Problem Dependent Metaheuristic Performance in Bayesian Network Structure Learning

Yanghui Wu

A thesis submitted in partial fulfilment
of the requirements of
The Robert Gordon University
for the degree of Doctor of Philosophy

September 2012

Abstract

Bayesian network (BN) structure learning from data has been an active research area in the machine learning field in recent decades. Much of the research has considered BN structure learning as an optimization problem. However, the finding of optimal BN from data is NP-hard. This fact has driven the use of heuristic algorithms for solving this kind of problem.

A major recent focus in BN structure learning is on search and score algorithms. In these algorithms, a scoring function is introduced and a heuristic search algorithm is used to evaluate each network with respect to the training data. The optimal network is produced according to the best score evaluated. This thesis investigates a range of search and score algorithms to understand the relationship between technique performance and structure features of the problems.

The main contributions of this thesis include (a) Two novel Ant Colony Optimization based search and score algorithms for BN structure learning; (b) Node juxtaposition distribution for studying the relationship between the best node ordering and the optimal BN structure; (c) Fitness landscape analysis for investigating the different performances of both chain score function and the CH score function; (d) A classifier method is constructed by utilizing receiver operating characteristic curve with the results on fitness landscape analysis; and finally (e) a selective off-line hyperheuristic algorithm is built for unseen BN structure learning with search and score algorithms.

In this thesis, we also construct a new algorithm for producing BN benchmark structures and apply our novel approaches to a range of benchmark problems and real world problem.

Keywords: Bayesian Network, Search and Score Algorithm, Ant Colony Optimization, Node Juxtaposition Distribution, Fitness Landscape Analysis, Off-line Hyperheuristic, Receiver Operating Characteristic

Acknowledgements

I would like to give thanks to the many individuals who have helped me over the years in various ways.

First and foremost I want to extend my deepest gratitude to my principal supervisor, John McCall, for his guidance during my research, for his patience and support, for his expert advice and for his encouragement throughout all my PhD studies. Without John I wouldn't have reached the end of writing my thesis. I am also grateful to my secondary supervisor, David Corne, for his scientific advice and insightful guidance.

I would like to thank my darling wife, Shan and my daughter, Jiayu, for their love, understanding and support. I also want to give thanks to my family. For my father and mother, they provided the best environment for my brothers and myself to grow up with unconditional love and care. Although my father is no longer with us, he is forever remembered. I am sure he shares our joy and happiness in the heaven. For my brothers, their continuous encouragement and support helped make me concentrate on my research.

I am also grateful to many other people within RGU's school of computing: Olivier in particular, with whom I had many helpful discussions, provided me valuable suggestions in the project. Francois, for his evaluating the specific proposed approaches towards industry problems. Also to Noura, Clarie, Michael, Mehdi, Guofu, Peng, Lei, Ibrahim, Leszek, Nuka, Sadiq for the amazing time in CTC. Thanks also to Dawei, Andrei, Horacio and Qiang in the Computational Intelligence group for being so helpful and friendly. Thanks to the other staff in the department who kindly provided labs for my experiments and other help over the past three and half years. Thanks to Ratiba for her initial code to the project.

I would like to thank the Scottish Informatics and Computer Science Alliance (SICSA) and the Robert Gordon University (RGU) for providing the scholarship that allow me to work on my PhD. I am grateful also to Northwest A & F University, China

and my previous colleagues for encouragements and supports.

Published Papers

Some results and conclusions in this thesis have been published in the following publications:

- Yanghui Wu, John A. W. McCall, David W. Corne and Olivier Regnier-Coudert (2012). Landscape Analysis for Hyperheuristic Bayesian Network Structure Learning on Unseen Problems. IEEE Congress on Evolutionary Computation (*2012CEC*).

- Yanghui Wu, John A. W. McCall, David W. Corne (2011). Fitness Landscape Analysis of Bayesian Network Structure Learning. IEEE Congress on Evolutionary Computation(*2011CEC*). IEEE Press, pp.981-988, New Orleans, LA. 2011.

- Yanghui Wu, John A. W. McCall, David W. Corne. Two Novel Ant Colony Optimization Approaches for Bayesian Network Structure Learning. IEEE Congress on Evolutionary Computation (*2010CEC*). IEEE Press, Barcelona, Spain. 2010.

- Yanghui Wu, John A. W. McCall, David Corne. Comparative Analysis of Search and Score Metaheuristics for Bayesian Network Structure Learning Using Node Juxtaposition Distributions (*PPSN2010*). New York: Springer. pp. 424-433. Krakow, Poland.

- Fournier Francois., Yanghui Wu, John A. W. McCall, Andrei Petrovski, Peter J. Barclay. Application of evolutionary algorithms to learning evolved Bayesian Network models of rig operations in the Gulf of Mexico, 2010 UK Workshop on Computational Intelligence (*2010UKCI*), IEEE Press.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bayesian networks (BNs) are probabilistic graphical models which are used to represent knowledge about uncertain domains. The theoretical basis of BN is derived from the centuries-old Bayesian probability theory [Stigler, 1986]. The framework of a BN includes directed acyclic graphs (DAG), whose nodes represent random variables and whose edges represent conditional dependencies to the corresponding nodes [Heckerman, 1999] [Jensen, 1996]. From the late 1980s [Pearl, 1988] [Neapolitan, 1990], BN structure learning has been an attractive area of research in data mining. It has been proved that the BN structure learning from data is NP hard [Chickering, 1996]. With the problem complexity increasing, for large data size either in the number of instances, or the number of attributes, automatically constructing a BN structure presents a challenge for machine learning. In recent decades, learning BN structure from data has been often treated as an optimization problem. The purpose of this optimization is to seek a structure which can maximize a statistically defined score.

In this thesis, we first propose two novel approaches for BN structure learning. One of them is ChainACO algorithm, which is based on a chain model structure combining with Ant Colony Optimization (ACO) technique to search a best node ordering (with best Chain score fitness). This node ordering will give a reduced search space for K2 greedy algorithm to seek a BN structure. Another algorithm

is K2ACO, which utilizes ACO to evolve a structure that maximizes a statistically defined score, the CH score. We introduce standard benchmark problems for testing these two novel approaches and comparing with other two existing algorithms which are based on Genetic Algorithms.

We then conduct comparative analysis on these proposed algorithms using node juxtaposition distributions for studying relationships between the best node ordering and the optimal BN structure. We also introduce fitness landscape analysis for deeply investigating the performance of two scoring functions in this project. The results of fitness landscape analysis give us valuable inspiration for algorithm selecting on unseen problem structure learning. Finally, with the results from above research, we utilize receiver operating characteristic curve to construct a classifier method. This classifier method helps us to build a selective off-line hyperheuristic algorithm for unseen BN structure learning with search and score algorithm.

1.1 Thesis Outline

The whole thesis is divided into seven chapters.

Chapter 2 introduces the background of this research. It first reviews the BN problem, particularly the BN structure learning. It then goes on to review the existing BN structure learning algorithms, especially the search and score algorithms. It also includes a brief depiction of topological node ordering and probabilistic logic sampling for use in later chapters. Finally, it gives a summary of standard benchmark problems which are referred to throughout the thesis.

Chapter 3 proposes two approaches based on ACO with K2 greedy search algorithm, ChainACO and K2ACO. It indicates the details of each algorithm first and then describes the properties of the ACO metaheuristic in both ChainACO and K2ACO approaches. The chapter then looks at the experimental design and result analysis of the two new algorithms ChainGA and K2GA algorithms.

Chapter 4 This chapter investigates the performance of search and score algorithms using node juxtaposition distributions.

Chapter 5 This chapter takes advantage of fitness landscape analysis around the optimal solution to investigate the performance of two score functions, Chain score and CH score with benchmark problems.

Chapter 6 builds an off-line hyperheuristic algorithm for unseen BN structure learning with search and score algorithm.

Chapter 7 collects the original works in this thesis and draws conclusions and discusses future possible research directions which could proceed from these findings.

Chapter 2

Literature Review

Bayesian Network (BN) structure learning from data has become an attractive and active research area and has been attracting considerable interest over the last few decades. In this chapter, we will give a review of BN structure learning and explore the background to our research from some existing BN structure learning algorithms. This mainly includes scoring functions, search and score algorithms based on evolutionary computation techniques, Genetic Algorithm and Ant Colony Optimization. The chapter also includes the introduction of topological node ordering, hyperheuristic algorithms and probabilistic logical sampling which will be used for algorithm evaluation and experimental data generation. The chapter concludes with a review of eight standard benchmark structures which will be used in the experiments described throughout the subsequent chapters.

2.1 Bayesian Network

2.1.1 Terminology

In Table 2.1, we describe the symbols used in this thesis. One special thing to be noted is that we consider the terms ”node” and ”variable” ,”edge” and ”arc” interchangeably in the following chapters of the thesis.

Table 2.1: Symbols used throughout the thesis

symbol	symbol meaning
D	Data set studied
N	Number of points in data set *i.e.* $\|D\|$
$X, Y, Z, \ldots$	One dimensional random variables
$x, y, z, \ldots$	The corresponding values of variables $X, Y, Z, \ldots$
n	The size of node set
E	Set of edges of a BN
m	Number of edges of the BN *i.e.* $\|E\|$
(G, P)	Bayesian network, consisting of directed acyclic graph and parameters
P_{a_i}	Set of parents of variable X_i
$P_{a_{ij}}$	Set of values for value assignment j of each member of the set of parents P_{a_i} of X_i
r_i	Number of values of the discrete variables X_i
q_i	Number of configurations of values of the set of parents of X_i

2.1.2 Introduction to Bayesian Network

In probability theory, statistics and machine learning, a graphical model (GM) [Jordan, 1999] [Murphy, 2001] [Bishop, 2006] is a graph that represents dependencies among random variables. Probabilistic Graphical Models (PGMs) or probabilistic networks [Koller and Friedman, 2009] [Borgelt et al., 2009] is one kind of GM useful for reasoning with, or representing knowledge under uncertainty. Two general probabilistic networks are Markov networks and Bayesian networks (BNs). PGMs with undirected edges are generally called Markov networks. Markov networks are popular in fields such as statistical physics and computer vision [Edwards, 2000] [Koller and Friedman, 2009] [Korb and Nicholson, 2010]. PGMs with directed acyclic graph (DAG) are generally called BNs. In mathematics and computer science, a DAG is a directed graph with no directed cycles. Each node X_i in a DAG represents a variable and the arc represents dependency between these variables [Heckerman, 1999] [Jensen, 1996] [Daly and Shen, 2009] [Tsamardinos et al., 2006a].

