines of potential research, aiming at improving BN structure learning. Thus, future work is presented.

7.1 Contributions

The following section highlights the key contributions of the work presented in this thesis.

- A review on BN structure learning using permutation representation and on quality metrics associated with this representation.
- A detailed analysis of the difficulty of learning BN structures from data using permutation representation. The analysis was performed by considering the fitness

landscapes associated with several BN benchmarks, but also investigated the correlation between CH score and BN structure quality. These findings were further analyzed by studying the behavior of K2GA. BN structure learning by means of K2-based approaches generally offers multi-optimal landscapes, exhibiting a large number of plateaus, which is prone to affect the diversity in the population and hence lead to premature convergence.

- An investigation of the impact of IM on K2GA. This includes performance analysis of IM in the classic environment where the data is centralized, but also a novel implementation to enable BN structure leaning from distributed data while preserving privacy.

- Introduction of a novel meta-heuristic to maintain a steady level of diversity in its population. COMMA uses several distance mutation operators in a population of agents in order to assign different roles to each of them. Results show that COMMA is efficient in learning BN structures.

- Applications to prostate cancer data modeling. These include the implementation and comparison of BN-based predictive tools with tools currently used clinically. The use of IM to handle distributed medical data is also approached.

7.2 Directions for Further Research

7.2.1 Parameter Tuning

Regardless of the method that is applied to BN structure learning, the number of parameters that need to be considered and that can affect the search is large. In addition, some of these parameters being of continuous nature, the number of possible combinations of parameter values can sometimes be infinite. It has been seen throughout the experiments carried in this thesis that certain parameters are very influent on the diversity, that is the population size and the tournament size. Tests have been done to

determine two sets of configurations that each defines a different environment, one that is prone to lead to a fast loss of diversity and one that is less prone to it. However, it is important to note that these values result from decisions that were made when designing the experiments and cannot be considered as optimal. The same experimental bias applies to other settings such as the number of islands and migrations in IMK2GA or the number of agents or rules for mutation distances in COMMA.

It is often assumed that experimental decisions are part of the process of comparing algorithms. Experiments presented in this work followed the classical way of running some preliminary studies to find out good values for some selected parameters of importance, before running the final implementations using several repeat runs. However, it would be of interest to look at other experimental approaches that have been proven as efficient. For example, the field of hyper-heuristics (Burke et al., 2003) has given rise to methods that could be applied in the context of BN structure learning. Although there are several paradigm in hyper-heuristics, the main idea is to compose a search procedure by combining search operations that are applied in turn following some strategies. This represents a higher level strategy that is less problem dependent that classical meta-heuristics. Another type of higher level experimental approach consists in preselecting a range of possible algorithm configurations and run a competitions between these, on several instances of a problem. For instance, the F-race algorithm (Balaprakash et al., 2007) starts with a range of configurations and evaluates them on distinct problem instances. When some can be considered as statistically outperformed, they are discarded, and the process is repeated until reaching a satisfactory number of configurations. This approach allows more configurations to be tested since the bad ones will not be applied to a large number of instances before being rejected. Implementation of F-race on K2GA, IMK2GA and COMMA would help to draw a more consistent conclusion on their overall performance.

7.2.2 Use of Alternative Fitness Functions

This work has focus exclusively on one specific type of EAs, that are based on the K2 greedy search. It implies that only one fitness function was used while many more have been implemented as seen in Chapter 2. Different fitness function may bring benefits to the search. For example, a problem related to the use of K2 relates to the size of data that needs to be used. It was seen that when trying to model distributed data, some subsets of the data were not composed of sufficient instances to allow efficient structure learning by K2. K2 showed difficulties in distinguishing between several distinct orderings and was only able to produce a handful of distinct structures. Despite being promising to model distributed data according to experiments on benchmark problems, the use of IM did not lead to usable models when applied to the real-world medical data because of this limitation. It would be of interest to compare other scoring functions on the same datasets, including the BAUS datasets.

Finally, modeling data based on CH score or any BD-derived metric leads to general model of the data. Such approaches are adapted to situations where it is important to have an overview of the relationships that exist between problem variables or where there is not a specific class variable defined. Yet, in other contexts, such as prostate cancer staging, the focus is on a particular variable, the pathological stage for example. In such cases, BN structure learning would benefit from a scoring function that can discriminate between a class variable and other features of the data. A scoring function that fits in this description has been recently implemented (Carvalho et al., 2011) and its application on BAUS data modeling is suggested in order to enhance the predictive power of BN classifiers.

7.2.3 Dynamic selection of mutation parameters in COMMA

In its current implementation, COMMA's strategy is based on using a single type of mutation per run. Mutation distances act as the only way to define the degree of alteration

that is applied to a solution. However, mutation types have different relationships with the mutation distance. For example, two families of mutation operators were defined, that applies an alteration whose value can be expressed either in a deterministic or stochastic way. One of the drawback of the current implementation lies in the fact that using a specific mutation type sets constraints on the maximum and minimum extent of the alterations that are applied to each solution. For example, choosing swap mutation means that all solutions that are produced from a generation to another do not differ much from their parents, relative to other mutation types, regardless of the mutation distance used. Similarly, small alterations are very unlikely with invert+displacement, even with low mutation distances.

A proposed way to overcome this limitation of COMMA is to build lookup tables containing the ordering similarity distance between a solution and its potential offspring for each combination of mutation type and distance. These tables, similar to Tables 5.1 and 5.2 can then be used to dynamically allocate a mutation distance and a mutation type to a given agent according to its rank in the population. Hence, instead of assigning a different mutation to each agent, they would be assigned a level of alteration they need to apply to their associated solution. Such approach is also beneficial in that it reduces the number of parameters involved in the algorithm design as no mutation types, nor distance would need to be decided arbitrary.

COMMA is a novel meta-heuristic approach that has been implemented for BN structure learning. Yet, it is a method that search in the space of permutations and thus that could suit many other applications. The aforementioned idea of dynamic choice of level alteration within the algorithm would lead to better conclusions if applied to a range of problems. These includes the classic benchmark problems such as the Traveling Salesman Problem or Flow Shop Scheduling for example.

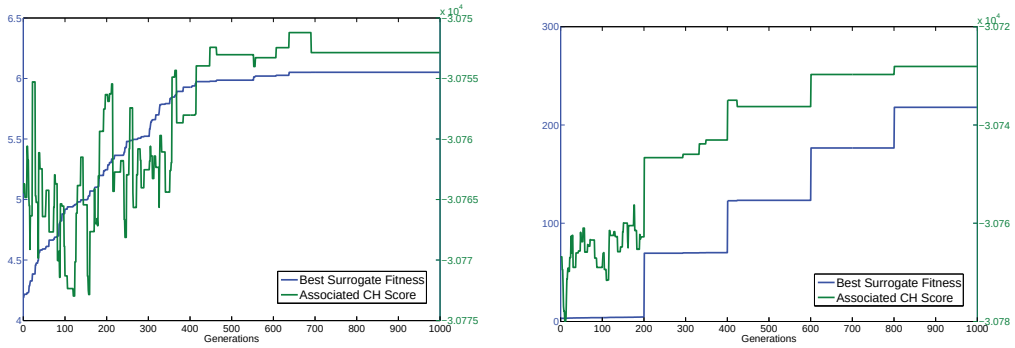
7.2.4 Implementation of Surrogate Fitness Function

Running K2 can be a computationally expensive process on large data. This results in long runtime associated with K2-based EAs, such as K2GA or COMMA. Although this point is not treated in this thesis, the need for surrogate fitness model is real and such techniques would bring a lot of advantages to BN structure learning. With the current implementations, it can take days to perform a single run of K2GA on large datasets, which fixes some limits on the number of variables that can be modeled using K2GA or COMMA.

The idea of surrogate fitness model consists in assessing a few solutions using a given fitness function, such as the CH score resulting from K2, and to build a model based on these fitness values and the solution genes. Naturally, a lot of work in the field of fitness surrogate is derived from EDAs, whose strategy is to model the distributions of good solutions in order to produce new individuals. For example, work has been carried out in implementing a surrogate fitness model based on Markov Random Fields in (Brownlee et al., 2010, 2012) for binary representations. Once built, the model was fitted into a GA in order to guide the search, showing important benefits in terms of runtime, but to the cost of a drop in solution quality. Work has also been carried out in modeling the fitness associated to permutation spaces (Ceberio et al., 2012). This lead to the implementation of surrogate fitness function used to support the search in EAs. In (Zhang and Sun, 2006; Chen et al., 2012), a first-order marginal model is used to guide a tournament between several crossover and mutation types within an EA. Such model typically builds a table of probabilities for each variable-position pair. This work is also a good example of how tuning steps can be introduced in an algorithm at no cost. The idea was extended in (Ceberio et al., 2011*b*) to include more orders into the marginal representation within an EDA. Another permutation-based EDA (Ceberio et al., 2011*a*) makes use of the so-called Mallows model that defines probability distributions associated with permutations, based on their distance to a central permutation.

Preliminary runs were performed to get an insight on how surrogate models would perform on the problem of learning BNs. Because of its simplicity, the k-order marginal model (Ceberio et al., 2011*b*) was adapted to guide the search in K2GA on five benchmark problems, *asia*, *tank*, *credit*, *car* and *boerlage*. A population of 100 individuals is initially evaluated using K2. The 30 solutions with the highest CH scores are selected and a k-order marginal model is built from their characteristics. 1000 generations are then performed, based on the surrogate fitness obtained by bringing together the probabilities for each variable-position pair of the permutations. For this set of experiments, the probabilities were simply summed to provide a fitness value. 10 runs were performed on each benchmark problem, with datasets of 3000 instances. Finally, Laplace correction was used to avoid the presence of zero probabilities in the model. Experiments were run with and without model update, with a model of order 2. When the model is updated, a new model is built from the 30 best solutions in the population and replaces the previous one. Figures 7.1, G.1, G.2, G.3 and G.4 illustrate how the surrogate fitness of the best solution evolves over time. In addition, the CH score corresponding to the solution with the highest surrogate fitness value is also displayed. This enable comparison between the two fitness functions. In theory, a perfect surrogate would match the curve of the real fitness function. Although, there is a lot of noise and the correlation between the two fitness functions is not perfect, there exists clearly a relationship between them. This is particularly apparent when observing the runs where no update is involved. Results also show that update helps fixing some approximations of the surrogate model. However, it seems that the crude update strategy chosen here does not help the search and better strategies that keep properties of previous models should be used in place.

This series of experiments is very rudimentary and no objective conclusion can be reached. However, the correlation between surrogate 2-order marginal fitness function and CH score suggests that a more thorough analysis should be performed. Considering other types of permutation-based models and a more meaningful way to produce a fitness function from it are two guidelines that may benefit BN structure learning.



(a) No Update

(b) Model Update

Figure 7.1: Evolution of surrogate fitness of the best solution and its associated CH score on *boerlage* with and without model update

7.2.5 Investigation of techniques to handle small datasets

Experiments on benchmark datasets showed that the use of IM can have another benefit than enhancing diversity in the population. IMK2GA was successful at modeling distributed data with distinct distributions but similar set of variables. It also showed that using distributed data can enhance results that would be obtained after centralization. Yet, the same technique was not as successful at modeling the distributed medical data. The small number of records at some of the island was identified as a potential cause, preventing K2 to produce good structures.

Investigating ways to handle data with such properties is a natural way to follow up on the research on privacy-preserving distributed data modeling presented in this thesis. The issue is very likely to be observed when modeling real-world data and this line of research would ensure that there is a link between the theory presented here and its application to practical problem.

7.3 General Conclusion

The work presented in this thesis covers many points of the specific application of BN structure learning using EAs. Among other things, it highlights the importance

of considering the fitness landscape of any problem in order to understand algorithm performance. In the context of BN structure learning, different data present different landscape characteristics. It is also important to consider the differences that can exist between fitness values and actual BN properties. It appeared for example that despite CH score and BN quality being overall correlated, exception can be found in some datasets such as *tank*. Such knowledge about a problem helps understanding findings. Generally, experimental work have shown that the landscapes associated with BN structure learning exhibit many local optima and plateaux when using permutation representations and K2 to score solutions. This means that many distinct solutions can produce the same BN structures following K2, and hence be set with similar CH scores. This property of the landscape favors early convergence and fast loss of diversity in the population.

With population-based algorithms, one of the main factor influencing the search is the level of diversity that exists at any time in the population. Typically, this factor can be managed by consistent and thorough parameter tuning. For instance, the population and tournament sizes affect highly selection pressure and thus the diversity. However, at occasions, these parameter values can become impractical to ensure sufficient diversity in the population (very large population size for instance). IM were applied to K2GA as an alternative to thorough parameter tuning. Using IM when K2GA was set with small population and large tournament size shows that IM can enhance the performance in environments that are prone to generate a loss of diversity. However, in other types of environments where K2GA does not suffer from premature convergence, the quality of the BNs obtained by IMK2GA is not always as good as those of K2GA. This is especially true on large datasets where a slow convergence is observed. COMMA was designed in order to observe the performance of search and score population-based algorithms that can be reached when the loss of diversity is not a concern anymore in the search. COMMA shows that it is competitive with K2GA and IMK2GA on the experimental bounds set but could lead to better results if its runtime was increased.

Yet, although both proposed approaches, IMK2GA and COMMA, show promising

insights, their implementations introduce another set of new parameters that need careful tuning. For example, particular parameter configurations on IMK2GA can result in too slow convergence, another limitation. Thus, on some experiments and despite solving the problem of premature convergence, satisfactory results may not be reached. IMK2GA and COMMA were implemented as proofs of concept and were not optimized in this work. To ensure an optimum use of both IMK2GA and COMMA, it is important to try to reduce the number of parameters or to change the approach behind the tuning of parameters, such as using some surrogate fitness to perform tuning steps within the search.

The work carried out also considered a common limitation to many data modeling tasks. It relates to the use of distributed data. While the IM principle represents a natural and straightforward way to model such data, good results are also dependent on the quality of the data and its size. When using K2, small sample size impedes K2 from learning generalisable networks. Solutions to these problems may lie in the topics of resampling, or in studying alternative to K2 that can handle small datasets. On the other hand, IMK2GA appears as a good method to model data from distinct locations when the size of each local dataset is sufficient. When datasets present important dissimilarities in their distributions, the use of IM on distributed data offers advantages over modeling the same data after centralization.

The aforementioned topics were analyzed in a real-world medical context. It was shown that prostate cancer staging can benefit from using BNs to build both general model and classifiers. However, the issue previously raised regarding the sample size acted as a barrier to using IM to model distributed medical data from the data available at hand.

In general, we have showed that diversity is an essential factor of performance for population-based algorithms that can be managed in several ways. This particularly applies to BN structure learning because of its characteristic fitness landscape and should always be considered when implementing EAs. This study contributes to the

understanding of well-established EA in the domain of BN structure learning and suggests further applications in other domains. In addition, it also demonstrates that these methods can benefit real-world data modeling and in particular in the medical domain.

Appendix A

Example of CH score calculation

This appendix describes how the addition of a node to the parent set of a given variable can lead to a decreased CH score. We consider three binary variables X_1, X_2 and X_3. The dataset given in Table A.1 is used throughout this example.

In this example and given the ordering $X_2 - X_1 - X_3$, we investigate the effect of including X_1 in the parent set $Pa(X_3)$ of X_3. Note that we assume that X_2 was added to $Pa(X_3)$ prior to this step. To do so, $CH(X_3, Pa(X_3))$ is first computed using X_2 as only parent. In the second step, $CH(X_3, Pa(X_3))$ is calculated when both X_2 and X_1 are set as parent of X_3. We recall that the calculations are based on (2.19).