For a given set of variables $X = \{X_1, X_2, \ldots, X_n\}$, a BN is composed of two aspects:

(1) a network structure G. Each variable of G has a finite set of states, and each

edge represents the direct dependency between two variables;

(2) a joint probability distribution (JPD) over the random variables $X_i (i = 1, 2, \ldots, n)$. According to the structure of a DAG in a BN, the JPD normally factorizes into a product of conditional probability distributions (CPD). In particular, given a structure G, the JPD for X is defined as equation (2.1)

$$P(X) = \prod_{i=1}^{n} P(X_i|P_{a_i}) \tag{2.1}$$

Here, P_{a_i} indicates the set of parents of node X_i, and $P(X_i|P_{a_i})$ shows the CPD of each node.

To deeply understand the above definition, a simple example of BN is introduced in Figure 2.1 [Murphy, 1998].

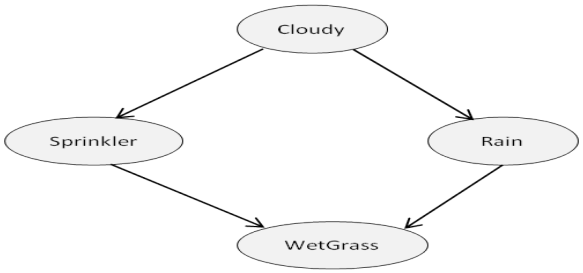


Figure 2.1: A simple Bayesian Network with Binary Variables

We suppose all variables in this structure are binary, i.e., each variable has two possible values denoted as T(true) and F(false). For a given BN problem, in addition to the structure graph, it is necessary to specify the parameters of this model. We use Conditional Probability Tables (CPT) to specify and understand the CPD of each variable. Take the node WetGrass (W) as an example. This variable has two parents, Sprinkler (S) and Rain (R), but one ancestor, Cloudy (C). From this relation, we can assume that the WetGrass is only dependent on Cloudy through Sprinkler or Rain. We see from equation (2.1) that the CPT of $P(C)$, $P(S|C)$, $P(R|C)$, $P(W|S, R)$ should be specified. The details of the values are shown in Table 2.2a– 2.2d.

A BN is a complete model for the variables and their relationships. It can be used to answer probabilistic queries about them. For example, we can use information of

Table 2.2: Conditional Probability Tables

(a) $P(C)$

$P(C = F)$	$P(C = T)$
0.5	0.5

(b) $P(S|C)$

C	$P(S = F)$	$P(S = T)$
F	0.5	0.5
T	0.9	0.1

(c) $P(R|C)$

C	$P(R = F)$	$P(R = T)$
F	0.8	0.2
T	0.2	0.8

(d) $P(W|S, R)$

S	R	$P(W = F)$	$P(W = T)$
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

some variables to obtain the probabilities for other variables. This is a most common task when using BN and is called probabilistic inference. This process can be expressed as given values (prior probabilities) for some variables and a set of query variables, to compute the posterior distribution over the query variables. In the example of Figure 2.1, suppose we have observed that the grass was wet. Which reason is more likely for it: raining or sprinkler? We can answer this by computing the posterior probability of each explanation through Bayes' rule as follows (Here, 1 represents T, and 0 represents F):

$$P(S = 1|W = 1) = P(S = 1, W = 1)/P(W = 1) = 0.430 \tag{2.2}$$

$$P(R = 1|W = 1) = P(R = 1, W = 1)/P(W = 1) = 0.708 \tag{2.3}$$

We can get these probabilities by using the chain rule of probability. For instance,

$$P(W = 1) = \sum_{c,r,s} P(C = c, S = s, R = r, W = 1) \tag{2.4}$$

From the values of posterior probability in each event, we can see that the most possible reason for the grass is wet is raining. The likelihood ratio for two reasons, raining and sprinkler is $0.708/0.430 = 1.647$.

BN structure has the inherent characteristics for representing the probabilistic dependencies between variables with a large degree of uncertainty. The ability of BNs to perform reasoning tasks using both graph and probability theories has spread its usage for various applications. BNs have been widely applied in a range of areas. These include statistics, machine learning, data mining and the artificial intelligence communities. BNs have became a useful tool for researchers. Some real world applications are: modeling knowledge in bioinformatics, computational biology [Friedman et al., 2000], marketing (Customer clustering, Market segmentation) [Battiato et al., 2010], Finance/Banking (Credit scoring, Fraud detection) [Neil et al., 2008], information retrieval [de Campos et al., 2004], Healthcare (Diagnostic expert systems, Microarray analysis) [Lucas et al., 2004] [Kabli et al., 2008] [Nikovski, 2000] [Rudini M. Sampaio and Silva., 2008] and weather forecasting [Abramson et al., 1996].

2.1.3 Bayesian Network Structure Learning

In order to fully know and comprehend the structure of BNs and thus fully represent the JPD of all variables contained in it, both the structure and the parameters of CPD associated with each variable X_i and its corresponding parents need to be learned from data. Normally, the process of estimating the unknown conditional distribution including parameters from data is called parameter learning. Some classical approaches for parameter learning are likelihood approach [Aldrich, 1997], expectation-maximization algorithm [Dempster et al., 1977], etc. The other process is called structure learning. The problem of learning BN structure can be stated as: given a training set of data $D = \{D_1, D_2, \cdots, D_n\}$, find a BN structure G of the DAG that best explains D [Margaritis, 2003].

Learning BN structure from data is NP-hard [Chickering, 1996] [Cooper, 1990]. It is known that the number of possible structures grows super-exponentially with the number of nodes, and so evaluating all possible structures is infeasible in most practical domains, where the number of variables is typically large. BN structure

learning is computationally expensive. Robinson [Robinson, 1973] has proved that the number of possible structures grows as $O(n!2^{\binom{n}{2}})$ for a problem with n variables. For example, the number of possible structures is 25 for problems involving 3 variables, 29,281 for 5 variables and 3,781,503 for 6 variables.

The research of finding appropriate methods for recovering BN structure from large datasets is an area that has gained importance in recent years and is now widespread. Traditional search algorithms have limitations both in space and time for BN structure learning. A large number of improved approximation approaches for structure learning have been introduced in recent years.

These algorithms are commonly classified into two main areas: methods based on conditional independence(CI) tests, and methods based on search and scoring metrics. The CI approaches perform a qualitative study to estimate whether certain conditional dependence exists between the variables in the domain. A network is then built to represent the observed dependencies and independencies as far as possible. These dependency relationships are usually measured through statistical or information theory tests. Measures such as Pearson's product moment correlation coefficient, the χ^2-test and mutual information (MI) are often used [Pearl, 2000] [Glymour and Scheinnes, 2000] [Campos and Huete, 2000] [Cheng et al., 1997]. The algorithm BNPC [Cheng et al., 1997] applied a qualitative analysis of mutual information between the variables in the studied field to build a structure G. Tests of conditional independence are equivalent to determine a threshold for mutual information between couples of involved variables. However, these algorithms have some shortcomings [Campos and Huete, 2000] [Cheng et al., 1997]. In the condition of an insufficient number of cases describing the observed domain, the statistical tests of independence are not reliable. The number of tests to be independently carried out to cover all the variable is huge. Since CI tests are computationally expensive and may be unreliable, people try to avoid CI tests with large condition-sets and use as few CI tests as possible.

The search and score approach is more popular, which takes the BN structure

learning as an optimization problem. It attempts to search for the best structure in the space of possible structures, in combination with a predefined score metric measuring the fit of each structure to the training data. Thus, metrics must be developed to score the candidate structures. The scoring metric introduced to evaluate a network G with respect to dataset D here, is a statistically motivated scoring function. This metric is described as [Cooper and Herskovits, 1992]:

$$Score_B(G : D) = P(G|D) \tag{2.5}$$

In a search and score algorithm, the search method attempts to get the maximum value of (2.5). By using Bayes' law, the above formulation can be expressed as:

$$Score_B(G : D) = P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{2.6}$$

In this formulation, the main part is the $P(D|G)$, for $P(D) > 0$ and $P(G)$ is the prior probability. To calculate $P(D|G)$, the Bayesian approach averages over all possible parameters, weighting each by their posterior probability:

$$P(D|G) = \int P(D|G, \mathbf{p})P(\mathbf{p}|G)d\mathbf{p} \tag{2.7}$$

then for multinomial local probability distributions, the Bayesian score will be in [Cooper and Herskovits, 1992]:

$$P(D|G) = P(G)\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(N'_{ij})}{\Gamma(N_{ij}+N'_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(N_{ijk}+N'_{ijk})}{N'_{ijk}} \tag{2.8}$$

here, $P(G)$ means the prior probability of the graph G. n is the number of discrete variables in the dataset. q_i denotes the number of possible different instances in the parents P_{ai} of the variable X_i can take. N_{ijk} denotes the number of cases in the dataset D in which X_i takes value k of its r_i possible values when its parents P_{ai} have their j-th configuration of values. N_{ij} is the sum of all N_{ijk} for all values X_i can take. $N'_{ijk} > 0(k = 1, 2, \ldots, r_i)$ are the hyper-parameters of the Dirichlet distribution

[Larrañaga et al., 1996]. N'_{ij} is the sum of all N'_{ijk} for all values X_i can take. $\Gamma(.)$ is the *Gamma* function, $\Gamma(c) = \int_0^\infty e^{-u}u^{c-1}du$. It should be noted that if c is an integer, $\Gamma(c) = (c-1)!$.

The most frequently search approaches applied in search and score algorithms are local search methods [Buntine, 1991] [de Campos et al., 2003] [Heckerman et al., 1995] [Cooper and Herskovits, 1992]. However, with the increasing of data size and search space, other more powerful heuristic search techniques have been introduced, such as Genetic Algorithms (GAs) [Larrañaga et al., 1994] [Wong et al., 1999], Simulated Annealing [Wang et al., 2004], Ant Colony Optimization (ACO) [Daly and Shen, 2009] [Pinto et al., 2009], Markov chain Monte Carlo [Castelo and Boutilier, 2001], PSO [Shibata et al., 2011] [Du et al., 2005], and Estimation of Distribution Algorithms (EDAs) [Blanco et al., 2003] [Romero et al., 2004].

Most of these algorithms employ the search space of DAGs. A range of other alternative search spaces are usually used in algorithms on BN structure learning. These spaces mainly focused on: the space of node orderings [Larrañaga et al., 1996] [de Campos and Puerta, 2001] [Kabli et al., 2007], the space of BN equivalence classes [Daly and Shen, 2009] and the space of possible edge incidence matrices [Pinto et al., 2008a] [Pinto et al., 2009].

2.1.3.1 Scoring Functions on Bayesian Network Structure Learning

The general idea of a Bayesian scoring function is to evaluate the posterior probability, through the known knowledge of prior probability distribution to the possible networks G, conditioned to data D, that is, $P(G|D)$. For all possible structures, the term $P(D)$ here has the same value, so for comparative purpose, we only need to evaluate the value of $P(G, D)$. Moreover, as the probability value of $P(G, D)$ is less than 1, it is easier to compute in the format of logarithmic space, so in real application, the scoring function $P(G, D)$ is always replaced by the value $log(P(G, D))$. The most common scoring metrics defined in search and score algorithms include the BDeu [Buntine,

1991], BIC [Schwarz, 1978], AIC [Akaike, 1970], CH metric [Cooper and Herskovits, 1992] and Minimum Description Length (MDL) [Yang and Chang, 2002] [Heckerman et al., 1995]. Here, we introduce some of them:

Cooper and Herskovits [Cooper and Herskovits, 1992] introduced the first used scoring function the CH score. It is a particular case of Bayesian Dirichle (BD) score. BD score is derived from some assumption of the network parameters distribution [Heckerman et al., 1995], such as multinomiality, lack of missing values, parameter independence, parameter modularity, and uniformity of the prior distribution of the parameters given the structures.

$$BD(G,D) = \log(P(G)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}(\log(\frac{\Gamma(N'_{ij})}{\Gamma(N_{ij}+N'_{ij})}) + \sum_{k=1}^{r_i}\log(\frac{\Gamma(N_{ijk}+N'_{ijk})}{\Gamma(N'_{ijk})})) \tag{2.9}$$

In (2.9), once the BD score with the uninformative assignment $N'_{ijk} = 1$, it is defined as CH score function:

$$g_{CH}(G,D) = \log(P(G)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}[\log(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}) + \sum_{k=1}^{r_i}\log(N_{ijk}!)] \tag{2.10}$$

here, r_i is the number of possible values assignments X_i has. In both (2.9) and (2.10), the term $\log(P(G))$ is quite common to assume a uniform distribution so that it becomes a constant and can be removed in real applications.

BDe is another variant of BD score. On an assumption of equivalency of likelihood is presented, this score has the advantage of preventing a particular configuration of a variable X_i and of its parents P_{ai} from being regarded as impossible. Buntine [Buntine, 1991] proposed a particular case of BDe score, called the BDeu score,

$$BDeu(G,D) = \log(P(G)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}(\log(\frac{\Gamma(\frac{N'}{q_i})}{\Gamma(N_{ij}+\frac{N'}{q_i})}) + \sum_{k=1}^{r_i}\log(\frac{\Gamma(N_{ijk}+\frac{N'}{r_iq_i})}{\Gamma(\frac{N'}{r_iq_i}}))) \tag{2.11}$$

which appears when

$$P(X_i = x_{ik}, \Pi_{X_i} = \omega_{ij}|G) = \frac{1}{r_iq_i},$$

This score only depends on one parameter, the equivalent sample size $N^{'}$.

BIC, AIC and MDL scores belong to information-theoretic scoring functions. They are based on compression:

- The score of a BN G is related to the compression that can be achieved over the data D with an optimal code induced by G.
- Shannon's source coding theorem establishes the limits to possible data compression.

In this compression, the score of BIC is defined as

$$BIC(G|D) = \log(P(G)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log(\frac{N_{ijk}}{N_{ij}}) - \frac{1}{2}|B|\log N \tag{2.12}$$

Here, $|B| = \sum q_i(r_i - 1)$ denotes the network complexity. It takes into account the complexity of the model and the complexity of encoding data related to this model.

2.1.3.2 K2 Algorithm for Structure Learning

K2 greedy search algorithm, proposed by Cooper and Herskovits [Cooper and Herskovits, 1992], is one of the most widely-studied algorithms for recovering BN structures from data. K2 algorithm starts by assuming that all nodes are without parents (i.e. independent), after which in every step it adds nodes incrementally to the parent set P_{a_i} of each node $X_i (i = 1, 2, \ldots, n)$. Assuming a uniform prior for $P(G)$, the following function score (2.13) is used to measure the conditional probability of each node and its parents. With new nodes added to the set of parent P_{a_i}, it should maximize the function (2.13) which is the main part in CH score function .

$$g(X_i, P_{a_i}) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2.13}$$

The process stops adding the nodes to the set of parents when the addition of a single parent cannot increase the probability $g(X_i, P_{a_i})$. A pseudo code representation of K2 algorithm is shown in Algorithm 1.

Algorithm 1 K2 structure learning algorithm

Input: A set of n nodes, an ordering on these nodes, an upper u on the number of parents a node may have.
Output: for each node, a printout of the parents of the node.
Begin K2
for $i \leftarrow 1, n$ **do**
  $P_{a_i} = \phi$
  $P_{old} = g(i, P_{a_i})$
  ok_to_proceed=true
  while (ok_to_proceed) and ($|P_{a_i}| < max\_parents$) **do**
    let z be the node in $Pred(x_i) - P_{a_i}$ that maximizes $g(i, P_{a_i} \cup z)$;
    $P_{new} = g(i, P_{a_i} \cup z)$
    if ($P_{new} > P_{old}$) **then**
      $P_{old} = P_{new}$;
      $P_{a_i} = P_{a_i} \cup z$;
    else
      ok_to_proceed=false;
    end if
  end while
  Write Node: x_i, parent: P_{a_i}
end for
End K2

In K2 algorithm, the search space is composed of node orderings. This contributes to the reducing of computational complexity. The node ordering imported for K2 algorithm greatly affects the performance of K2 algorithm. The proper ordering will generate the correct result, the improper order will result in a poor result. The ordering required in K2 algorithm is such that if the node X_i comes in advance to the node X_j in the ordering, then the node X_j can not be as a parent to the node X_i.

The algorithm stops when any of the following conditions are met:

- The maximum number of parents for that particular node has been reached.
- There are no more legal parents to add.
- No additional parent improves the score.

2.2 Genetic Algorithm for Bayesian Network Structure Learning

2.2.1 Genetic Algorithm

In the field of Evolutionary Algorithms, GA is one of the most widely applied heuristic algorithms which was first introduced by Holland in 1975 [Holland, 1975]. GA is used to generate optimal solution to optimization problems through taking some inspirations from natural evolution. These techniques include selection, mutation and crossover.

In a GA, the search space is composed of a certain number of individuals, which is represented by character strings. The purpose of a GA is to find the best individual from search space to get the best objective function. The score function defined in GA is to assess the quality of the possible solution.

The evolution of a GA usually works as follows [Whitley, 1994] [Goldberg, 2002]:

Initialization: It starts from an initial population constructed with randomly generated individual solutions. The quality of this population is determined by all of these individuals. The size of the initial population depends on the property of the

problem, through hundreds to thousands of the potential solutions to cover the entire range of the problem.

Selection: A certain percentage of the individuals are selected from the search space to produce a new search space for later evolving. The individuals are chosen according to a certain kind of select method, for example the fitness-based process, stochastic universal sampling, tournament selection, etc.

Reproduction: Crossover and Mutation operations are applied to the chosen individual solutions to form a new solution population. Both crossover and mutation processes can help to expand the search space, avoid producing local optimal and improve the average quality of the population.

The GA iterates the selection and reproduction processes. The termination condition of GA is either the generation number reaching the maximum number or when a satisfactory score value has been achieved.

Algorithm 2 describes the pseudocode of the GA.

Algorithm 2 The pseudocode of Genetic Algorithm

```
1: Begin GA
2: Make initial population at random
3: while (termination conditions not met) do
4:    select parents from the population.
5:    produce children from the selected parents.
6:    Mutate the individuals.
7:    Extend the population by adding the children to it.
8:    Reduce the extended population.
9: end while
10: Output: the best individual found.
11: End GA
```

GA has been utilised in solving real world problems in many fields with success, such as bioinformatics [Imade et al., 2004], image processing [Naik and Murthy, 2006], data mining [Minaei-Bidgoli and Punch, 2003] and real time systems [Wegener et al., 1997].