X_1	X_2	X_3
Y	N	N
Y	Y	Y
N	N	Y
Y	Y	Y
N	N	N
N	Y	Y
Y	Y	Y
N	N	N
Y	Y	Y
N	N	N

Table A.1: Dataset used as an example for CH score calculation (Y = Yes, N = No).

$$Pa(X_3) = \{X_2\}$$

$CH(X_3, Pa(X_3)) = \frac{1}{6!} * 4! * \frac{1}{6!} * 5!$

$CH(X_3, Pa(X_3)) = 0.0055$

$$Pa(X_3) = \{X_2, X_1\}$$

$CH(X_3, Pa(X_3)) = \frac{1}{5!} * 3! * 1! * \frac{1}{2!} * 1! * 0! * \frac{1}{2!} * 0! * 1! * \frac{1}{5!} * 0! * 4!$

$CH(X_3, Pa(X_3)) = 0.0025$

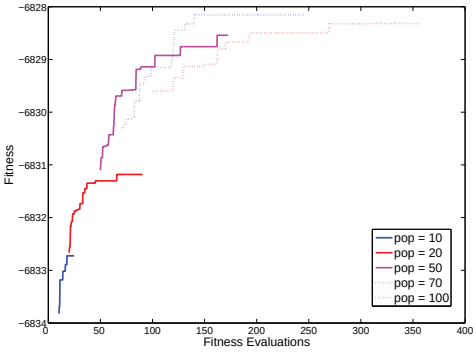
$$CH(X_3, \{X_2\}) > CH(X_3, \{X_2, X_1\})$$

Hence, K2 will discard X_1 as a choice of parent for X_3.

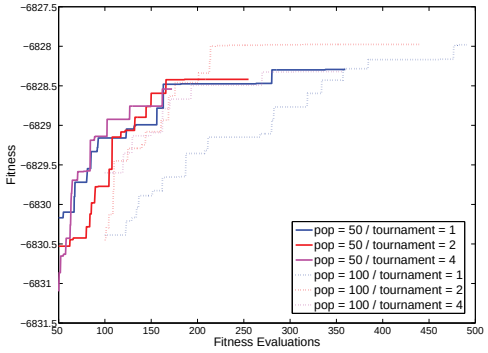
Appendix B

Population and tournament sizes and convergence of K2GA

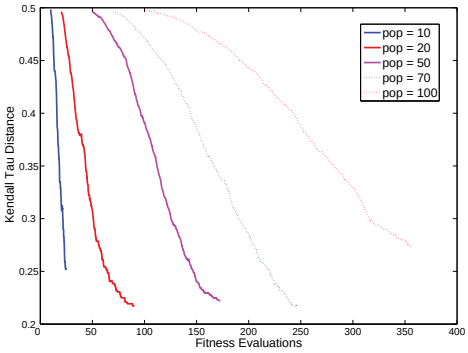
This set of figures relate to experiments described in Section 3.2 of Chapter 3.



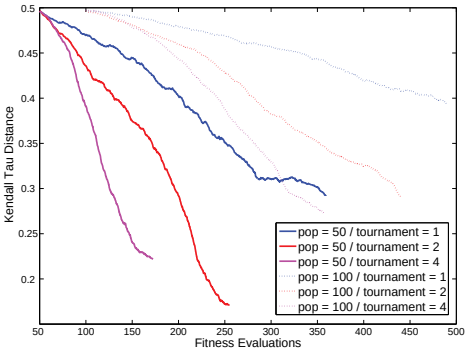
(a) Effect of population size on fitness



(b) Effect of tournament size on fitness

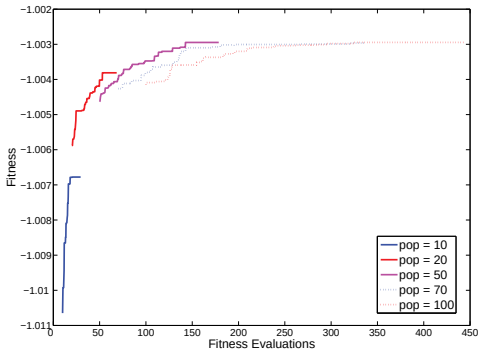


(c) Effect of population size on KTD

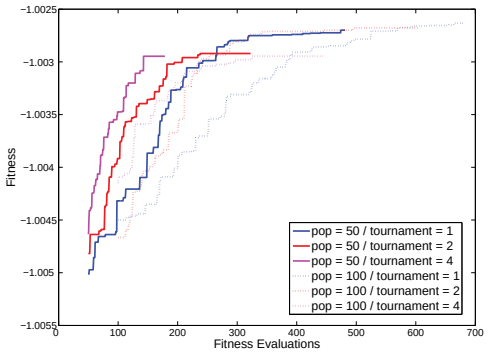


(d) Effect of tournament size on KTD

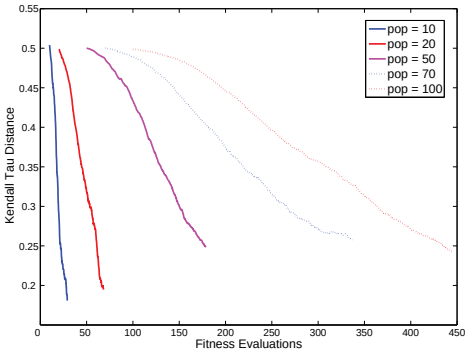
Figure B.1: Convergence speed of K2GA on *asia*



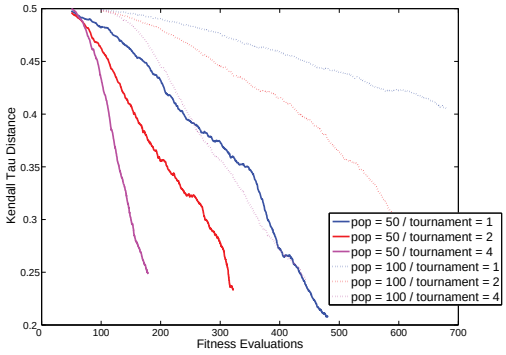
(a) Effect of population size on fitness



(b) Effect of tournament size on fitness

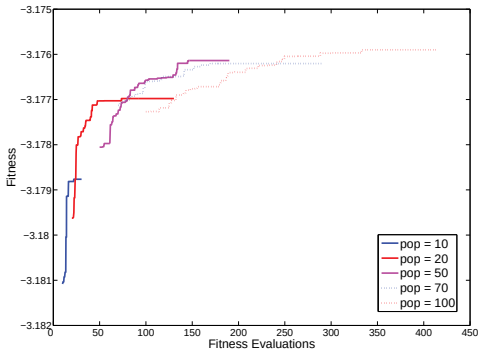


(c) Effect of population size on KTD

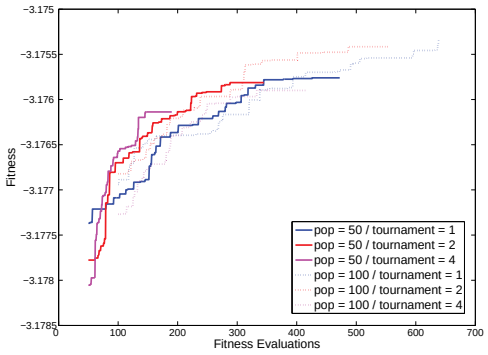


(d) Effect of tournament size on KTD

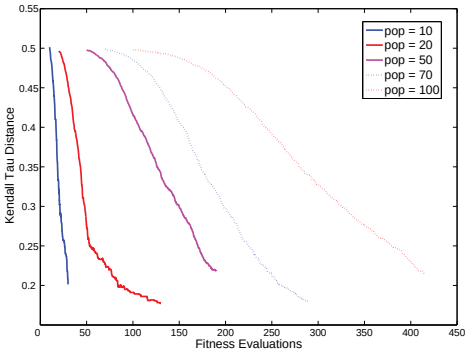
Figure B.2: Convergence speed of K2GA on *tank*



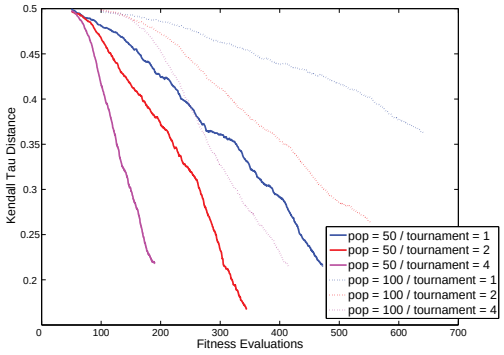
(a) Effect of population size on fitness



(b) Effect of tournament size on fitness

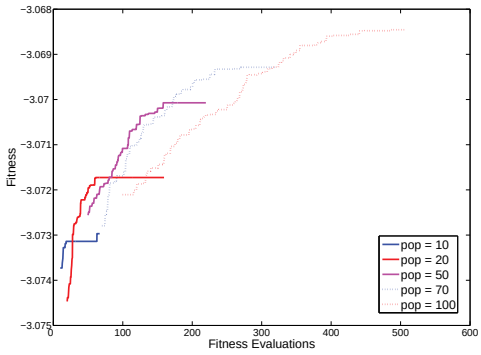


(c) Effect of population size on KTD

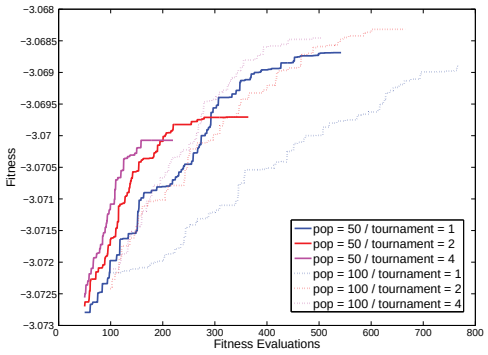


(d) Effect of tournament size on KTD

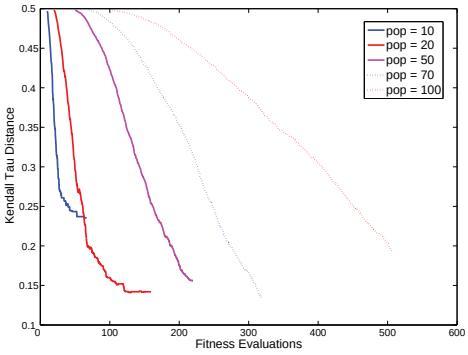
Figure B.3: Convergence speed of K2GA on *credit*



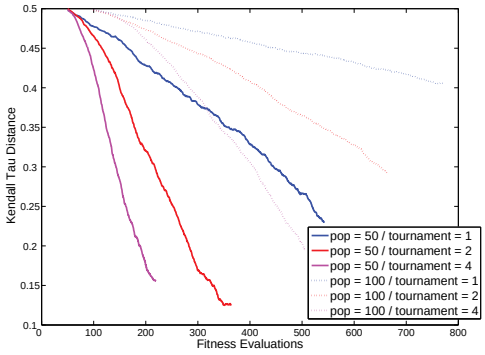
(a) Effect of population size on fitness



(b) Effect of tournament size on fitness

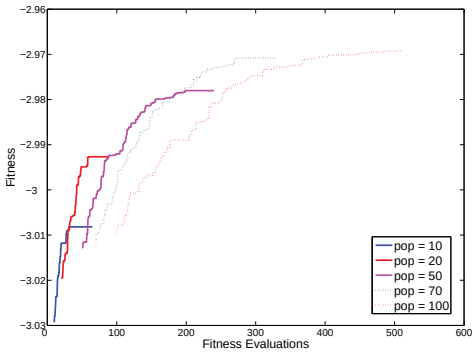


(c) Effect of population size on KTD

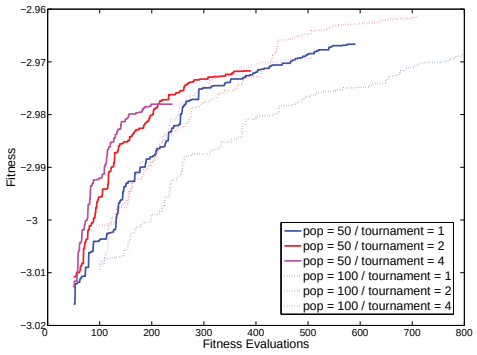


(d) Effect of tournament size on KTD

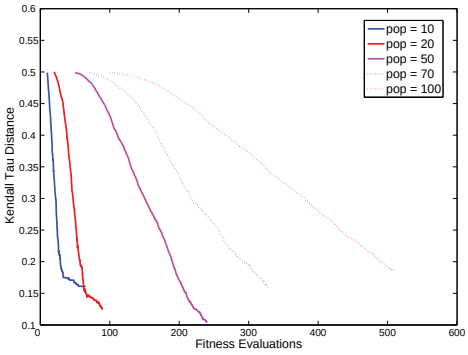
Figure B.4: Convergence speed of K2GA on *boerlage*



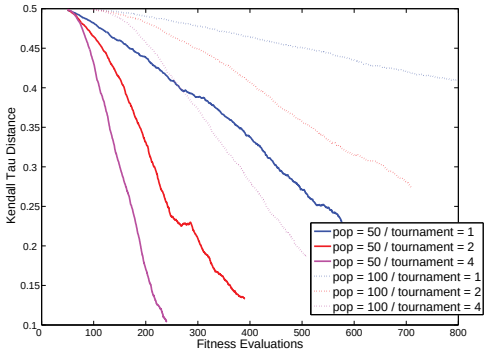
(a) Effect of population size on fitness



(b) Effect of tournament size on fitness



(c) Effect of population size on KTD



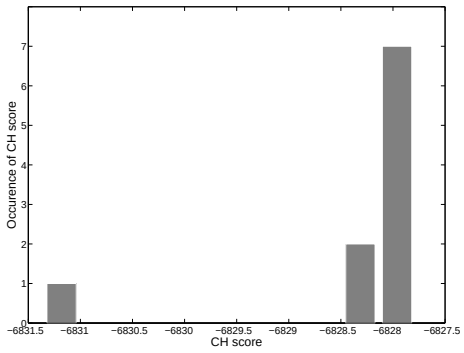
(d) Effect of tournament size on KTD

Figure B.5: Convergence speed of K2GA on *alarm*

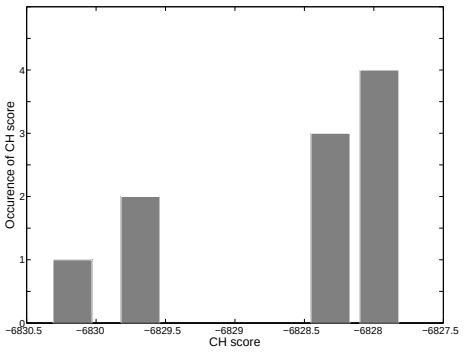
Appendix C

Evaluation of the presence of local optima using K2GA

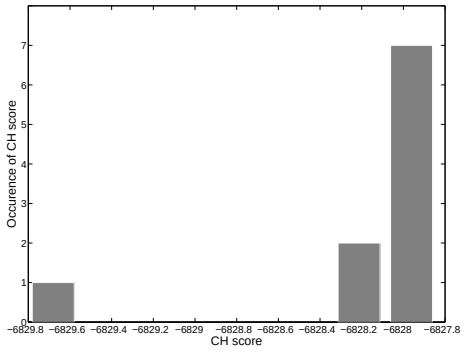
This set of figures relate to experiments described in Section 3.2 of Chapter 3



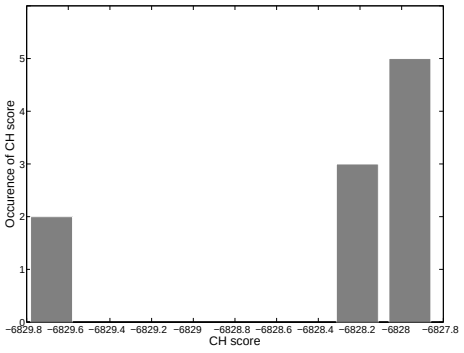
(a) Population size = 50, tournament size = 1



(b) Population size = 50, tournament size = 4

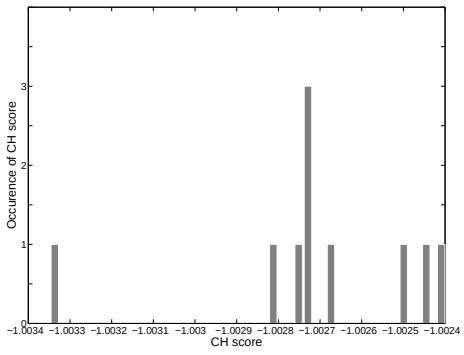


(c) Population size = 100, tournament size = 1

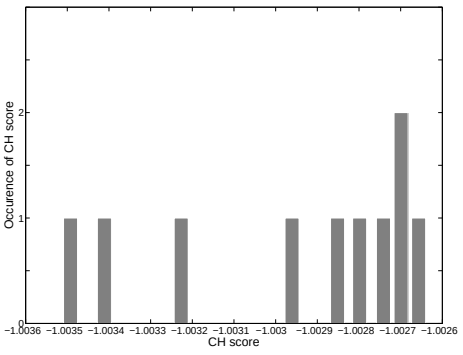


(d) Population size = 100, tournament size = 4

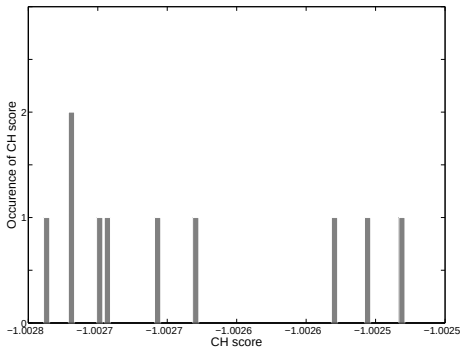
Figure C.1: Frequency of best orderings obtained over 10 runs of K2GA on *asia*



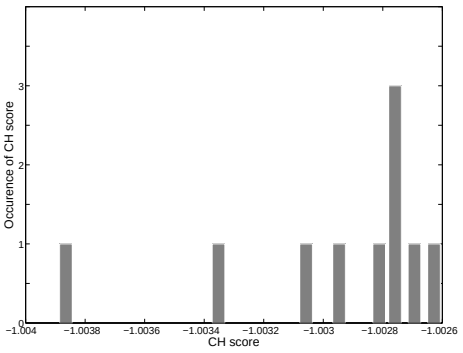
(a) Population size = 50, tournament size = 1



(b) Population size = 50, tournament size = 4

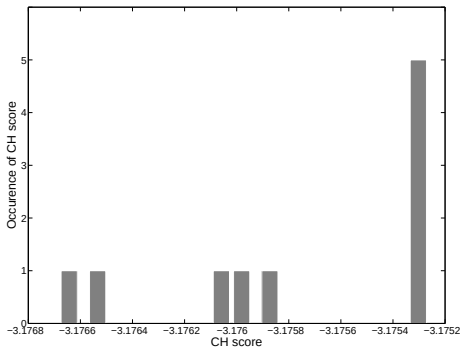


(c) Population size = 100, tournament size = 1

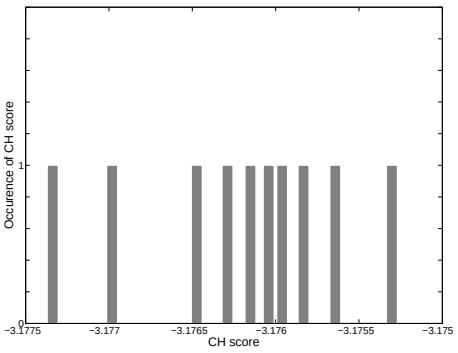


(d) Population size = 100, tournament size = 4

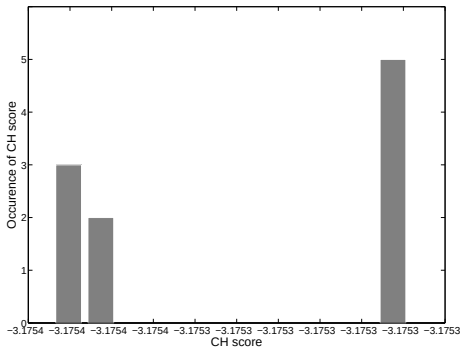
Figure C.2: Frequency of best orderings obtained over 10 runs of K2GA on *tank*



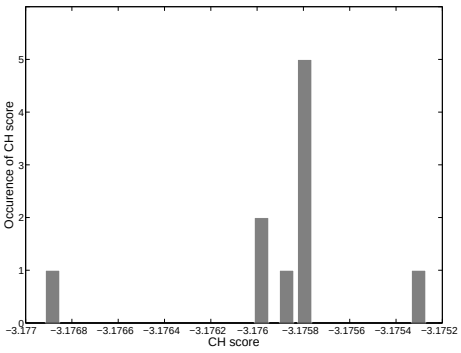
(a) Population size = 50, tournament size = 1



(b) Population size = 50, tournament size = 4

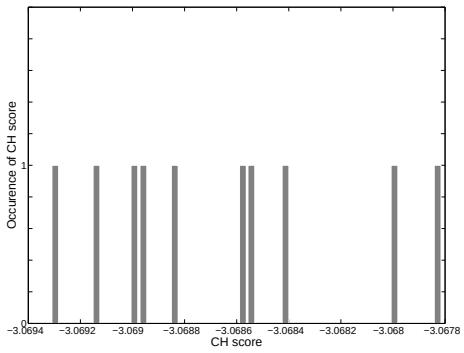


(c) Population size = 100, tournament size = 1

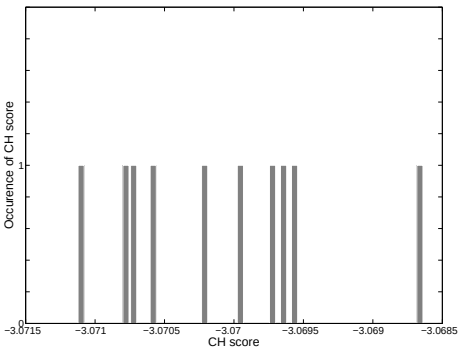


(d) Population size = 100, tournament size = 4

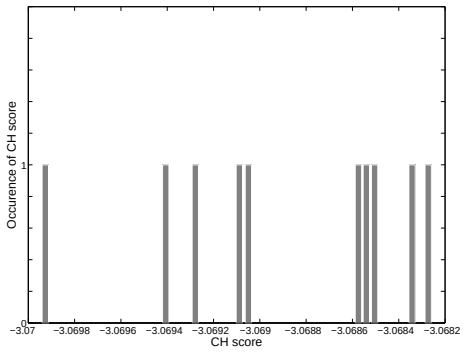
Figure C.3: Frequency of best orderings obtained over 10 runs of K2GA on *credit*



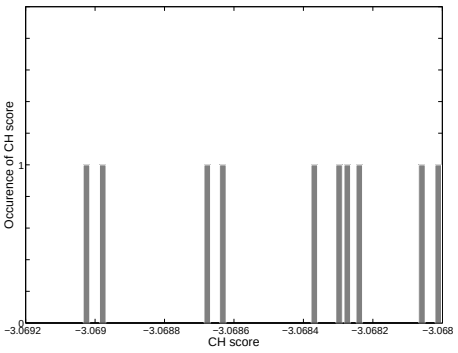
(a) Population size = 50, tournament size = 1



(b) Population size = 50, tournament size = 4

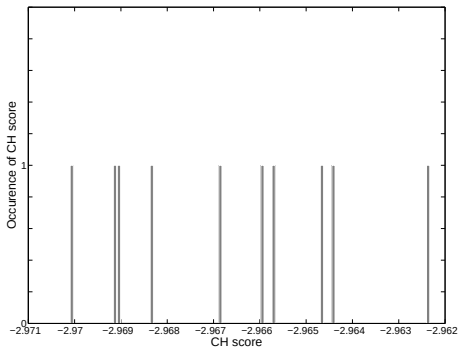


(c) Population size = 100, tournament size = 1

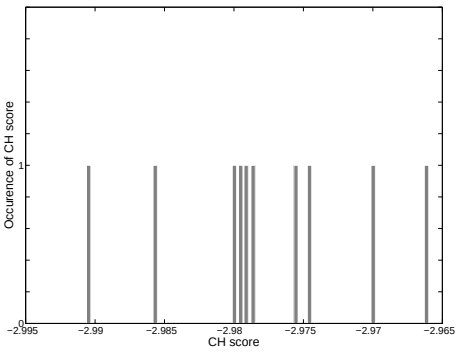


(d) Population size = 100, tournament size = 4

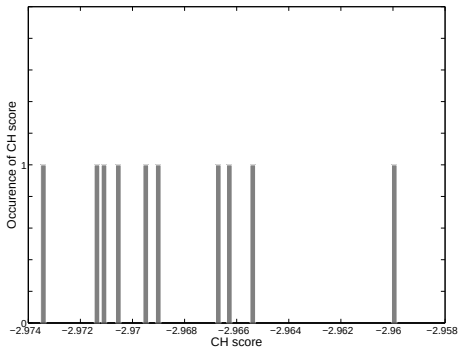
Figure C.4: Frequency of best orderings obtained over 10 runs of K2GA on *boerlage*



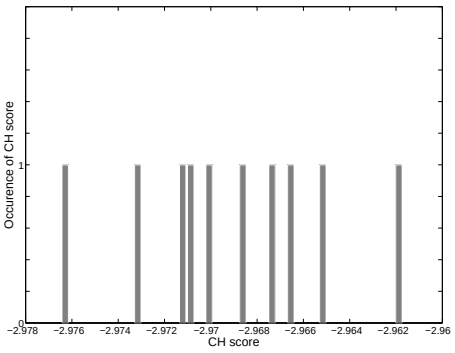
(a) Population size = 50, tournament size = 1



(b) Population size = 50, tournament size = 4



(c) Population size = 100, tournament size = 1



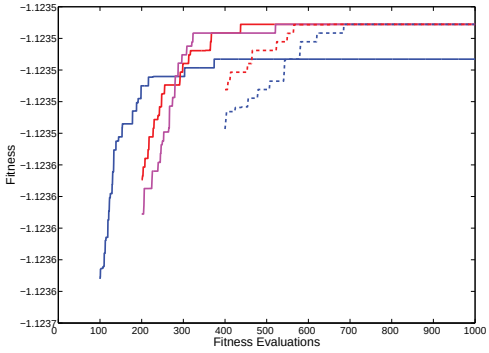
(d) Population size = 100, tournament size = 4

Figure C.5: Frequency of best orderings obtained over 10 runs of K2GA on *alarm*

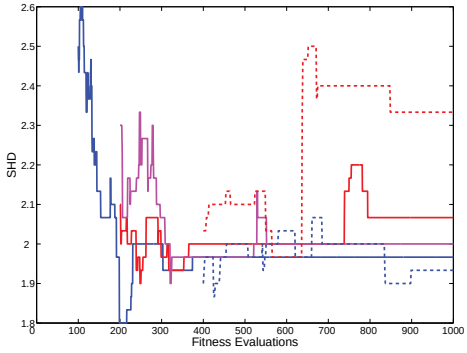
Appendix D

Performance analysis of IMK2GA

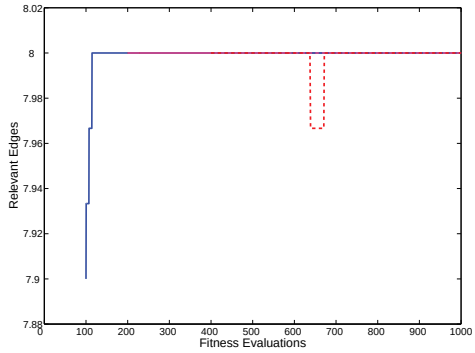
This set of figures relate to experiments described in Section 4.2 of Chapter 4



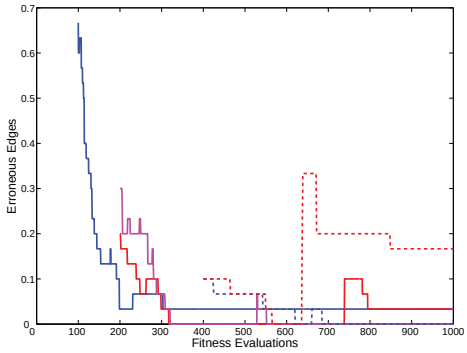
(a) Fitness



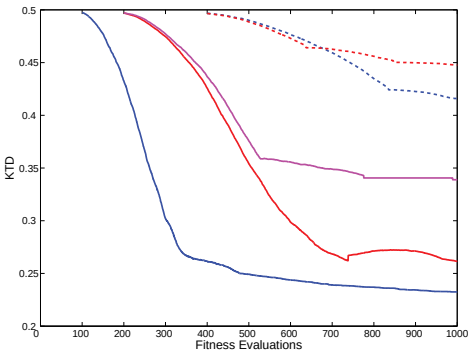
(b) SHD



(c) Relevant edges

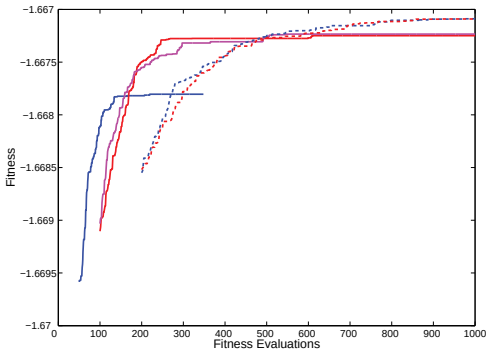


(d) Erroneous edges

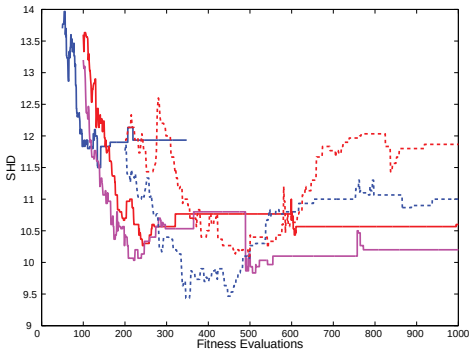


(e) KTD

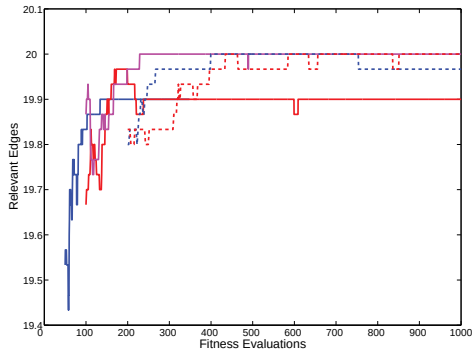
Figure D.1: Evolution of solution quality and population diversity on *asia*