2.2.2 Genetic Algorithm for Bayesian Network Structure Learning

We have known that K2 algorithm has been applied over the space of node orderings for BN structure learning problem. However, as a greedy search algorithm, it has the disadvantage of converging to a local optimum. The advantages of GA make it a feasible choice for solving the optimization problem in BN structure learning. These advantages are mainly: (1) the initial individuals in the search space represent several possible solutions. With the selection and iteration, it contributes to reduce the risk of convergence on a local optimum. (2) The mutation operator can explore the area of solutions, enlarge the robustness of GA to a single local optimum. For these reasons, in recent decades, GA has already been successfully used in various forms on BN structure learning problem. Some research as [Larrañaga et al., 1994] [Larrañaga et al., 1996] [Wong et al., 1999] [Muruzabal and Cotta, 2004] [Acid and de Campos, 2003] [Brouard et al., 2008]. Among these works, we are mainly interested in the research [Larrañaga et al., 1996], for in this thesis, we will use it as comparison to our results.

In [Larrañaga et al., 1996], the authors proposed to search for the best ordering in the space of node orderings using a GA. In this approach, the K2 algorithm is used to retrieve the BN structure on each node ordering. In this thesis, we name it as K2GA algorithm. The pseudocode of K2GA algorithm is shown in Algorithm 3.

2.3 Ant Colony Optimization for Bayesian Network Structure Learning

2.3.1 Ant Colony Optimization

Ant Colony Optimization (ACO) [Dorigo and Stützle, 2002] [Gambardella et al., 2002] [Dorigo et al., 1996] is a relatively new paradigm of natural inspired algorithms. In the early 1990s, it was first proposed by M. Dorigo and his colleagues as a novel naturally inspired metaheuristic for solving combinatorial optimization problem [Dorigo and

Algorithm 3 Pseudocode of K2GA Algorithm

1: **Begin K2GA**
2: Make initial population $P(t)$ at random, $t = 0$;
3: Train model by K2 algorithm and evaluate BN score $S(t)$ with CH score;
4: **while** (termination conditions not met) **do**
5: $t = t + 1$;
6: Select a sub-population for offspring production $P' = selectparentsP(t)$;
7: Recombine the individuals of selected parents recombine $P'(t)$;
8: stochastically Mutate the individuals $P'(t)$;
9: Evaluate its new fitness evaluate $P'(t)$;
10: Select the survivors from actual fitness $P = \{surviveP, P'(t)\}$;
11: **end while**
12: **Output:** the best individual found, the BN score and structure.
13: **End K2GA**

Blum, 2005]. So far, there are some different kinds of ACO algorithms, such as Ant System (AS) [Dorigo et al., 1996], Ant Colony System (ACS) [Dorigo and Gambardella, 1997] and *MAX-MIN* Ant System (*MMAS*) [Stützle and Hoos, 2000].

ACO is inspired by cooperative behaviour in real ant colonies which have been observed to find shortest paths when seeking food. When ACO is used for optimization, artificial ants walk a random path on a graph, depositing pheromone. Following ants use pheromone levels on edges of the graph to bias their walk. In the absence of pheromone, ants choose randomly. The more ants choose a path, the more pheromone is deposited on it. After a number of iterations, paths with high pheromone will be selected with a high probability [Dorigo and Caro, 1999].

2.3.1.1 The Framework of Ant Colony Optimization

Algorithm 4 is a description of a basic framework of ACO [Dorigo and Gambardella, 1997].

We explain each of the component as follows:

Ant Based Solution Construction: Any ant constructing a solution starts from the empty solution, then moves through the nodes of the construction graph. At

Algorithm 4 Ant Colony Optimization metaheuristic

```
1: while termination conditions not met do
2:    ScheduleActivities
3:        Ant Based Solution Construction()
4:        Pheromone Update()
5:    end ScheduleActivities
6: end while
```

each step, the ant selects the following node in the construction graph by using the stochastic local decision rule, which is composed of the heuristic information and the pheromone values. One of the possible stochastic local decision rules is Ant System. The transition probability is (2.14)

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in u}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}} \tag{2.14}$$

where τ_{ij} is the pheromone value, η_{ij} represents heuristic information about the problem. α and β are two positive parameters, whose values determine the relative influence of the pheromone and the heuristic, u denotes the set of candidate solutions to be chosen.

Pheromone Update:

As the ants are walking, in good or favorable solutions, the values of the pheromone are increased. On the contrary, the pheromone values on the bad ones will be decreased. These processes are pheromone updating. This updating process includes: (1) increasing pheromone level to a certain selected good path S_{good} by a value $\Delta\tau$; (2) decreasing pheromone level to all the path solutions. This is a pheromone evaporation process. The aim of this process is to avoid a fast convergence of the algorithm to a local optimum. A typical pheromone updating rule is expressed in (2.15),

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau(i,j) & \text{if } \tau_{ij} \in S_{good}, \\ (1-\rho)\tau_{ij} & \text{otherwise.} \end{cases} \tag{2.15}$$

where $\rho \in (0, 1]$ is the evaporation rate. Global pheromone updating and local pheromone updating are two main pheromone updating rules. Once an ant has completed its own construction step, a certain amount of pheromone is deposited to its corresponding path. This is a local updating pheromone rule. Once all ants have finished constructing their tours, a certain percentage of pheromone evaporates to all paths, and then the fraction of pheromone deposit to the path belongs to the best tour. This process is the global pheromone updating [Dorigo et al., 1996] [Dorigo and Gambardella, 1997]. One definition of the local pheromone updating rule is described as (2.16)

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau(i, j) \tag{2.16}$$

where $\rho \in (0, 1]$ is the evaporation rate. The value of $\Delta\tau$ is related to the specific problem studied or by other methods.

2.3.1.2 Ant Colony Optimization Application

ACO algorithm has shown very good performance when applied to solving combinatorial problems such as the traveling salesman problem (TSP) [Dorigo and Gambardella, 1997] [Bianchi et al., 2002], which is one of the first problems studied. Recently, researchers have expanded the application of ACO algorithm to the fields of data mining [Parpinelli et al., 2002], multi-objective problems [Guntsch and Middendorf, 2003], Bioinformatics problems [Shmygelska et al., 2002] and dynamic or stochastic problems [Guntsch and Middendorf, 2001]. Many applications on other fields have been proposed including the quadratic assignment problem [Tsutsui, 2008], routing problems [Cicirello and Smith, 2001], sorting problems [Hartmann and Runkler, 2007] and industrial problems [Dorigo et al., 2002].

2.3.2 Ant Colony Optimization for Bayesian Network Structure Learning from Data

It is a natural choice for applying ACO to the problem of determining BN structures. Some researchers have used ACO on solving BN structure learning. These approaches include ACO-E [Daly and Shen, 2009], MMACO [Pinto et al., 2008b] [Pinto et al., 2009], ACO-B [de Campos and Puerta, 2002] and ACO-K2SN [de Campos et al., 2002]. These approaches typically integrate with other greedy construction heuristic algorithms, such as Algorithm B [Buntine, 1991], K2SN algorithm [Campos et al., 2001] and the local discovery algorithm MMPC [Tsamardinos et al., 2006a]. The experimental results with some benchmark problems have shown that ACO based algorithms can achieve better results than other techniques, such as GAs, Max-Min Hill Climbing (MMHC), EDAs, etc.

The ACO based algorithms, which have been developed so far for BN structure learning, have focused on the program of representations to the problem, the definition of heuristic information, pheromone updating rules and the probabilistic transition rule to help ants move from one stage to the next stage. We now compare these existing ACO based algorithms, in terms of search space, heuristic information, score metrics and pheromone initialization.

2.3.2.1 ACO-E

ACO-E [Daly and Shen, 2009]

Search Space: The space in ACO-E algorithm is composed of equivalence classes. Two DAGs are equivalent if and only if they have the same skeletons and the same set of v-structures [Verma and Pearl, 1990]. The skeleton is the undirected graph that results in un-directing all edges in a DAG and a v-structure is the head-to-head meeting of two arcs.

Heuristic Information: the heuristic information defined in ACO-E is the similar

to the definition in the TSP problem. It is

$$\eta_{ij} = \frac{1}{|f(X_j, P_{aj})|} \tag{2.17}$$

in ACO-E, the heuristic information is dynamic. It depends on the current state of the ant. The value of the heuristic information is given by the score gained for each edges correctly inserted, or inserted in reverse.

Pheromone Update: Two ways for pheromone updating: Local evaporation rule, whereby pheromone is removed from a path as an ant traverses it.

$$\tau_m = (1 - \rho)\tau_m + \rho\tau_0 \tag{2.18}$$

Global pheromone update rule that deposits new pheromone on the best-so-far-path

$$\tau_m = (1 - \rho)\tau_m + \frac{\rho}{|Score(P^+)|} \tag{2.19}$$

here, P^+ is the best-so-far solution. *Score* is the score fitness for the BN structure.

Probabilistic Transition Rule: the rule normally uses the values given by the heuristic information and pheromone to inform the choice of which node to pick. An ant chooses component c_m, where m is given by

$$\tau_{ij} \leftarrow \begin{cases} argmax_{m \in N(x)} \tau_m [\eta_m]^\beta, & \text{if } q < q_0, \\ \text{random proportional}, & \text{otherwise.} \end{cases} \tag{2.20}$$

here, $N(x)$ is the set of components that an ant at state x can move to, given the problem constraints Ω. τ_m and η_m are the pheromone and heuristic information, β is a parameter that says how much to favor the heuristic over the pheromone. If the number q is greater than q_0, then a random proportional rule is used to select which component to visit next. This rule is

$$p_m = \frac{\tau_m [\eta_m]^\beta}{\sum_{u \in N(x)} \tau_u [\eta_u]^\beta} \tag{2.21}$$

It can be seen that the probability that an ant moves to component c_m is directly given by $\tau_m[\eta_m]^\beta$, normalized over the other possible moves so that it is in the range $[0, 1]$.

Initial pheromone: the initial pheromone for the local pheromone rule is defined as

$$\tau_0 = \frac{1}{n|Score(P^+)|} \tag{2.22}$$

n is the number of variables that are in the data.

2.3.2.2 ACO-B

ACO-B [de Campos and Puerta, 2002] is built through arranging the behavior of ants on a different BN structure learning algorithm, as in Algorithm B [Buntine, 1991].