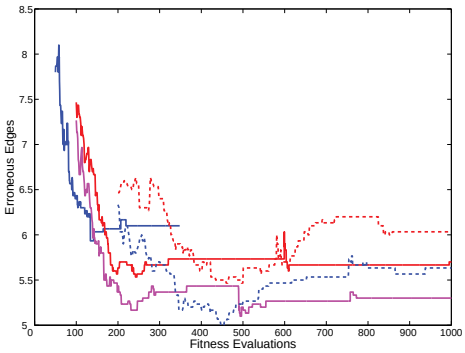
(a) Fitness



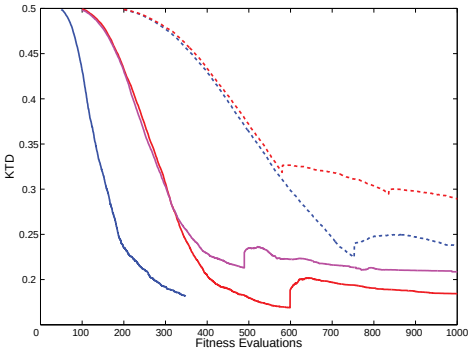
(b) SHD

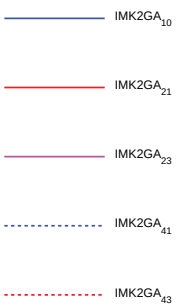


(c) Relevant edges



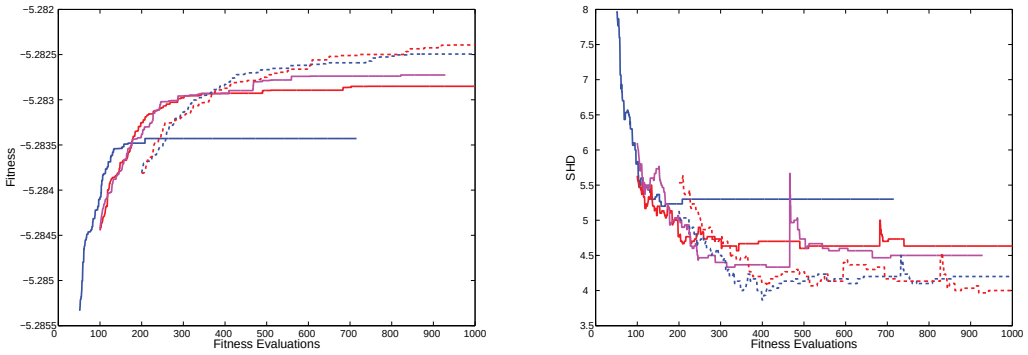
(d) Erroneous edges





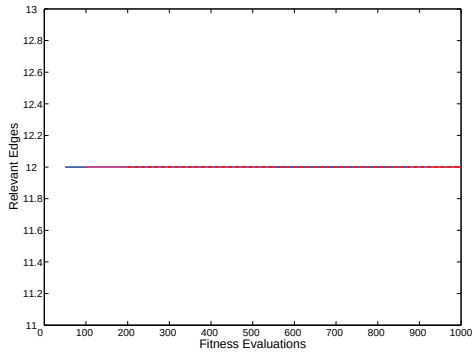
(e) KTD

Figure D.2: Evolution of solution quality and population diversity on *tank*

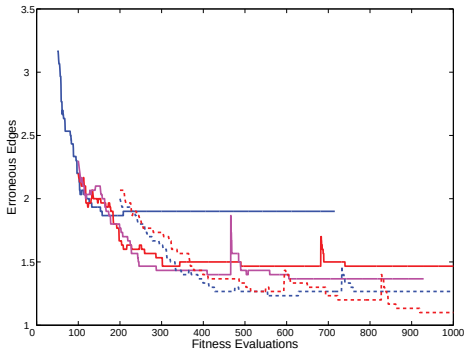


(a) Fitness

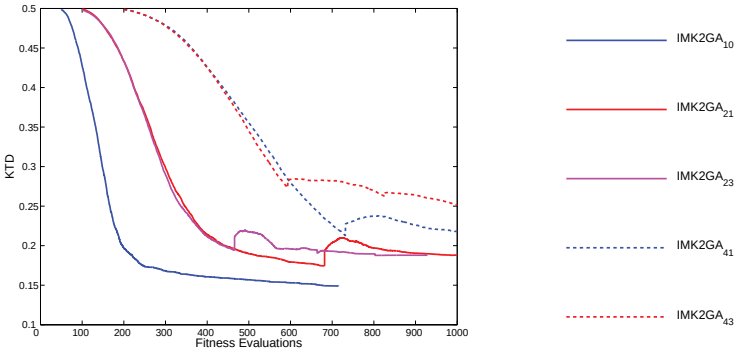
(b) SHD



(c) Relevant edges

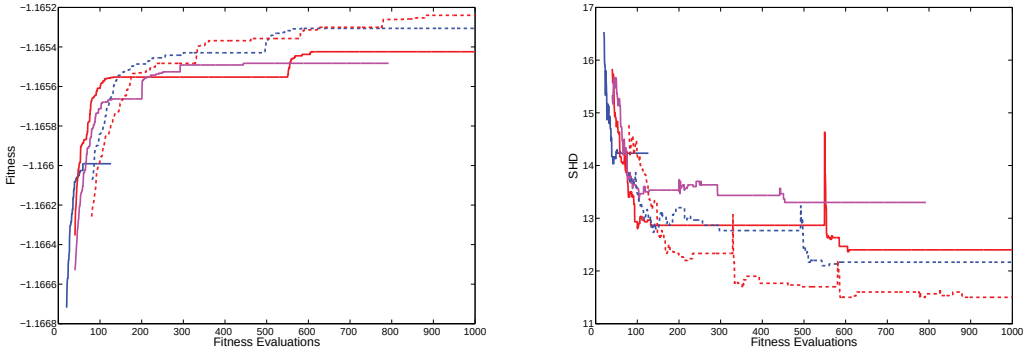


(d) Erroneous edges



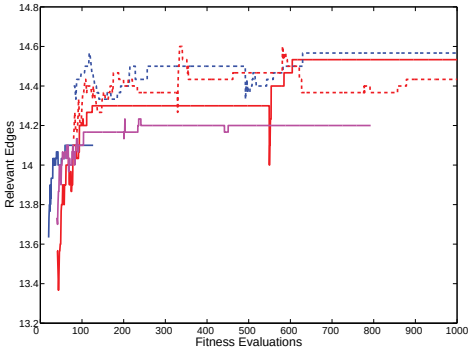
(e) KTD

Figure D.3: Evolution of solution quality and population diversity on *credit*

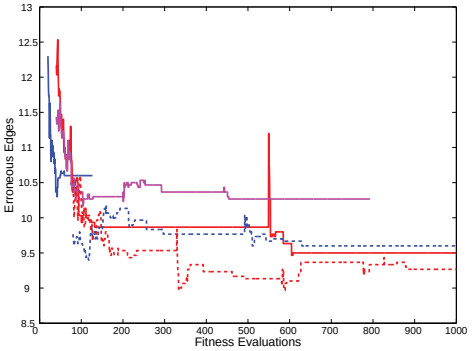


(a) Fitness

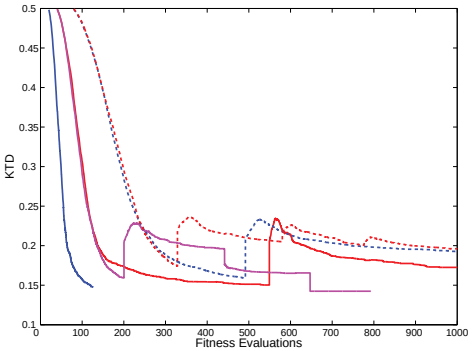
(b) SHD

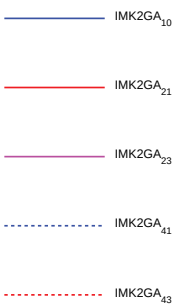


(c) Relevant edges



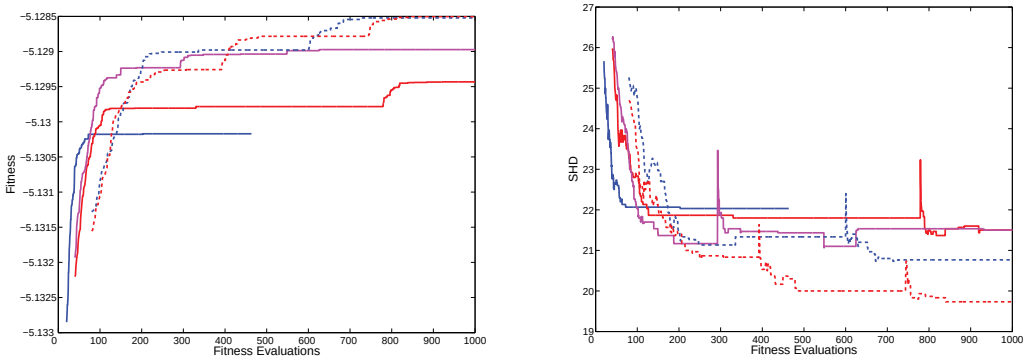
(d) Erroneous edges





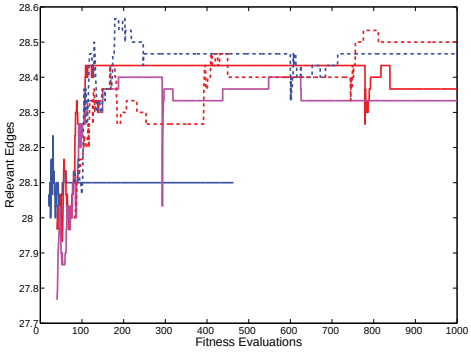
(e) KTD

Figure D.4: Evolution of solution quality and population diversity on *car*

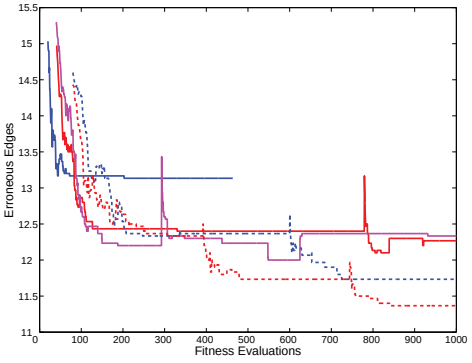


(a) Fitness

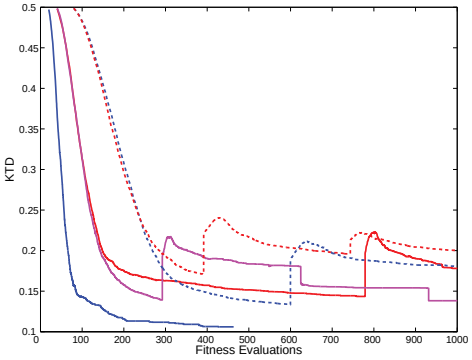
(b) SHD

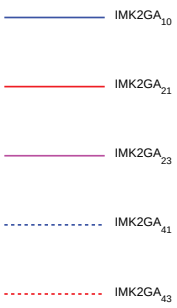


(c) Relevant edges



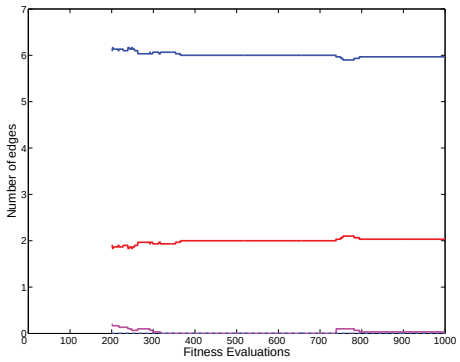
(d) Erroneous edges



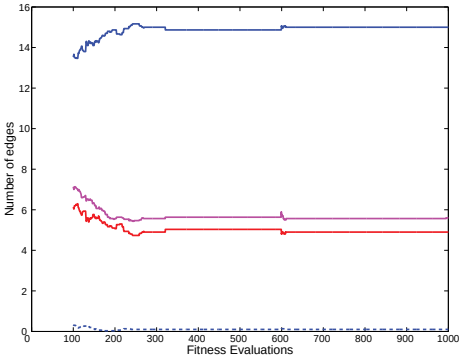


(e) KTD

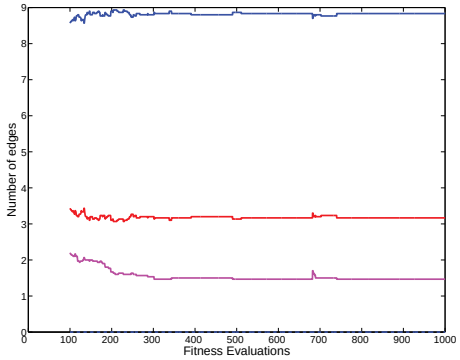
Figure D.5: Evolution of solution quality and population diversity on *boerlage*