Search Space: the algorithm conducts the search in the space of DAGs.

Heuristic Information: the heuristic component is defined as

$$\eta_{ij} = f(x_i, P_{ai} \cup x_j) - f(x_i, P_{ai}) \tag{2.23}$$

here f is the scoring metric being used, and P_{aj} , the parents of X_j are found by K2 algorithm.

Pheromone Update: it uses the classical local and global pheromone updating rules, as expressed in (2.15) and (2.16).

Probabilistic Transition Rule: the next node to be visited in the sequence is selected in the same way as expressed in (2.14).

Initial pheromone: in ACO-B, the initial amount of pheromone is calculated as

$$\tau_0 = \frac{1}{n|f(S_{K2SN})|} \tag{2.24}$$

S_{K2SN} is the solution obtained by the algorithm K2SN, and $f(S_{K2SN})$ is the score of this solution.

2.3.2.3 ACO-K2SN

ACO-K2SN [de Campos et al., 2002] uses local changes in the space of DAGs. These changes include arc addition, arc deletion and arc reverse. It uses CH metric as the scoring function to evaluate the search performance.

Search Space: the space of ordering of the DAGs is searched in ACO-K2SN algorithm.

Heuristic Information: the heuristic information is defined as

$$\eta_{ij} = \frac{1}{|f(X_j, P_{aj})|} \tag{2.25}$$

Pheromone Update: it uses the classical local and global pheromone updating rules, as expressed in (2.15) and (2.16). In these equations, the update value for the pheromone is given by

$$\Delta_{ij} = \frac{1}{|f(S^+)|} \tag{2.26}$$

here S^+ is the best-so-far structure.

Probabilistic Transition Rule: the next node to be visited in the sequence is selected in the same way as expressed in (2.14).

Initial pheromone: in ACO-K2SN, the initial amount of pheromone is calculated as

$$\tau_0 = \frac{1}{n|f(S_{K2SN})|} \tag{2.27}$$

S_{K2SN} is the solution obtained by the algorithm K2SN, and $f(S_{K2SN})$ is the score of this solution.

2.3.2.4 MMACO

This algorithm is built on the local discovery algorithm MAX-Min Parents and Children (MMPC) and ACO. MMPC is used to construct the skeleton of the BN, and ACO

is used to orientate its edges. With these processes the final structure is built. The main description of this algorithm is as below [Pinto et al., 2008b] [Pinto et al., 2009]:

Search Space: the search space is the set of possible edges in this algorithm.

Heuristic Information: in MMACO algorithm, the heuristic information is defined as

$$\eta_{ij} = 1 + f(s_k \cup < E_{ij}, e >) - f(s_k \cup < E_{ji}, e >) \tag{2.28}$$

here f is the scoring metric found by BDeu score function, s_k is the partial solution of the problem, $s_k \cup < E_{ij}, e >$ is the partial solution of the problem after assigning the value e to edge E_{ij}.

Pheromone Update: two ways of pheromone updating during the optimization process. First, the pheromone matrix is updated step-by-step after each ant chooses an edge to visit,

$$\tau_{ij} \leftarrow \tau_{ij}(1 - \rho_{sbs}) \tag{2.29}$$

here, ρ_{sbs} is the evaporation coefficient of the step-by-step pheromone update. Second, the global pheromone is conducted at the end of iteration N_{iter}, by the best solution found so far. The rule is

$$\tau_{ij}(N+1) = \tau_{ij}(N)(1-\rho) + {\delta_{ij}}^k \tag{2.30}$$

here, ${\delta_{ij}}^k$ is defined as ${\delta_{ij}}^k = \frac{1}{|f(G)|}$. f is the BDEu score.

Probabilistic Transition Rule: the probability of edge E_{ij} being assigned the value e is given by

$$p_e = \frac{{\tau_e}^\alpha {\eta_e}^\beta}{{\tau_e}^\alpha {\eta_e}^\beta + {\tau_{\bar{e}}}^\alpha {\eta_{\bar{e}}}^\beta} \tag{2.31}$$

here, β and α are two parameters which control the importance of heuristic information and pheromone respectively. e is 0 or 1, and η_e is 0 or 1 with the quality of the solution

increasing or decreasing respectively.

Initial pheromone: in MMACO, the initial amount of pheromone is calculated as

$$\delta_{ij}{}^{k} = \frac{1}{|f(G)|} \tag{2.32}$$

here, f is the BDEu score.

2.4 Topological Node Ordering

In computer science and data mining, a topological node ordering of a graph $G = (V, E)$ is defined as a linear ordering of its variables as $X_1, X_2, \cdots, X_n$. For every edge from node i to node j, i comes before j in the ordering, $i < j$ [Thomas H. Cormen, 2001] [Arge et al., 2003]. In [Vernet and Markenzon, 1997] it is proved that:

- If a graph G is a DAG, a topological node ordering can be constructed, and the number of topological orderings is at least one.
- If G is a DAG, then G has a node with no incoming edges.
- If a DAG has two or more topological node orderings, in the case where there exist two consecutive nodes that are not connected by an edge it is always possible to form a second topological ordering by swapping these consecutive nodes.

An example of a DAG and any two of the corresponding topological orderings are present in Figure 2.2.

Breadth first search (BFS) and depth first search (DFS) are two typical sorting methods for a DAG. The BFS algorithm starts from the root variable and then looks through all the neighbor variables. For each of those variables which are the nearest ones to the selected variable, BFS looks through its unexplored neighbor variables till it achieves the final goal [Diekert and Gastin, 1995]. One of the best-known BFS algorithms is the Kahn algorithm, described by Kahn [Kahn, 1962]. It works by

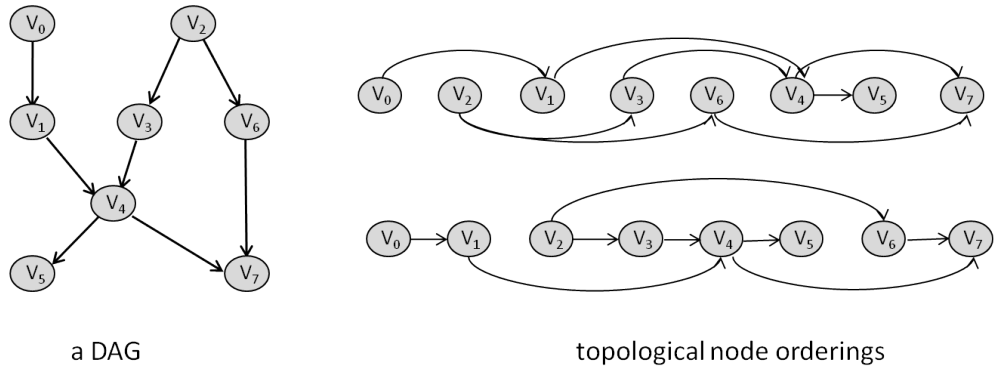


Figure 2.2: DAG and topological sorts

selecting variables in the same level, after each level, it then continues to select from the next level till the last one to construct a topological ordering. In this thesis, we call the ordering made by Kahn algorithm as Kahn ordering. The details of how to construct a Kahn is described in Algorithm 5.

Algorithm 5 Pseudo code of Kahn algorithm

```
1: L ← Empty list that will contain the sorted elements.
2: S ← Set of all nodes with no incoming edges.
3: while (S is non-empty) do
4:    remove a node X_i from S
5:    for (each node X_j with an edge E from X_i to X_j) do
6:       remove edge E from the graph
7:       if (X_i has no other incoming edges) then
8:          insert X_i into S
9:       end if
10:   end for
11: end while
12: Output: message (proposed to topologically sorted order: L)
```

One of the best-known DFS algorithms is the Tarjan algorithm, described by Tarjan [Tarjan, 1976]. The algorithm starts with variables in the first level of the graph, then through unexplored variables, which is its ancestor, or a descent connected with edges. The DFS stops when it hits any variable that has already been chosen. We

name the topological ordering built by DFS algorithm as DFS ordering. A pseudo code representation of Tarjan algorithm is described in Algorithm 6.

Algorithm 6 Pseudo code of Tarjan algorithm

```
1: L ← Empty list tat will contain the sorted nodes.
2: S ← Set of all nodes with no incoming edges.
3: for (each node X_j in S) do
4:    visit (X_i)
5:    FUNCTION visit(node (X_i))
6:    if (X_i has not been visited yet) then
7:       mark (X_i) as visited
8:       for (each node X_j with an edge E from X_i to X_j) do
9:          visit (X_i)
10:         add (X_i) to L
11:      end for
12:   end if
13: end for
14: Output: message (proposed to topologically sorted order: L)
```

Figure 2.3 represents an example of two topological node orderings, comparing with the true edges on the BN structure. This structure has 20 true edges. The Kahn sort is 0-1-3-5-4-2-6-7-8-10-11-13-12-9, and the Tarjan sort is 0-6-2-1-4-5-7-9-12-13-3-8-11-10. We can find that different topological sorts has different node ordering. However, they have the same number of arcs. This appearance makes it very complex to investigate the characteristic of topological orderings. However, we know, through both Kahn and Tarjan node ordering, we can get the best BN structure through K2 algorithm, for they all are the best node ordering to K2 greedy algorithm.

2.5 Probabilistic Logic Sampling

Probabilistic Logic Sampling is the simplest and the first proposed sampling algorithm for BNs [Henrion, 1988]. In this algorithm, the nodes are sampled in the order which are consistent with the partial orders of the structure, so that by the time a node is

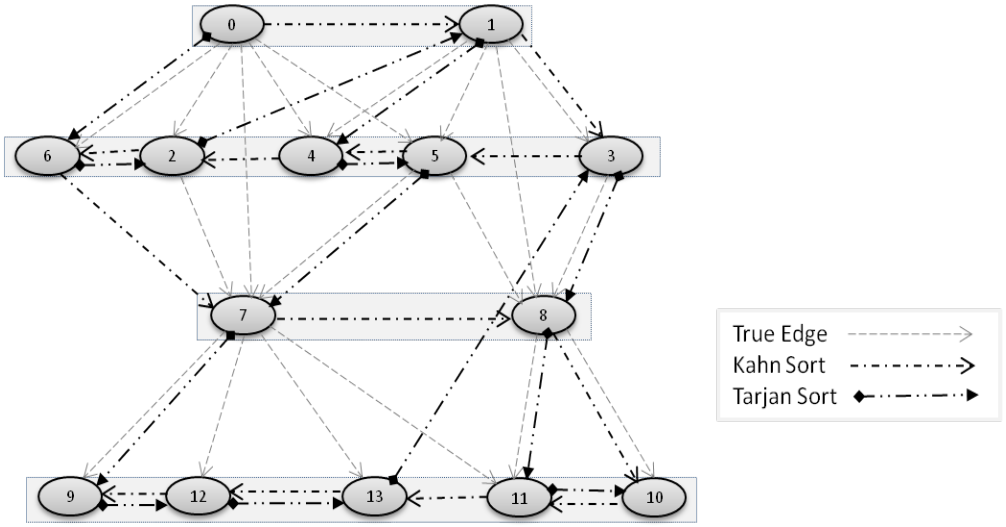


Figure 2.3: DAG and topological sorts

sampled the values for all of its parents are also known. We can then sample from the distribution defined by the CPD and by the chosen values for the node's parents. Note that the algorithm requires that we have the ability to sample from the distributions underlying our CPD. Such sampling is straightforward in the discrete case [Koller and Friedman, 2009]. Algorithm 7 is the pseudo code of a probability logic sampling algorithm.

Algorithm 7 Pseudo code of Probability Logic Sampling

```
1: Order the nodes X_1, X_2, ··· , X_n according to their topological order.
2: for i ← 1, Nuber_of_Case do
3:    for j ← 1, n do
4:       if (root nodes) then
5:          select state with probability P(X = X_j);
6:       else
7:          select state with probability P(X = X_j|P_aj);
8:       end if
9:    end for
10:   return x_1, x_2, ··· , x_n;
11: end for
```

For a given structure in Figure 2.1, the topological node ordering of four nodes

are C-S-R-W for Kahn algorithm, and C-S-W-R for Tarjan algorithm. In each ordering, the root node is C, so select C=F with probability based on value of probability 0.5. Select value of S with probability based on value of the C, if C=F select S=F with probability 0.5 and S=T with probability 0.5. However, for the third node, select the value of R based on S in Kahn ordering, and select the value of W based on S in Tarjan node ordering. The process continues similarly for the last node W or R. In spite of two different topological node orderings, the data sampled from them have the same effect in BN structure learning.

2.6 Hyperheuristics

Metaheuristic algorithms are very widely used to optimize problems in operation research, for the greatest quantity of these problems are NP hard and exhaustive search in large search space. It is hard to develop problem specific heuristic algorithms for these kind of computational problems. In recent years, a novel concept, hyperheuristic has been provided in the area of search and optimization problems which helps to overcome the problem of such drawbacks in metaheuristics [Özcan et al., 2008] [Bilgin et al., 2006] [Kendall, 2001]. A hyperheuristic algorithm is constructed through a specific heuristic search algorithm integrated with some data mining methods. These data mining methods are applied as selecting, generating and adapting to the simple heuristic algorithm in order to produce an efficient algorithm for solving the complex optimization problem.

The search space for a hyperheuristic is the space of heuristics. This differs to the heuristic algorithm, whose search space is all possible solutions of the problem. Therefore, the objective of using a hyperheuristic is not to solve the problem itself, but to seek an appropriate heuristic algorithm to the problem. Moreover, the motivation for investigating hyperheuristics is to search for a generally applicable methodology rather than solving a single problem instance.

Hyperheuristic algorithms can be grouped into two categories according to whether

they are based on constructive or perturbative search [Bader-El-Den and Poli, 2008]. The first one is on-line learning hyperheuristics. In this case, the problem couldn't provide any prior knowledge to the learning algorithm. The learning process usually occurs through gaining properties of problem dependent as well as the problem instance they are working on purely by trial and error. A typical on-line Hyperheuristic algorithm is using the reinforcement learning skill for heuristic selection and according to metaheuristics approach as a search strategy on a search space of heuristics algorithms [Kaelbling et al., 1996].

The second type is off-line learning hyperheuristics. This kind of process needs a set of training problems provided before it is actually used. The hyperheuristic then tries to gather knowledge through some methods from these problems. The knowledge is expressed as some kind of rules or programs. All this knowledge achieved will be applied to solve unseen problems outside the training instances. Some examples of off-line hyperheuristics learning approaches are: from rules or learning classifier [Marín-Blázquez and Schulenburg, 2007] [Ross and Hart, 2002], over case-base reasoning [Burke et al., 2006] and genetic programming [Ross and Marín-Blázquez, 2005] [Terashima-Marín et al., 2005].

2.7 Benchmark Structures

A number of test benchmark structures are commonly used to evaluate the performance of novel approaches in a BN structure learning problem. Normally, once the benchmark structure is selected, a database of the cases D is simulated from these benchmark structures for later experiments. The dataset D here implies the relationships of conditional independence within the variables. The proposed approaches are then applied to the database D to retrieve a structure G^*, which can achieve the maximum value of the probability $P(D|G)$. The fitness of the solutions G^* found is evaluated to measure the performance of the algorithm proposed. Figure 2.4 shows the process of benchmark and database in BN structure learning. These test prob-

lems are designed mostly from real-world problems and can be used to predict an algorithm's performance. We should clearly know that an algorithm which has good performance in relation to one problem can not be guaranteed to fit to another.

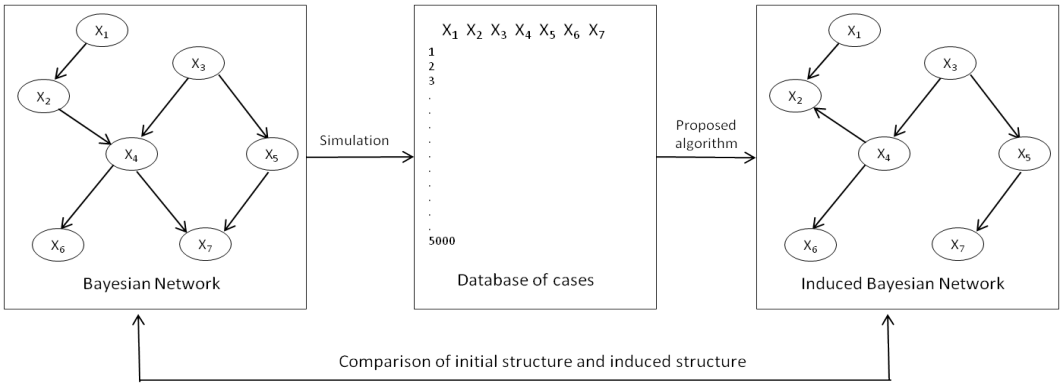


Figure 2.4: The process of proposed approach for structure learning from simulated database of cases

All benchmark structures described here are discrete problems without missing values. There are two reasons for selecting these benchmark structures. Firstly, all problems have been used as standard benchmarks for other proposed algorithms on BN structure learning. The results have proven these structures are useful for testing these algorithms. This is important as it allows comparison to be made between the novel algorithms and the existing algorithms. Secondly, the structures cover a wide range of fields, for example, medicine, industry, commerce, etc. When sampling data from benchmark structures, different sampling methods may be used, according to the demands of the problem. The number of cases in these problems are different and the size of each case is not fixed. This size can be changed according to the specific problem and the algorithm itself.

In the following chapters, we use GeNIe&SMILE [GeNIe and SMILE,] software as a tool to generate databases from selected benchmarks. GeNIe&SMILE (Structural Modeling, Inference, and Learning Engine) is developed at the Decision Systems Laboratory, University of Pittsburgh. In the following experiments, some data cas-

es are sampled using likelihood sampling, which is a kind of stochastic sampling algorithm [Fung and Chang, 1989]. The likelihood sampling algorithm makes an attempt at improving the efficiency of the probabilistic logic sampling algorithm by instantiating only non-evidence nodes [Shachter and Peot, 1990]. Other data cases are simulated through probabilistic logic sampling [Henrion, 1988] in the Genie tool software [GeNIe and SMILE,].

2.7.1 Asia Structure

The Asia [Lauritzen and Spiegelhalter, 1988] structure is a simple BN network but is popular used for BN structure learning, which is present in Figure 2.5.

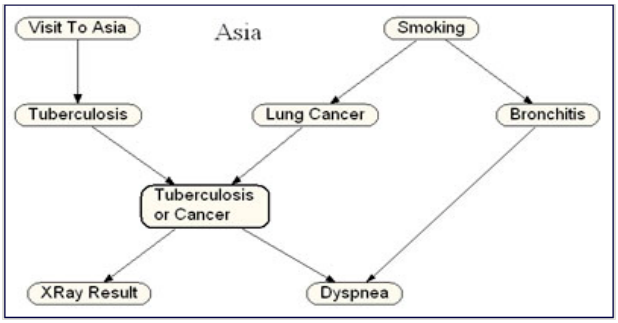


Figure 2.5: The structure of the Asia network

Asia structure is a simple BN model used in patient diagnosis. Each variable in the Asia network represents a special condition of the patient, for example, "Visit to Asia" indicates whether the patient recently visited Asia. An edge between any two nodes indicates a probabilistic relationship between the states of the corresponding variables. The detailed states of each node in this structure are listed in Table 2.3.