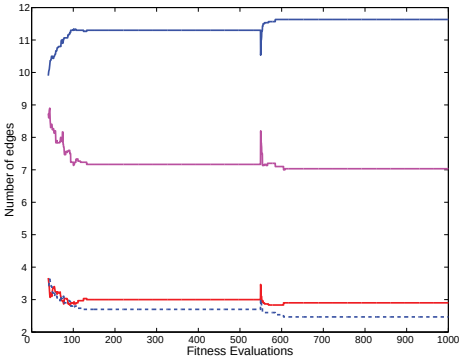
(a) asia



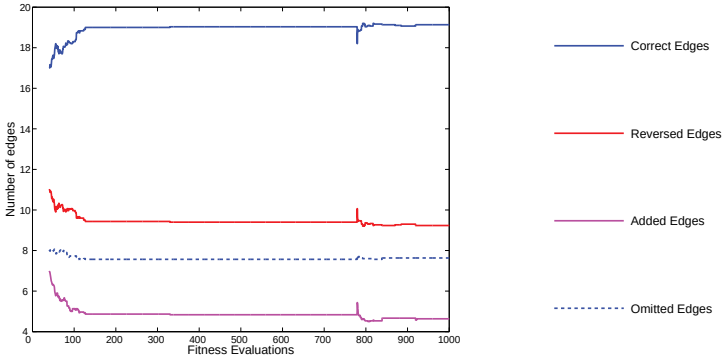
(b) tank



(c) credit

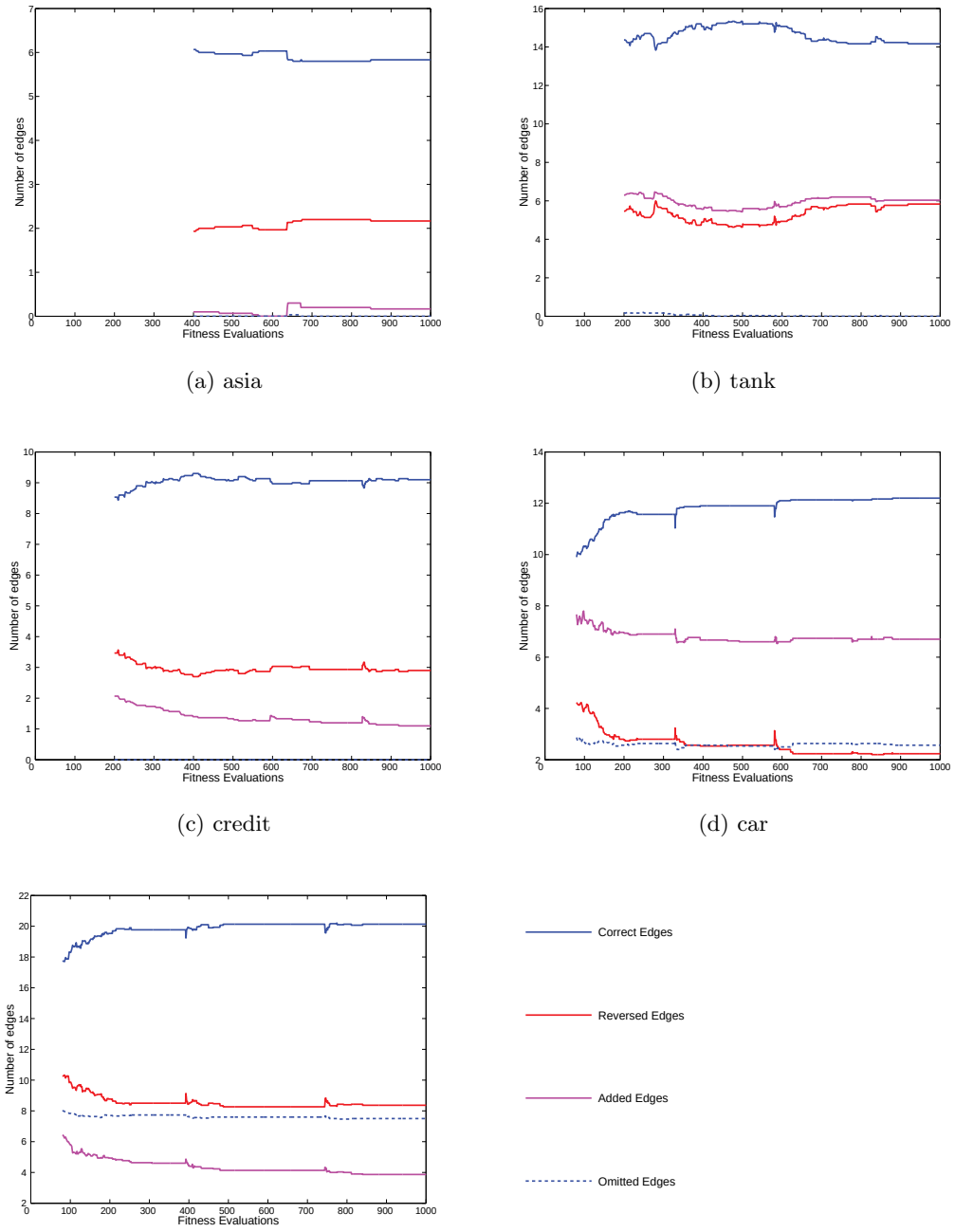


(d) car



(e) boerlage

Figure D.6: Evolution of correct, reversed, added and omitted edges throughout the search with $IMK2GA_{21}$



(a) asia

(b) tank

(c) credit

(d) car

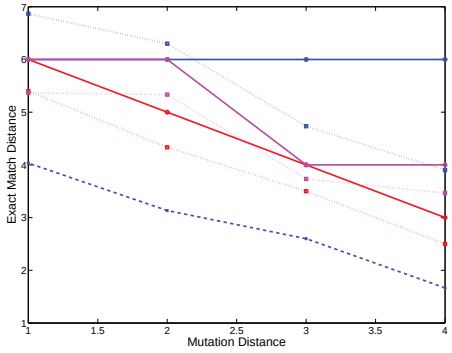
(e) boerlage

Figure D.7: Evolution of correct, reversed, added and omitted edges throughout the search with $IMK2GA_{43}$

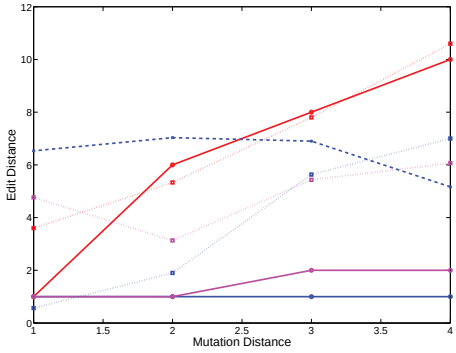
Appendix E

Choice of Distance-Based Mutation Operators

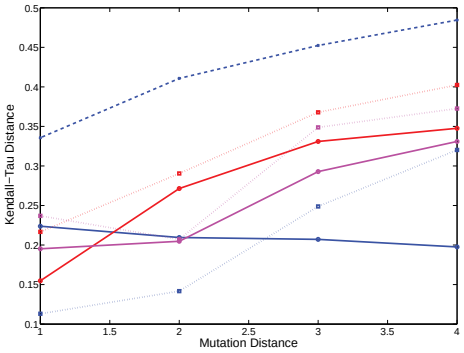
This set of figures relate to experiments described in Section 5.2 of Chapter 5



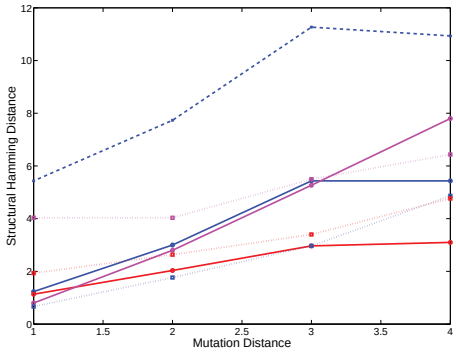
(a) Exact distance vs. mutation distance



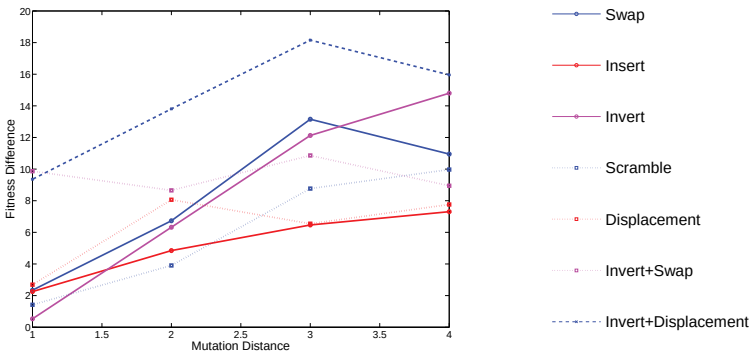
(b) Edit distance vs. mutation distance



(c) KTD vs. mutation distance

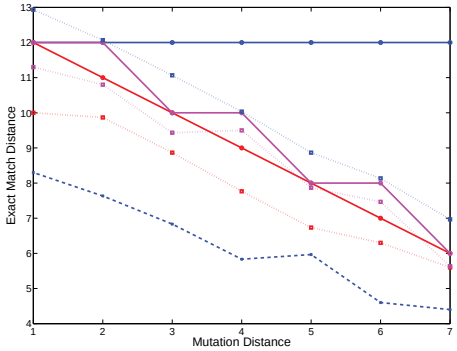


(d) SHD vs. mutation distance

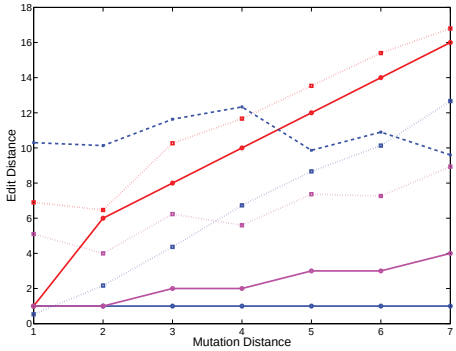


(e) Fitness difference vs. mutation distance

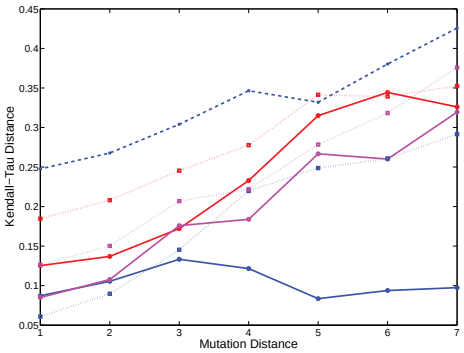
Figure E.1: Effect of choice of mutation on produced solution on *asia*



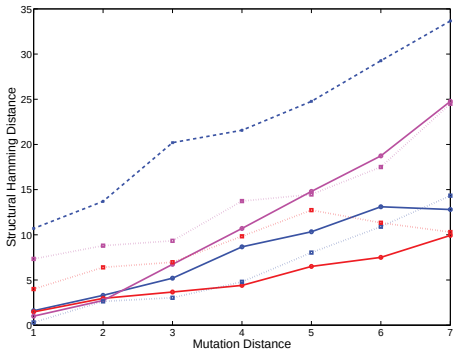
(a) Exact distance vs. mutation distance



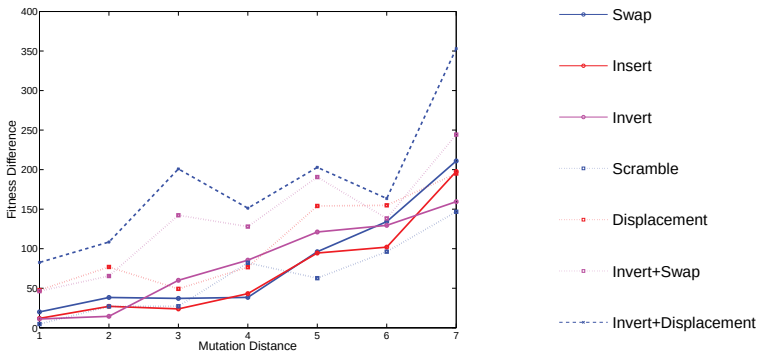
(b) Edit distance vs. mutation distance



(c) KTD vs. mutation distance

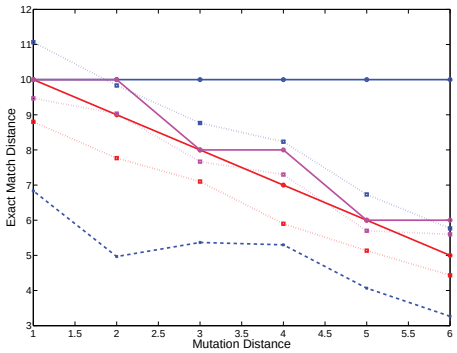


(d) SHD vs. mutation distance

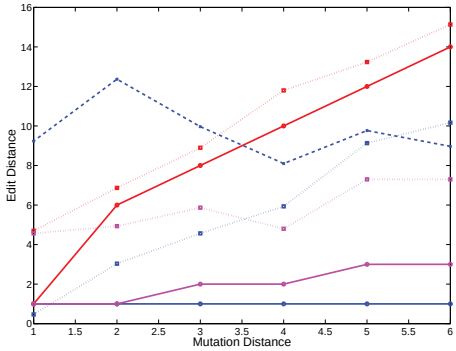


(e) Fitness difference vs. mutation distance

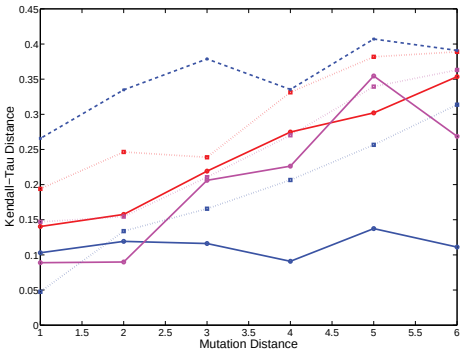
Figure E.2: Effect of choice of mutation on produced solution on *tank*



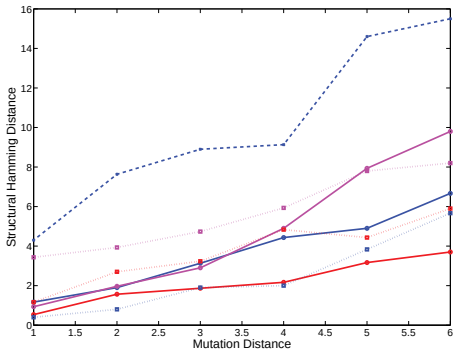
(a) Exact distance vs. mutation distance



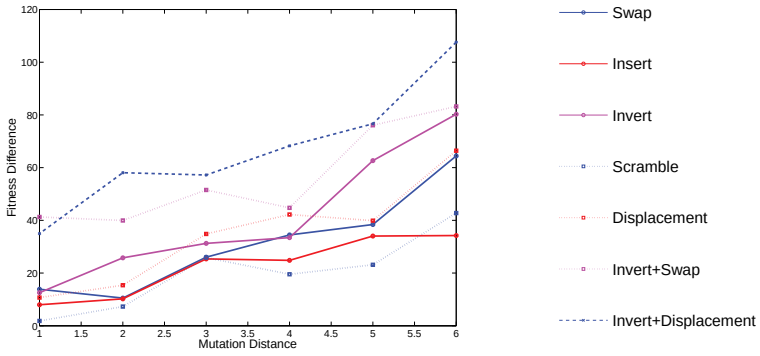
(b) Edit distance vs. mutation distance



(c) KTD vs. mutation distance

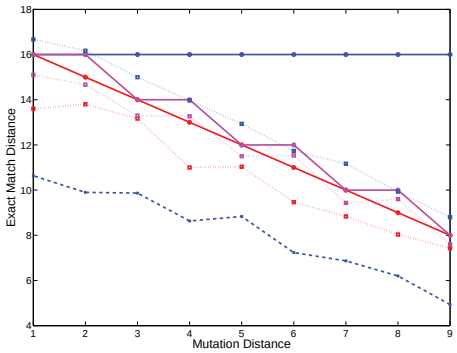


(d) SHD vs. mutation distance

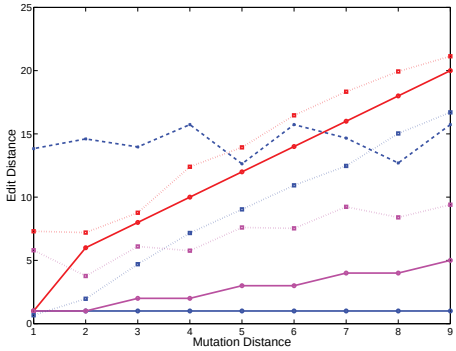


(e) Fitness difference vs. mutation distance

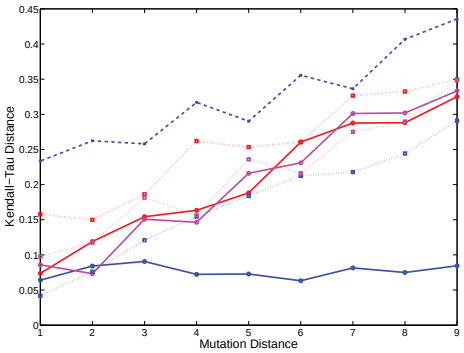
Figure E.3: Effect of choice of mutation on produced solution on *credit*



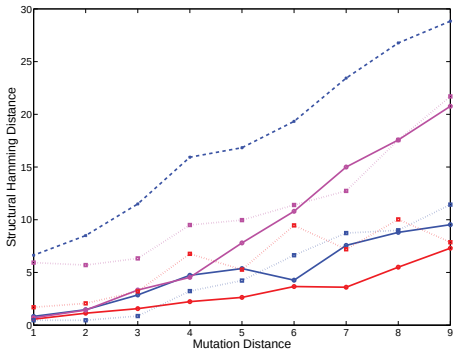
(a) Exact distance vs. mutation distance