In our following experiments, for the sake of convenience, we number each variable as: $VisittoAsia = 1, tuberculosis = 2, Smoker = 3, Lungcancer = 4, Lungcanerortuberculosis = 5$, $PositiveX - ray = 6$, $Brochitis = 7$, $Dysponoca = 8$. In this situation, the database simulated from the structure is present as Figure 2.6

In the above database, both 0 and 1 indicate the states of each variable. For

Table 2.3: States of each node in the Asia structure

Node	state:01	state:02
Visit to Asia	visit	no visit
Tuberculosis	present	absent
Smoking	smoker	no smoker
Lung cancer	present	absent
Lung caner or tuberculosis	true	false
Positive X-ray	abnormal	normal
Brochitis	present	absent
Dysponoca	present	absent

No.	1	2	3	4	5	6	7	8
0001	0,	0,	0,	0,	0,	0,	0,	0
0002	0,	1,	0,	0,	1,	1,	0,	1
0003	0,	1,	0,	0,	1,	1,	0,	0
0004	0,	0,	0,	0,	0,	0,	0,	1
0005	0,	0,	1,	0,	0,	0,	0,	0
.								
.								
.								
4999	0,	0,	1,	0,	0,	0,	1,	1
5000	0,	0,	1,	0,	0,	0,	1,	0

Figure 2.6: The database of the Asia network simulated

example, 0 in variable "1"(= *VisitToAsia*) means *VisitToAsia* = *visit*, and 1 in variable "1"(= *VisitToAsia*) means *VisitToAsia* = *noVisit*.

To describe other benchmark structures in the following section, we will not use the original name but instead use a number in each structure. This kind of expression for the name will not effect the experimental results.

2.7.2 Credit Structure

Credit structure is another simple model for assessing the credit worthiness of an individual, developed by Gerardina Hernandez [GeNIe and SMILE,] as a class homework at the University of Pittsburgh. It has 12 nodes and 12 edges. All parentless nodes

are described by uniform distributions. This is a weakness of the model, although it is offset by the fact that all these nodes will usually be observed and the network will compute the probability distribution over credit worthiness correctly. The structure is present in Figure 2.7.

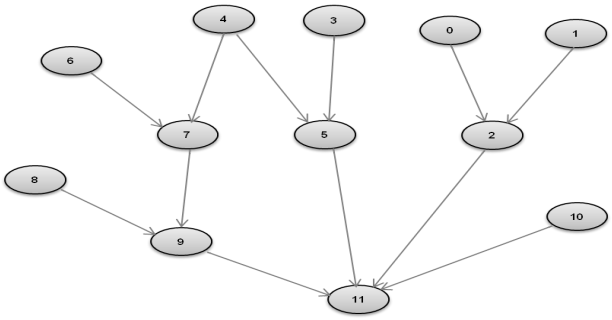


Figure 2.7: The structure of the Credit network

2.7.3 Tank Structure

Tank Structure is a network used for diagnosing possible explosions in a tank, developed by Gerardina Hernandez [GeNIe and SMILE,] as a class homework at the University of Pittsburgh. It has 14 nodes and 20 edges. The structure is present in Figure 2.8.

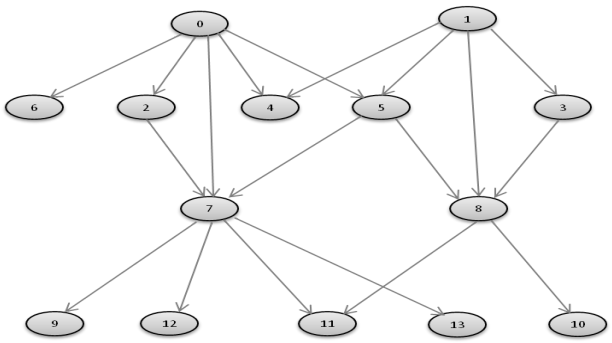


Figure 2.8: The structure of the Tank network

2.7.4 Car Structure

The Car Diagnostic Network consists of 18 nodes and 17 edges. It can be applied to diagnose the malfunctioning of self-propelling vehicles. We got this structure and data sets from [Kabli et al., 2007] and [Javabayes,]. The structure is present in Figure 2.9.

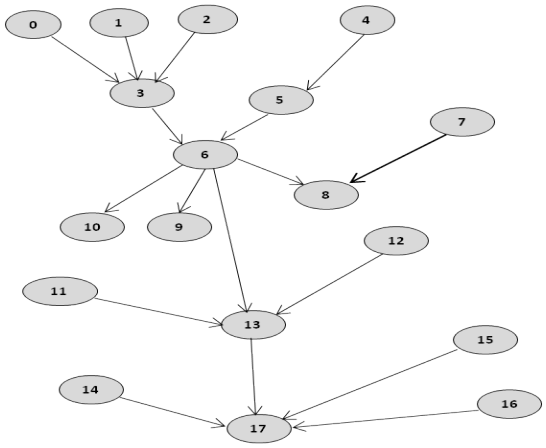


Figure 2.9: The structure of the Car network

2.7.5 B Structure

Alex Kozlov generated B network at random. It contains 18 nodes but 39 edges. Some variables have a high density in edges [Kozlov and Singh, 1996]. The structure is present in Figure 2.10.

2.7.6 Boerlaga Structure

The Boerlaga network has 23 nodes and 36 edges. All variables in this structure has two discrete states [Boerlage, 1992]. The structure is present in Figure 2.11.

2.7.7 Insurance Structure

Insurance structure [Binder et al., 1997] is a model for assessing risks in car insurance. This network has 27 nodes and 52 edges. The number of states number of each

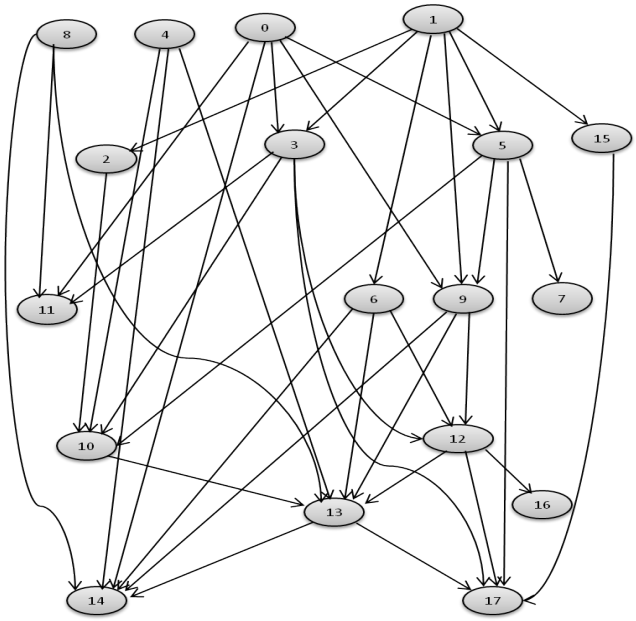


Figure 2.10: The structure of the B network

variable range from 2 to 5. In all 27 nodes, only 15 are observable, and over 1400 parameters. Three of the observable nodes are designed as ”outputs”. The structure is present in Figure 2.12.

2.7.8 Alarm Structure

The Alarm network is a commonly used network which is a representative of a real life Bayesian network. It is a medical diagnostic system for patient monitoring in intensive care. This structure was originally described by Beinlich [Beinlich et al., 1989]. The Alarm network consists of 37 nodes, with 8 diagnoses, 16 findings and 13 intermediate variables. Some nodes have two states, others have three or four states. The structure of Alarm is present in Figure 2.13.