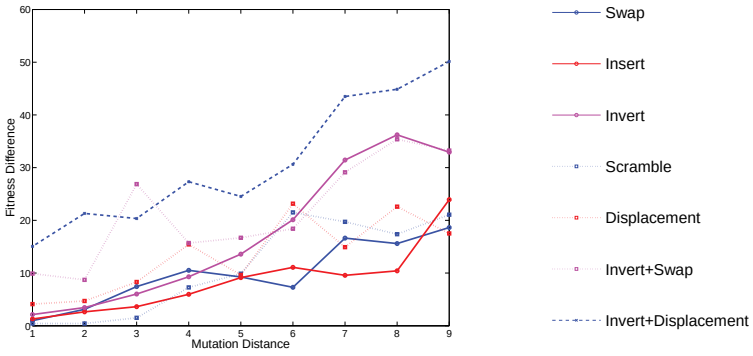
(b) Edit distance vs. mutation distance



(c) KTD vs. mutation distance

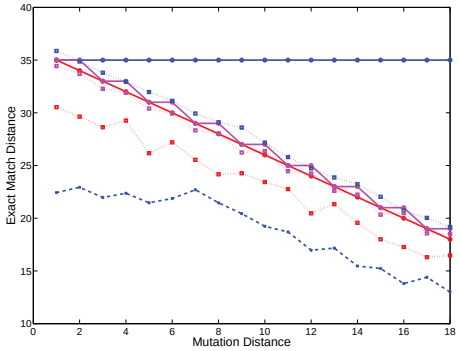


(d) SHD vs. mutation distance

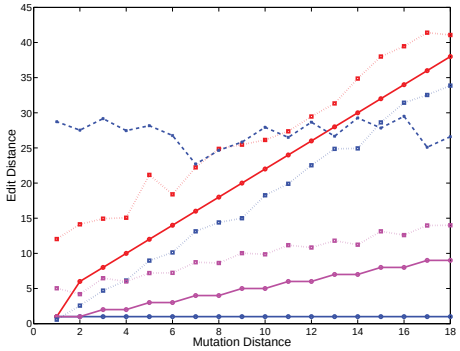


(e) Fitness difference vs. mutation distance

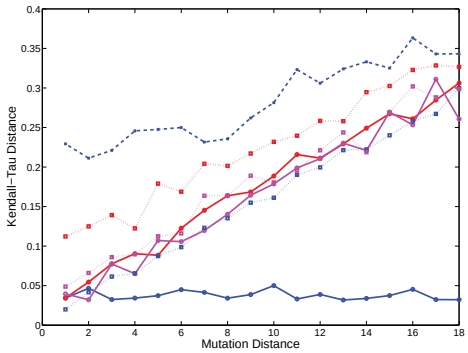
Figure E.4: Effect of choice of mutation on produced solution on *car*



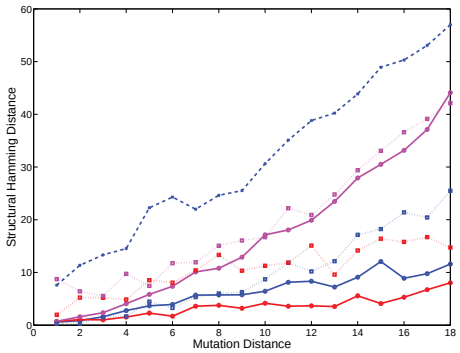
(a) Exact distance vs. mutation distance



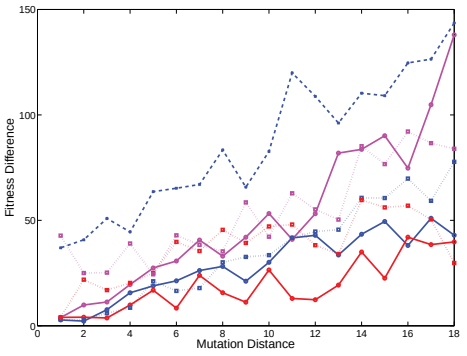
(b) Edit distance vs. mutation distance

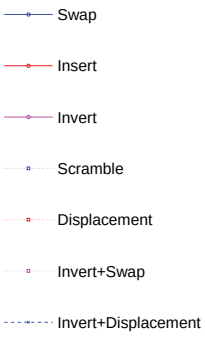


(c) KTD vs. mutation distance



(d) SHD vs. mutation distance





(e) Fitness difference vs. mutation distance

Figure E.5: Effect of choice of mutation on produced solution on *alarm*

Appendix F

BAUS data distribution across NHS Cancer Networks

This set of figures relates to experiments described in Section 6.4 of Chapter 6

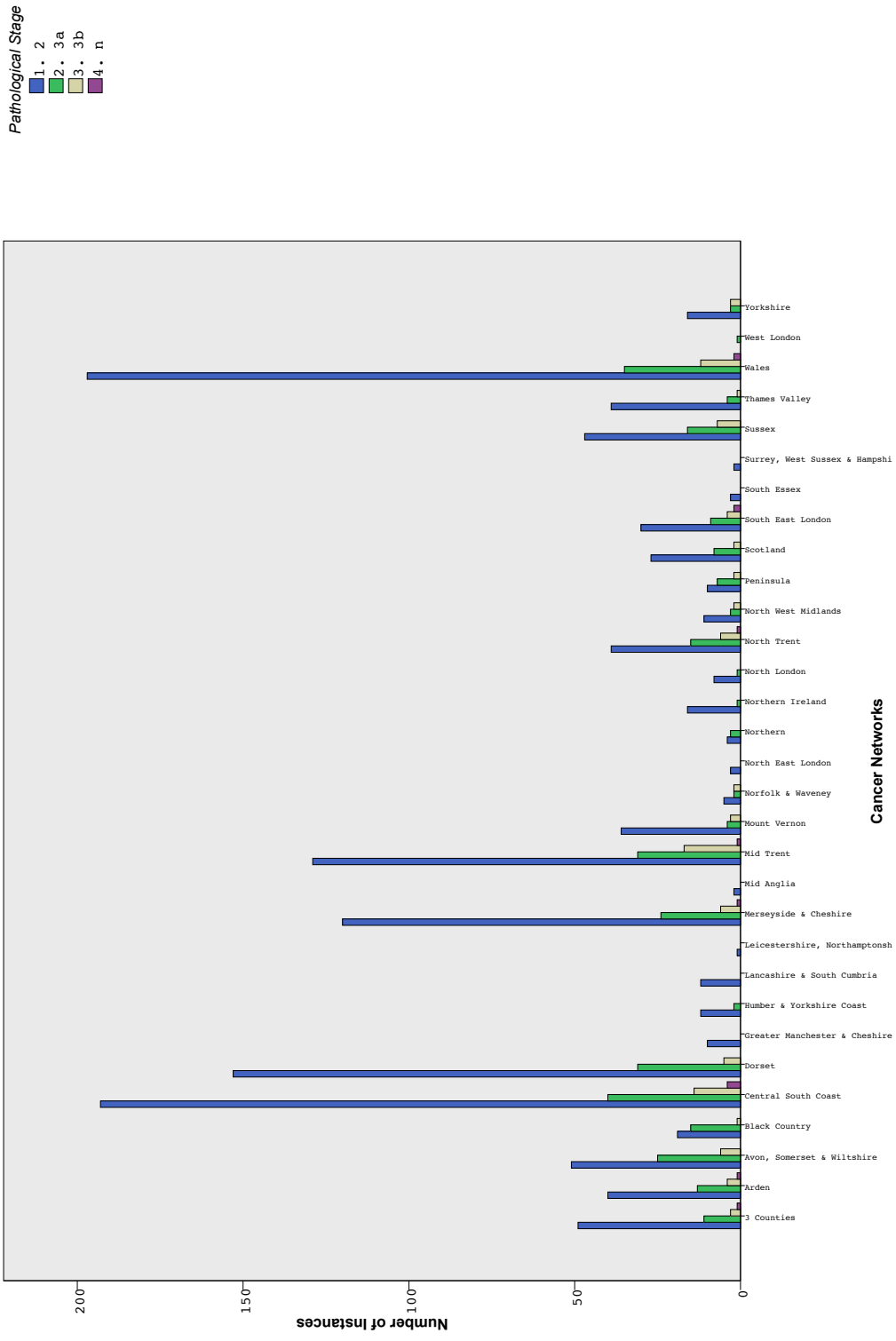
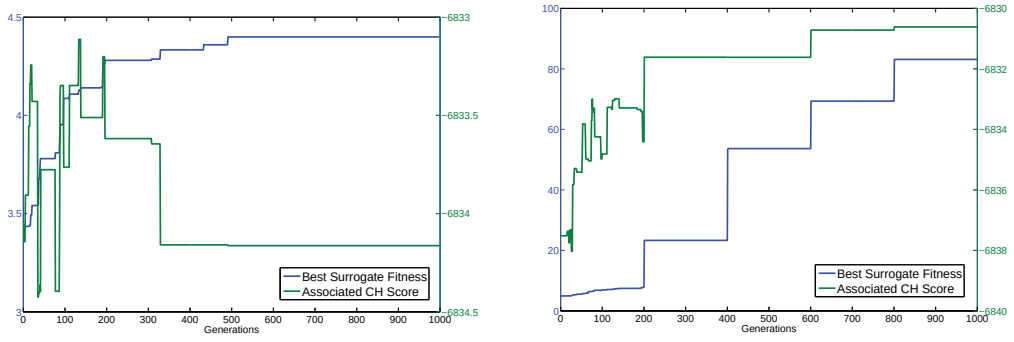


Figure F.1: Distribution of the pathological stage by cancer network

Appendix G

Use of k-order marginal fitness surrogate model for BN structure learning

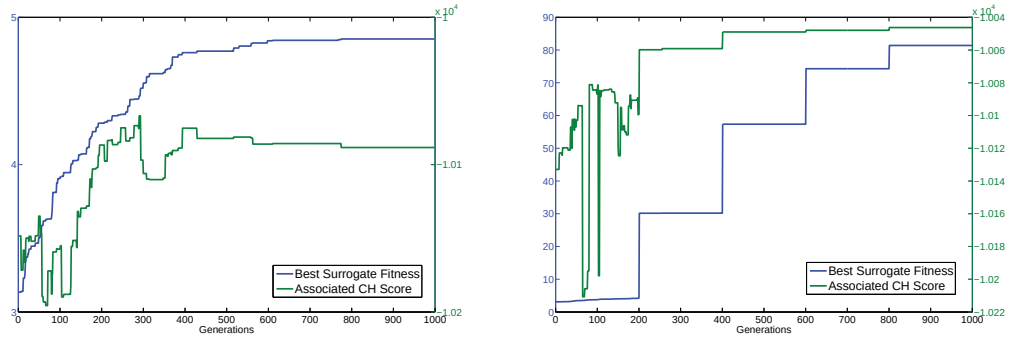
This set of figures relates to experiments described in Section 7.2.4 of Chapter 7



(a) No Update

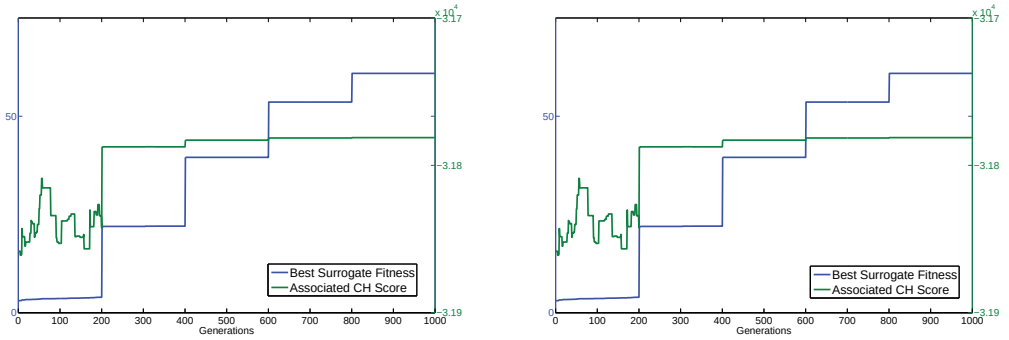
(b) Model Update

Figure G.1: Evolution of surrogate fitness of the best solution and its associated CH score on *asia* with and without model update



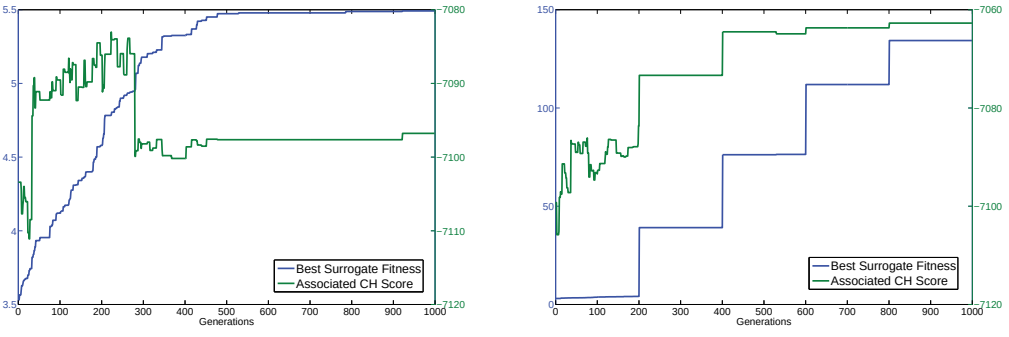
(a) No Update (b) Model Update

Figure G.2: Evolution of surrogate fitness of the best solution and its associated CH score on *tank* with and without model update



(a) No Update (b) Model Update

Figure G.3: Evolution of surrogate fitness of the best solution and its associated CH score on *credit* with and without model update



(a) No Update (b) Model Update

Figure G.4: Evolution of surrogate fitness of the best solution and its associated CH score on *car* with and without model update

Bibliography

Aha, D., Kibler, D. and Albert, M. (1991), "Instance-based learning algorithms", *Machine learning* , Vol. 6, Springer, pp. 37–66.

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

Anagnostou, T., Remzi, M., Lykourinas, M. and Djavan, B. (2003), "Artificial neural networks for decision-making in urologic oncology", *European Urology* , Vol. 43, Elsevier, pp. 596–603.

Anastasiadis, A., Lichy, M., Nagele, U., Kuczyk, M., Merseburger, A., Hennenlotter, J., Corvin, S., Sievert, K., Claussen, C., Stenzl, A. et al. (2006), "MRI-guided biopsy of the prostate increases diagnostic performance in men with elevated or increasing PSA levels after previous negative TRUS biopsies", *European Urology* , Vol. 50, Elsevier, pp. 738–749.

Augustin, H., Eggert, T., Wenske, S., Karakiewicz, P., Palisaar, J., Daghofer, F., Huland, H. and Graefen, M. (2004), "Comparison of accuracy between the Partin tables of 1997 and 2001 to predict final pathological stage in clinically localized prostate cancer", *The Journal of Urology* , Vol. 171, Elsevier, pp. 177–181.

Back, T. (1994), "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms", *Proceedings of the first IEEE Conference on Evolutionary Computation* , pp. 57–62.

Back, T., Hammel, U. and Schwefel, H. (1997), "Evolutionary computation: Comments on the history and current state", *IEEE Transactions on Evolutionary computation* , Vol. 1, IEEE, pp. 3–17.

Balaprakash, P., Birattari, M. and Stützle, T. (2007), "Improvement strategies for the f-race algorithm: Sampling design and iterative refinement", *Hybrid Metaheuristics* , Springer, pp. 108–122.

Baluja, S. (1994), Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning, Technical report, Carnegie Mellon University.

Beinlich, I., Suermondt, G., Chavez, R. and Cooper, G. (1989), "The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks", *Proceedings of the second European Conference on Artificial Intelligence in Medicine* , Springer-Verlag, pp. 247–256.

Bhojani, N., Ahyai, S., Graefen, M., Capitanio, U., Suardi, N., Shariat, S., Jeldres, C., Erbersdobler, A., Schlomm, T., Haese, A. et al. (2009), "Partin Tables cannot accurately predict the pathological stage at radical prostatectomy", *European Journal of Surgical Oncology* , Vol. 35, Elsevier, pp. 123–128.

Bhojani, N., Salomon, L., Capitanio, U., Suardi, N., Shariat, S., Jeldres, C., Zini, L., Pharand, D., Péloquin, F., Arjane, P. et al. (2009), "External validation of the updated partin tables in a cohort of French and Italian men", *International Journal of Radiation Oncology, Biology, Physics* , Vol. 73, Elsevier, pp. 347–352.

Blute, M., Bergstralh, E., Partin, A., Walsh, P., Kattan, M., Scardino, P., Montie, J., Pearson, J., Slezak, J. and Zincke, H. (2000), "Validation of Partin tables for predicting pathological stage of clinically localized prostate cancer", *The Journal of Urology* , Vol. 164, Elsevier, pp. 1591–1595.

Boerlage, B. (1992), Link strength in bayesian networks, Master's thesis, University of British Columbia.

Borque, A., Sanz, G., Allepuz, C., Plaza, L., Gil, P. and Rioja, L. (2001), "The use of neural networks and logistic regression analysis for predicting pathological stage in men undergoing radical prostatectomy: a population based study", *The Journal of Urology* , Vol. 166, Elsevier, pp. 1672–1678.

Bott, S., Emberton, M. and Sydes, M. (2008), "Prostate cancer staging tables - A predictive model for the UK", *British Journal of Medical and Surgical Urology* , Vol. 1, Elsevier, pp. 107–119.

Brain, D., Webb, G., Richards, D., Beydoun, G., Hoffmann, A. and Compton, P. (1999), "On the effect of data set size on bias and variance in classification learning", *Proceedings of the fourth Australian Knowledge Acquisition Workshop* , University of New South Wales, pp. 117–128.

Breiman, L. (2001), "Random forests", *Machine Learning* , Vol. 45, Springer, pp. 5–32.

Broomhead, D., Lowe, D., Signals, R. and Malvern, R. (1988), Radial basis functions, multi-variable functional interpolation and adaptive networks, Technical report, Royal Signals and Radar Establishment, Malvern, UK.

Brown, L. E., Tsamardinos, I. and Aliferis, C. F. (2005), "A comparison of novel and state-of-the-art polynomial bayesian network learning algorithms", *Proceedings of the twentieth National Conference on Artificial Intelligence* , AAAI, pp. 739–745.

Brownlee, A., Regnier-Coudert, O., McCall, J. and Massie, S. (2010), "Using a markov network as a surrogate fitness function in a genetic algorithm", *Proceedings of the IEEE Congress on Evolutionary Computing* , IEEE, pp. 4525–4532.

Brownlee, A., Regnier-Coudert, O., McCall, J., Massie, S. and Stulajter, S. (2012), "An application of

a GA with markov network surrogate to feature selection", *International Journal of Systems Science* , Taylor & Francis, pp. 1–18.

Buntine, W. (1991), "Theory refinement on bayesian networks", *Proceedings of the seventh Conference on Uncertainty in Artificial Intelligence* , Morgan Kaufmann, pp. 52–60.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. and Schulenburg, S. (2003), "Hyper-heuristics: An emerging direction in modern search technology", *Handbook of metaheuristics* , Springer, pp. 457–474.

Cano, R., Sordo, C. and Gutiérrez, J. M. (2004), "Applications of bayesian networks in meteorology", *Studies in Fuzziness And Soft Computing* , Vol. 146, Springer, pp. 309–328.

Cantu-Paz, E. (1995), A Summary of Research on Parallel Genetic Algorithms, Technical report, Illinois Genetic Algorithms Laboratory, Urbana-Champaign, IL, USA.

Carvalho, A. (2011), "A cooperative coevolutionary genetic algorithm for learning bayesian network structures", *Proceedings of the thirteenth Conference on Genetic and Evolutionary Computation* , ACM, pp. 1131–1138.

Carvalho, A., Oliveira, A. and Sagot, M. (2007), "Efficient Learning of Bayesian Network Classifiers: An extension to the TAN classifier", *Proceedings of the twentieth Australian joint Conference on Advances in Artificial Intelligence* , Springer, pp. 16–25.

Carvalho, A., Roos, T., Oliveira, A., Myllymäki, P. et al. (2011), "Discriminative learning of bayesian networks via factorized conditional log-likelihood", *Journal of Machine Learning Research* , Vol. 12, MIT Press, pp. 2181–2210.

Ceberio, J., Irurozki, E., Mendiburu, A. and Lozano, J. (2012), "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems", *Progress in Artificial Intelligence* , Vol. 1, Springer, pp. 103–117.

Ceberio, J., Mendiburu, A. and Lozano, J. (2011*a*), "Introducing the mallows model on estimation of distribution algorithms", *Proceedings of the eighteenth International Conference on Neural Information Processing* , Springer, pp. 461–470.

Ceberio, J., Mendiburu, A. and Lozano, J. (2011*b*), "A preliminary study on EDAs for permutation problems based on marginal-based models", *Proceedings of the thirteenth annual Conference on Genetic and Evolutionary Computation* , ACM, pp. 609–616.

Chen, R., Sivakumar, K. and Kargupta, H. (2004), "Collective mining of bayesian networks from

distributed heterogeneous data", *Knowledge and Information Systems* , Vol. 6, Springer, pp. 164–187.

Chen, S., Chang, P., Cheng, T. and Zhang, Q. (2012), "A self-guided genetic algorithm for permutation flowshop scheduling problems", *Computers & Operations Research* , Vol. 39, Elsevier, pp. 1450–1457.

Cheng, J., Greiner, R., Kelly, J., Bell, D. and Liu, W. (2002), "Learning bayesian networks from data: An information-theory based approach", *Artificial Intelligence* , Vol. 137, pp. 43–90.

Chickering, D. (1995), "A transformational characterization of equivalent bayesian network structures", *Proceedings of the eleventh Conference on Uncertainty in Artificial Intelligence* , Morgan Kaufmann, pp. 87–98.

Chow, C. and Liu, C. (1968), "Approximating discrete probability distributions with dependence trees", *IEEE transactions on Information Theory* , Vol. 14, IEEE, pp. 462–467.

Cios, K. and William Moore, G. (2002), "Uniqueness of medical data mining", *Artificial Intelligence in Medicine* , Vol. 26, Elsevier, pp. 1–24.

Cobb, H. and Grefenstette, J. (1993), "Genetic algorithms for tracking changing environments", *Proceedings of the fifth International Conference on Genetic Algorithms* , Morgan Kaufmann, pp. 523–530.

Collins, G., Lee, R., McKelvie, G., Rogers, A. and Hehir, M. (1993), "Relationship between prostate specific antigen, prostate volume and age in the benign prostate", *British Journal of Urology* , Vol. 71, Wiley, pp. 445–450.

Colorni, A., Dorigo, M., Maniezzo, V. et al. (1991), "Distributed optimization by ant colonies", *Proceedings of the first European Conference on Artificial Life* , MIT Press, pp. 134–142.

Cooper, G. and Herskovits, E. (1992), "A bayesian method for the induction of probabilistic networks from data", *Machine Learning* , Vol. 9, Springer, pp. 309–347.

Cotta, C. and Muruzábal, J. (2004), "On the learning of bayesian network graph structures via evolutionary programming", *Proceedings of the second European Workshop on Probabilistic Graphical Models* , Leiden University, pp. 65–72.

Cover, T. M. and Thomas, J. A. (1991), "Entropy, relative entropy and mutual information", *In Elements of Information Theory* , Wiley, pp. 12–49.

Cozman, F. (2000), "Generalizing variable elimination in Bayesian networks", *Workshop on Probabilistic Reasoning in Artificial Intelligence* , Editora Tecart, pp. 27–32.

Cruz, J. and Wishart, D. (2006), "Applications of machine learning in cancer prediction and prognosis", *Cancer Informatics* , Vol. 2, Libertas Academica, pp. 59–77.

De Bonet, J., Isbell, C. and Viola, P. (1996), "MIMIC: Finding optima by estimating probability densities", *Advances in Neural Information Processing Systems* , Morgan Kaufmann, pp. 424–430.

De Campos, L. M., Fernandez-Luna, J. M., Gámez, J. A. and Puerta, J. M. (2002), "Ant colony optimization for learning bayesian networks", *International Journal of Approximate Reasoning* , Vol. 31, Elsevier, pp. 291–311.

De Jongh, M. and Druzdzel, M. (2009), "A comparison of structural distance measures for causal bayesian network models", *Proceedings of International Joint Conference on Intelligent Information Systems* , pp. 443–456.

De la Ossa, L., Gámez, J. and Puerta, J. (2004), "Migration of probability models instead of individuals: an alternative when applying the island model to EDAs", *Parallel Problem Solving from Nature VIII* , Springer, pp. 242–252.

Deep, K. and Mebrahtu, H. (2011), "Combined mutation operators of genetic algorithm for the travelling salesman problem", *International Journal of Combinatorial Optimization Problems and Informatics* , Vol. 2, International Journal of Combinatorial Optimization Problems and Informatics, pp. 1–23.

Dempster, A., Laird, N. and Rubin, D. (1977), "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society. Series B (Methodological)* , JSTOR, pp. 1–38.

Deng, H., Runger, G. and Tuv, E. (2011), "Bias of importance measures for multi-valued attributes and solutions", *Prodeedings of the twenty-first International Conference on Artificial Neural Networks* , Springer, pp. 293–300.

Dos Santos, E., Hruschka, E. and Ebecken, N. (2010), "A distance-based mutation operator for learning bayesian network structures using evolutionary algorithms", *Proceedings of the IEEE Congress on Evolutionary Computation* , IEEE, pp. 1–8.

Dunn, O. J. (1961), "Multiple comparisons among means", *Journal of the American Statistical Association* , Vol. 56, Taylor & Francis, pp. 52–64.

Eskicorapci, S., Karabulut, E., Turkeri, L., Baltaci, S., Cal, C., Toktas, G., Akpinar, H., Ozer, G., Sozen, S., Tokuc, R. et al. (2005), "Validation of 2001 Partin tables in Turkey: a multicenter study", *European Urology* , Vol. 47, Elsevier, pp. 185–189.

Etxeberria, R. and Larranaga, P. (1999), "Global optimization using bayesian networks", *Proceedings of the second Symposium on Artificial Intelligence* , pp. 332–339.

Fogel, L., Owens, A. and Walsh, M. (1966), *Artificial intelligence through simulated evolution*, Wiley.

Fournier, F., McCall, J., Petrovski, A. and Barclay, P. (2010), "Evolved bayesian network models of rig operations in the gulf of Mexico", *Proceedings of the IEEE Congress on Evolutionary Computation* , IEEE, pp. 1–7.

Friedman, N., Geiger, D. and Goldszmidt, M. (1997), "Bayesian network classifiers", *Machine Learning* , Vol. 29, Springer, pp. 131–163.

Galfano, A., Novara, G., Iafrate, M., Cavalleri, S., Martignoni, G., Gardiman, M., D'Elia, C., Patard, J., Artibani, W. and Ficarra, V. (2008), "Mathematical models for prognostic prediction in patients with renal cell carcinoma", *Urologia Internationalis* , Vol. 80, Karger, pp. 113–123.

Gao, X., Ren, S., Lu, X., Xu, C. and Sun, Y. (2008), "The Newer the Better? Comparison of the 1997 and 2001 Partin Tables for Pathologic Stage Prediction of Prostate Cancer in China", *Urology* , Vol. 72, Elsevier, pp. 1096–1101.

George, E., Makov, U. and Smith, A. (1993), "Conjugate likelihood distributions", *Scandinavian Journal of Statistics* , Vol. 20, JSTOR, pp. 147–156.

Goldberg, D., Deb, K. and Clark, J. (1991), Genetic algorithms, noise, and the sizing of populations, Technical report, University of Illinois.

Graefen, M., Augustin, H., Karakiewicz, P., Hammerer, P., Haese, A., Palisaar, J., Blonski, J., Fernandez, S., Erbersdobler, A. and Huland, H. (2003), "Can predictive models for prostate cancer patients derived in the United States of America be utilized in European patients? A validation study of the Partin tables", *European Urology* , Vol. 43, Elsevier, pp. 6–11.

Hall, J., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. (2009), "The WEKA data mining software: An update", *SIGKDD Explorations Newsletter* , Vol. 11, ACM, pp. 11–18.

Han, M., Snow, P., Brandt, J. and Partin, A. (2001), "Evaluation of artificial neural networks for the prediction of pathological stage in prostate carcinoma", *Cancer* , Vol. 91, American Cancer Society, pp. 1661–1666.

Harrell Jr, F., Califf, R., Pryor, D., Lee, K. and Rosati, R. (1982), "Evaluating the yield of medical tests", *Journal of the American Medical Association* , Vol. 247, American Medical Association, pp. 2543–2546.

Heckerman, D. (2008), "A tutorial on learning with Bayesian networks", *Innovations in Bayesian Networks* , Springer, pp. 33–82.

Heckerman, D., Geiger, D. and Chickering, D. (1995), "Learning bayesian networks: the combination of knowledge and statistical data", *Machine Learning* , Vol. 20, Springer, pp. 197–243.

Henrion, M. (1986), "Propagating uncertainty in bayesian networks by probabilistic logic sampling", *Proceedings of the second annual Conference on Uncertainty in Artificial Intelligence* , Elsevier, pp. 149–164.

Holland, J. (1975), *Adaptation in natural and artificial systems*, MIT press.

Hordijk, W. and Kauffman, S. (2005), "Correlation analysis of coupled fitness landscapes", *Complexity* , Vol. 10, Wiley, pp. 41–49.

Hosmer, D. and Lemeshow, S. (2000), *Applied logistic regression*, Wiley-Interscience.

Hsu, W., Guo, H., Perry, B. and Stilson, J. (2002), "A permutation genetic algorithm for variable ordering in learning bayesian networks from data", *Proceedings of the Genetic and Evolutionary Computation Conference* , Morgan Kaufmann, pp. 383–390.

Iclănzan, D. and Dumitrescu, D. (2010), "Graph clustering based model building", *Parallel Problem Solving from Nature XI* , Springer, pp. 506–515.

Jensen, F. and Nielsen, T. (2007), *Bayesian networks and decision graphs*, Springer Verlag.

Jerrum, M. (1985), "The complexity of finding minimum-length generator sequences", *Theoretical Computer Science* , Vol. 36, Elsevier, pp. 265–289.

Jones, T. and Forrest, S. (1995), "Fitness distance correlation as a measure of problem difficulty for genetic algorithms", *Proceedings of the sixth International Conference on Genetic Algorithms* , Morgan Kaufmann, pp. 184–192.

Kabli, R., Herrmann, F. and McCall, J. (2007), "A chain-model genetic algorithm for bayesian network structure learning", *Proceedings of the ninth Conference on Genetic and Evolutionary Computation* , ACM, pp. 1271–1278.

Kabli, R., McCall, J., Herrmann, F. and Ong, E. (2008), "Evolved bayesian networks as a versatile alternative to partin tables for prostate cancer management", *Proceedings of the tenth Conference on Genetic and Evolutionary Computation* , ACM, pp. 1547–1554.

Kahn, A. (1962), "Topological sorting of large networks", *Communications of the ACM* , Vol. 5, ACM, pp. 558–562.

Karakiewicz, P., Lattouf, J., Perrotte, P., Valiquette, L., Benard, F., McCormack, M., Menard, C., Lebeau, T., Benayoun, S. and Ramirez, A. (2005), "Validation of 1997 Partin Tables lymph node invasion predictions in men treated with radical prostatectomy in Montreal Quebec.", *The Canadian Journal of Urology* , Vol. 12, The Canadian Journal of Urology, pp. 2588–2592.

Kattan, M. (2006), "Validating a prognostic model", *Cancer* , Vol. 107, American Cancer Society, pp. 2523–2524.

Kattan, M., Stapleton, A., Wheeler, T. and Scardino, P. (1997), "Evaluation of a nomogram used to predict the pathologic stage of clinically localized prostate carcinoma", *Cancer* , Vol. 79, American Cancer Society, pp. 528–537.

Kauffman, S. A. (1989), "Adaptation on rugged fitness landscapes", *Lectures in the Sciences of Complexity* , Vol. 1, pp. 527–618.

Kawakami, S., Numao, N., Okubo, Y., Koga, F., Yamamoto, S., Saito, K., Fujii, Y., Yonese, J., Masuda, H., Kihara, K. et al. (2008), "Development, validation, and head-to-head comparison of logistic regression-based nomograms and artificial neural network models predicting prostate cancer on initial extended biopsy", *European Urology* , Vol. 54, Elsevier, pp. 601–611.

Kirby, R. (2002), *The prostate: small gland, big problem*, Prostate Research Campaign UK.

Kirkpatrick, S., Gelatt Jr, C. and Vecchi, M. (1983), "Optimization by simulated annealing", *Science* , Vol. 220, American Association for the Advancement of Science, pp. 671–680.

Kohavi, R. (1995), "A study of cross-validation and bootstrap for accuracy estimation and model selection", *Proceedings of the fourteenth International Joint Conference on Artificial Intelligence* , AAAI, pp. 1137–1145.

Koller, D., Friedman, N., Getoor, L. and Taskar, B. (2007), "Graphical models in a nutshell", *In Introduction to statistical relational learning* , MIT Press, pp. 13–55.

Kruskal, J. (1956), "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical society* , Vol. 7, JSTOR, pp. 48–50.

Lam, W. and Bacchus, F. (1994), "Learning bayesian belief networks: An approach based on the MDL principle", *Computational intelligence* , Vol. 10, Wiley, pp. 269–293.

Lam, W. and Segre, A. (1997), "Distributed data mining of probabilistic knowledge", *Proceedings of the seventeenth International Conference on Distributed Computing Systems* , IEEE, pp. 178–185.

Langley, P., Iba, W. and Thompson, K. (1992), "An analysis of Bayesian classifiers", *Proceedings of the tenth National Conference on Artificial Intelligence* , AAAI Press, pp. 223–228.

Larrañaga, P. (2010), "Probabilistic graphical models and evolutionary computation", *Plenary and Invited Lectures of the IEEE World Congress on Computational Intelligence* , IEEE, pp. 23–54.

Larrañaga, P., Kuijpers, C., Murga, R. and Yurramendi, Y. (1996), "Learning bayesian network structures by searching for the best ordering with genetic algorithms", *IEEE Transactions on Systems, Man and Cybernetics* , Vol. 26, IEEE, pp. 487–493.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. and Kuijpers, C. (1996), "Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters", *IEEE Transactions on Pattern Analysis and Machine Intelligence* , Vol. 18, IEEE, pp. 912–926.

Lauritzen, S. and Spiegelhalter, D. (1988), "Local computations with probabilities on graphical structures and their application to expert systems", *Journal of the Royal Statistical Society. Series B (Methodological)* , JSTOR, pp. 157–224.

Lee, J., Chung, W. and Kim, E. (2008), "Structure learning of bayesian networks using dual genetic algorithm", *IEICE Transactions on Information and Systems* , Vol. 91, Oxford Journals, pp. 32–43.

Makarov, D., Trock, B., Humphreys, E., Mangold, L., Walsh, P. and Epstein, J. (2007), "Updated nomogram to predict pathologic stage of prostate cancer given prostate-specific antigen level, clinical stage, and biopsy Gleason score (Partin tables) based on cases from 2000 to 2005", *Urology* , Vol. 69, Elsevier, pp. 1095–1101.

Marmion, M., Jourdan, L. and Dhaenens, C. (2012), "Fitness landscape analysis and metaheuristics efficiency", *Journal of Mathematical Modelling and Algorithms* , Springer, pp. 1–24.

Matsui, Y., Egawa, S., Tsukayama, C., Terai, A., Kuwao, S., Baba, S. and Arai, Y. (2002), "Artificial neural network analysis for predicting pathological stage of clinically localized prostate cancer in the japanese population", *Japanese Journal of Clinical Oncolology* , Vol. 32, Oxford Journals, pp. 530–535.

Mendiburu, A., Lozano, J. and Miguel-Alonso, J. (2005), "Parallel implementation of EDAs based on probabilistic graphical models", *IEEE Transactions on Evolutionary Computation* , Vol. 9, IEEE, pp. 406–423.

Metz, C. (1978), "Basic principles of ROC analysis", *Seminars in nuclear medicine* , Vol. 8, Elsevier, pp. 283–298.

Mühlenbein, H. and Paaß, G. (1996), "From recombination of genes to the estimation of distributions I. Binary parameters", *Parallel Problem Solving from Nature IV* , Springer, pp. 178–187.

Neapolitan, R. (2004), *Learning bayesian networks*, Pearson Prentice Hall Series in Artificial Intelligence.

Nicholson, A., Twardy, C., Korb, K. and Hope, L. (2008), "Decision support for clinical cardiovascular risk assessment", *In Bayesian Networks: A Practical Guide to Applications* , Wiley, pp. 33–52.

Nielsen, J., Kocka, T. and Pena, J. (2003), "On local optima in learning bayesian networks", *Proceedings of the nineteenth Conference in Uncertainty in Artificial Intelligence* , Morgan Kaufmann, pp. 435–442.

Ocenasek, J. and Schwarz, J. (2001), "The distributed bayesian optimization algorithm for combinatorial optimization", *Proceedings of Evolutionary Methods for Design, Optimisation and Control* , CIMNE, pp. 115–120.

Oliver, I., Smith, D. and Holland, J. R. (1987), "A study of permutation crossover operators on the traveling salesman problem", *Proceedings of the Second International Conference on Genetic Algorithms and their Application* , L. Erlbaum Associates Inc., pp. 224–230.

Oniśko, A. (2008), "Medical diagnosis", *In Bayesian Networks: A practical guide to applications* , Wiley, pp. 15–32.

Oteniya, L. (2008), Bayesian belief networks for dementia diagnosis and other applications: a comparison of hand-crafting and construction using a novel data driven technique, PhD thesis, University of Stirling.

Park, B., Kargupta, H., Johnson, E., Sanseverino, E., Hershberger, D. and Silvestre, L. (2002), "Distributed, collaborative data analysis from heterogeneous sites using a scalable evolutionary technique", *Applied Intelligence* , Vol. 16, Springer, pp. 19–42.

Parpinelli, R. S., Lopes, H. S. and Freitas, A. A. (2002), "Data mining with an ant colony optimization algorithm", *IEEE Transactions on Evolutionary Computation* , Vol. 6, IEEE, pp. 321–332.

Partin, A., Kattan, M., Subong, E., Walsh, P., Wojno, K. and Oesterling, J. (1997), "Combination of prostate-specific antigen, clinical stage, and Gleason score to predict pathological stage of localized prostate cancer: A multi-institutional update", *Journal of the American Medical Association* , Vol. 277, American Medical Association, pp. 1445–1451.

Partin, A., Mangold, L., Lamm, D., Walsh, P., Epstein, J. and Pearson, J. (2001), "Contemporary

update of prostate cancer staging nomograms (Partin Tables) for the new millennium", *Urology* , Vol. 58, Elsevier, pp. 843–848.

Partin, A., Yoo, J., Carter, H., Pearson, J., Chan, D., Epstein, J. and Walsh, P. (1993), "The use of prostate specific antigen, clinical stage and Gleason score to predict pathological stage in men with localized prostate cancer.", *The Journal of Urology* , Vol. 150, American Urological Association, pp. 110–114.

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

Pelikan, M. (2005), "Bayesian optimization algorithm", *In Hierarchical Bayesian Optimization Algorithm* , Springer, pp. 31–48.

Penson, D., Grossfeld, G., Li, Y., Henning, J., Lubeck, D. and Carroll, P. (2002), "How well does the Partin nomogram predict pathological stage after radical prostatectomy in a community based population? Results of the cancer of the prostate strategic urological research endeavor", *The Journal of Urology* , Vol. 167, American Urological Association, pp. 1653–1658.

Platt, J. (1999), "Fast training of support vector machines using sequential minimal optimization", *In Advances in Kernel Methods* , MIT press, pp. 185–208.

Pourret, O., Naim, P. and Marcot, B. (2008), *Bayesian networks: a practical guide to applications*, Wiley.

Ramakrishnan, N., Hanauer, D. and Keller, B. (2010), "Mining electronic health records", *Computer* , Vol. 43, IEEE, pp. 77–81.

Robinson, R. (1977), "Counting unlabeled acyclic digraphs", *Proceedings of the fifth Australian Conference on Combinatorial Mathematics* , Springer, pp. 28–43.

Romero, T., Larranaga, P. and Sierra, B. (2004), "Learning bayesian networks in the space of orderings with estimation of distribution algorithms", *International Journal of Pattern Recognition and Artificial Intelligence* , Vol. 18, World Scientific, pp. 607–626.

Rumelhart, D., Hinton, G. and Williams, R. (1985), Learning internal representations by error propagation, Technical report, University of California San Diego.

Sanchez-Chapado, M., Angulo, J., Ibarburen, C., Aguado, F., Ruiz, A., Viano, J., Garcia-Segura, A., Gonzalez-Esteban, J. and Rodriguez-Vallejo, J. (1997), "Comparison of digital rectal examination, transrectal ultrasonography, and multicoil magnetic resonance imaging for preoperative evaluation of prostate cancer", *European urology* , Vol. 32, Elsevier, pp. 140–149.

Santana, R. (2011), "Estimation of distribution algorithms: from available implementations to potential developments", *Proceedings of the thirteenth Conference on Genetic and Evolutionary Computation* , ACM, pp. 679–686.

Schiavinotto, T. and Stützle, T. (2007), "A review of metrics on permutations for search landscape analysis", *Computers & Operations Research* , Vol. 34, Elsevier, pp. 3143–3153.

Sevaux, M., Sörensen, K. et al. (2005), "Permutation distance measures for memetic algorithms with population management", *Proceedings of 6th Metaheuristics International Conference* , pp. 1–8.

Shaida, N. and Malone, P. (2007), "Controversial topics in surgery. open versus laparoscopic radical prostatectomy: The case for open radical prostatectomy.", *Annals of The Royal College of Surgeons of England* , Vol. 89, The Royal College of Surgeons of England, pp. 108–110.

Shakya, S. and McCall, J. (2007), "Optimization by estimation of distribution with deum framework based on markov random fields", *International Journal of Automation and Computing* , Vol. 4, Springer, pp. 262–272.

Skolicki, Z. and De Jong, K. (2004), "Improving evolutionary algorithms with multi-representation island models", *Parallel Problem Solving from Nature VIII* , Springer, pp. 420–429.

Sobin, L. (2009), *TNM classification of malignant tumours*, Wiley-Blackwell.

Song, C., Kang, T., Ro, J., Lee, M., Kim, C. and Ahn, H. (2005), "Nomograms for the prediction of pathologic stage of clinically localized prostate cancer in Korean men", *Journal of Korean Medical Science* , Vol. 20, The Korean Academy of Medical Science, pp. 262–266.

Spirtes, P., Glymour, C. and Scheines, R. (2000), *Causation, prediction, and search*, MIT Press.

Sutherland, W. A. (2009), *Introduction to metric and topological spaces.*, Oxford University Press.

Sy, B. K. (1992), "Reasoning MPE to multiply connected belief networks using message passing", *Proceedings of the National Conference on Artificial Intelligence* , Wiley, pp. 570–576.

Tarjan, R. (1976), "Edge-disjoint spanning trees and depth-first search", *Acta Informatica* , Vol. 6, Springer, pp. 171–185.

Thierens, D. (2002), "Adaptive mutation rate control schemes in genetic algorithms", *Proceedings of the Congress on Evolutionary Computation* , Vol. 1, IEEE, pp. 980–985.

Van Dijk, S., Thierens, D. and van der Gaag, L. (2003), "Building a GA from design principles for learning bayesian networks", *Proceedings of the Conference on Genetic and Evolutionary Computation* , Springer, pp. 198–198.

Veltri, R., Chaudhari, M., Miller, M., Poole, E., O´ Dowd, G. and Partin, A. (2002), "Comparison of logistic regression and neural net modeling for prediction of prostate cancer pathologic stage", *Clinical Chemistry* , Vol. 48, American Association for Clinical Chemistry, pp. 1828–1834.

Weinberger, E. (1990), "Correlated and uncorrelated fitness landscapes and how to tell the difference", *Biological Cybernetics* , Vol. 63, Springer, pp. 325–336.

Wiggins, M., Saad, A., Litt, B. and Vachtsevanos, G. (2008), "Evolving a bayesian classifier for ecg-based age classification in medical applications", *Applied soft computing* , Vol. 8, Elsevier, pp. 599–608.

Wong, M., Lam, W. and Leung, K. (1999), "Using evolutionary programming and minimum description length principle for data mining of bayesian networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence* , Vol. 21, IEEE, pp. 174–178.

Wong, M. and Leung, K. (2004), "An efficient data mining method for learning bayesian networks using an evolutionary algorithm-based hybrid approach", *IEEE Transactions on Evolutionary Computation* , Vol. 8, IEEE, pp. 378–404.

Wu, Y., McCall, J. and Corne, D. (2010), "Two novel ant colony optimization approaches for bayesian network structure learning", *Proceedings of the IEEE Congress on Evolutionary Computation* , IEEE, pp. 4473–4479.

Wu, Y., McCall, J. and Corne, D. (2011), "Comparative analysis of search and score metaheuristics for bayesian network structure learning using node juxtaposition distributions", *Parallel Problem Solving from Nature XI* , Springer, pp. 424–433.

Wu, Y., McCall, J., Corne, D. and Regnier-Coudert, O. (2012), "Landscape analysis for hyperheuristic bayesian network structure learning on unseen problems", *Proceedings of the IEEE Congress on Evolutionary Computation* , IEEE, pp. 1–8.

Yang, Z. and Wright, R. (2006), "Privacy-preserving computation of Bayesian networks on vertically partitioned data", *IEEE Transactions on Knowledge and Data Engineering* , IEEE, pp. 1253–1264.

Zhang, Q. and Sun, J. (2006), "Iterated local search with guided mutation", *Proceedings of the IEEE Congress on Evolutionary Computation* , IEEE, pp. 924–929.