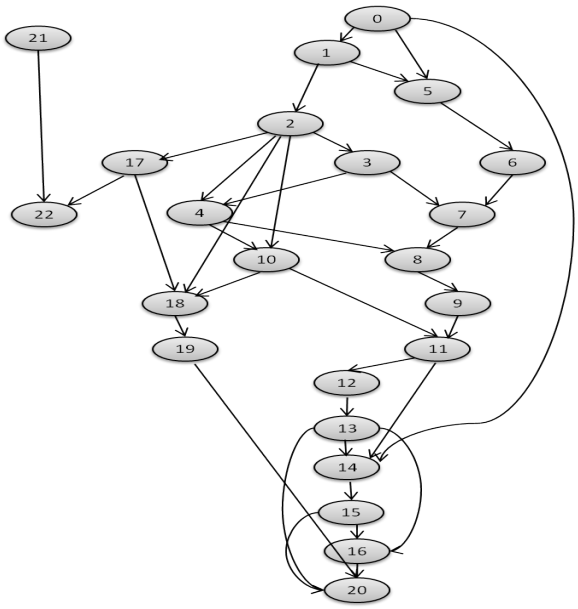


Figure 2.11: The structure of the Boerlaga network

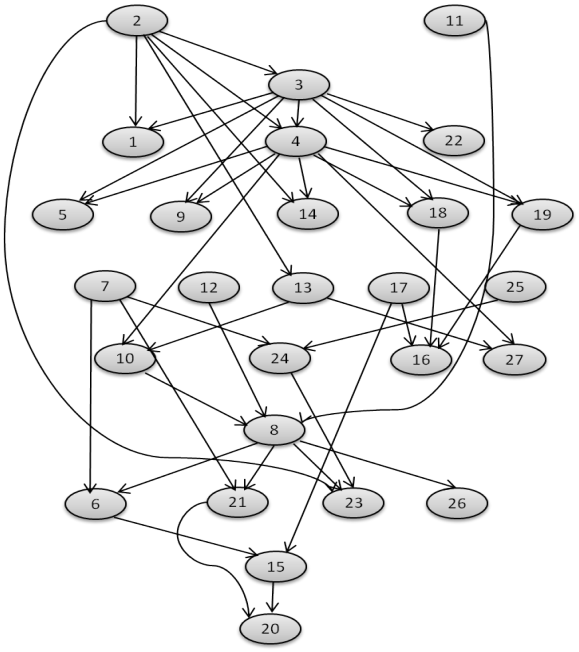


Figure 2.12: The structure of the Insurance network

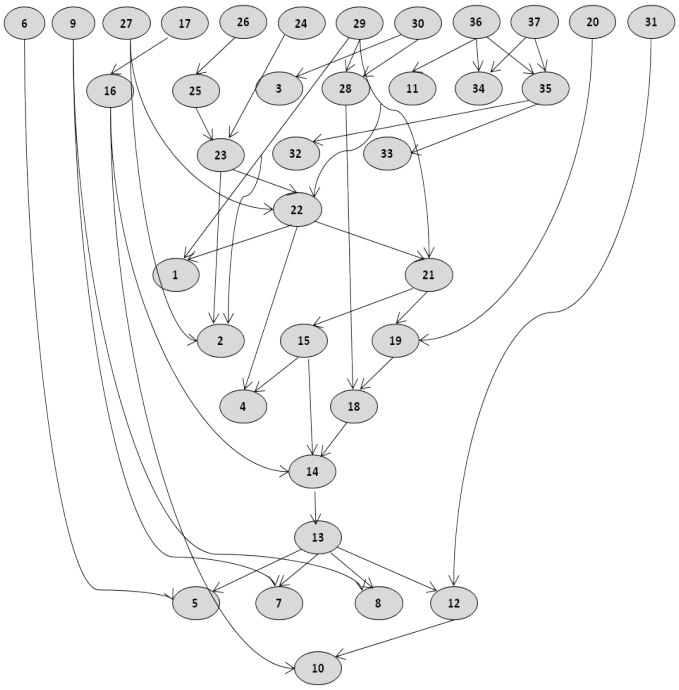


Figure 2.13: The structure of the Alarm network

Chapter 3

ACO with Chain Structure Model and K2 Greedy Search for BN Structure Learning

In this chapter we look at two proposed novel ACO based approaches on BN structure learning. The first one is using ACO to carry out searching with chain structure model in the space of node ordering, which we will call ChainACO algorithm. In the second approach, ACO is introduced to look for the best BN structure when the K2 greedy heuristic algorithm searches in the node ordering space. We will name it as K2ACO algorithm. Afterwards, we implement a series of experiments with results demonstrating the performance of each approach, comparing with other existing methods on some benchmark problems. We execute analysis and draw conclusions to explore the general questions: firstly, how the choice of metaheuristic (ACO and GA) and scoring function (scoring model, chain and CH) relate to the benchmark problems; secondly, what the trade-off is between structure complexity and the approach selected. Some results of this chapter have been published in [Wu et al., 2010b].

3.1 Why ACO ?

We propose two novel approaches using ACO metaheuristic. The first is with chain model structure to generate the best node ordering, afterward the K2 greedy search is executed for BN structure learning. The second one makes use of ACO to optimize the BN structures inducing by K2 search algorithm. We call these algorithms as ChainACO and K2ACO respectively.

There are two reasons for us to select ACO in our approaches. The first one is in both the chain model and the K2 greedy search, the search space is node orderings. The search for an optimal node ordering in this space is similar to the research of the Traveling Salesman Problem (TSP): for a set of cities, the distance between any two cities are known, look for the shortest tour that accesses each point precisely once and return to the starting city at last. We have known that the first successfully application of ACO is to solve this typical NP hard benchmark problem [Dorigo and Gambardella, 1997] [Bianchi et al., 2002]. The second reason is in a lot of previous research, ACO has been selected to solve BN structure learning problem.

The TSP is often assumed to be symmetrical. For example, in seven cities TSP, that means: the sort (1 2 3 4 5 6 7) is assumed to represent the same tour as the string (4 5 6 7 1 2 3). However, the node orderings in our problem do not have this property. Even a tiny change will lead to a complete difference between nodes and their corresponding parents. The similarities between TSP and our problem make us utilize the ACO for our problem. On the other hand, the difference requires us to update the ACO to our problem.

3.2 ACO with Chain Structure Model on BN Structure Learning

3.2.1 Chain Structure Model

From Chapter 2 we know the K2 greedy search is a popular and widely-studied search and scoring algorithm for BN structure learning. The node ordering of variables provides a reduced search space for improving the efficiency of K2 algorithm. Either poor or wrong node ordering will mislead the K2 algorithm, inducing a weak quality solution. For most of BN structure learning problems, the input node ordering is usually unknown. To improve the performance and make the K2 algorithm search more effective, in [Kabli et al., 2007], it proposed a chain model structure, from which to identify nodes ordering from the experimental data.

The main objective of chain structure modeling is to evaluate an initial topological node ordering as input node ordering for K2 greedy algorithm. It is based on the fact that any topological sort corresponds to a chain structure (Figure 3.1), which contains an ordering providing a sufficiently good scoring function to locate highly scored regions in the space of node orderings. In Figure 3.1, the specific chain structure is built by adding edges between successive nodes. Thus, X_{i-1} is the sole parent of X_i , and the arrow expresses the edge from X_{i-1} to X_i. The simplicity of chain model structure makes it cheap for generating the best node ordering on a given database, with evolutionary algorithms or other heuristic search algorithms. The best node ordering generated with higher possibility ensures all parents in the node ordering occur prior to their children in the node ordering (Figure 3.2). This will also make the K2 search algorithm cheap in term of computation.

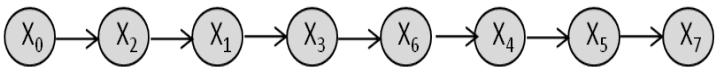


Figure 3.1: Chain mode structure

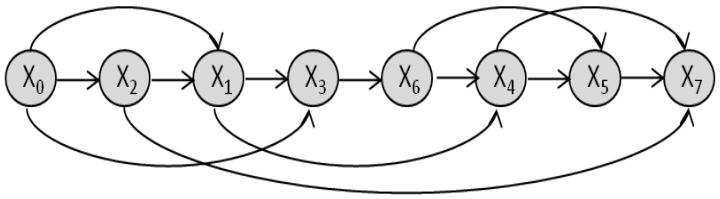


Figure 3.2: Chain mode structure

Normally, a score metric needs to be defined to evaluate the quality of the chain structure when we search it from the space of node orderings. We call it chain score function. The chain structure is still a DAG, can be regarded as a special BN structure. That is, each node in this structure has a sole parent in the chain model. So the chain score looked at the sum of probabilities on each pair of nodes in the chain model from the definition of conditional probability equation(2.13) in the last chapter. In this case, the chain score metric is formulated as (logarithmic format):

$$f_{chain} = \log(\sum_{i=1}^{n-1}(\eta(X_i, P_a(X_i)))) = \log(\sum_{i=1}^{n-1}(\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!)) \tag{3.1}$$

Here, n is the length of chain structure, $\eta(X_i, P_a(X_i))$ is the probability when node X_i has the sole parent $P_a(X_i)$. The best node ordering is the one which has the best chain score value, that is, the closer of the value is to 0, the better the chain model is.

3.2.2 Chain Structure Model with K2 Greedy Search on BN Structure Learning

Once we have used the chain structure model to achieve the best node ordering from the database, this ordering can then be regarded as prior input nodes, which will be fed into the K2 greedy algorithm for inducing the best overall BN structure. This two-stage algorithm contains searching in the space of node ordering and heuristic search for BN structure. However, some empirical experiments have identified that it outperforms other K2 based algorithms and can get a significant reduction in computational cost for large data sets [Kabli et al., 2007].

The pseudocode of Chain-based algorithm is shown in Algorithm 8.

Algorithm 8 Chain model based algorithm of BN structure learning algorithm

Input: A set of n node with database.
Output: for each node, a printout of the parents of the node.
Begin chain algorithm
Initialize population $(P_{old})_n$, $f_{max} = -\infty$;
Chain structure scores $(f_{old})_n$;
for $i \leftarrow 1, n$ **do**
  if $(f_{max} < (f_{old})_i)$ **then**
    $f_{max} = (f_{old})_i$
    $best\_node\_ordering = (P_{old})_i$
  end if
end for
while (terminate=false) **do**
  generating new population $(P_{new})_n$ from $(P_{old})_n$ using **M**;
  calculate new chain structure scores $(f_{new})_n$
  for $i \leftarrow 1, n$ **do**
    if $(f_{max} < (f_{new})_i)$ **then**
      $f_{max} = (f_{new})_i$
      $best\_node\_ordering = (P_{new})_i$
    end if
  end for
end while
Implement K2 algorithm on best_node_ordering to induce the BN structure
Write Node: X_i, parent: π_i
End chain algorithm

Here, M signifies a metaheuristic, which can be GA, ACO, etc.

3.2.3 Chain Model Genetic Algorithm for BN Structure Learning

In order to seek out the best node ordering with chain model structure from the space of node orderings, in [Kabli et al., 2007], they regarded this problem as an optimization problem and employ GA to evolve the best solution. This algorithm is named as ChainGA. In ChainGA, the GA starts by preselecting orderings, rejecting insufficiently good orderings. The ChainGA then adds on another step at the end of each evolution where K2 is run on a percentage of the best orderings to search for a

good structure. The structure of ChainGA is illustrated in Algorithm 9.

Algorithm 9 Pseudo code of ChainGA algorithm

```
Initialize population P, each individual as a chain structure model.
Calculate chain score fitness of each individual.
while (terminate=false) do
    select two individuals from P based on some Fitness criterion.
    apply crossover operator.
    apply mutation operator.
    extend the population by adding the children to it.
    evaluate each individual's fitness.
end while
Select best solution from the population generated.
Implement K2 algorithm on best solution to learn the best structure.
```

3.2.4 Chain Model Ant Colony Optimization Algorithm for BN Structure Learning

We introduce ACO metaheuristic with the chain model structure model for BN structure learning. The problem of learning BN structure with ACO can be stated as (S, f, Ω). The specific characteristics of each composition in the ChainACO approach as follows:

- S, the set of all candidate solutions, is the set of all node orderings with all variables in the investigated database. When the number of variables is n, the size of search space will be $n!$.
- f, the objective function is the function used to score a candidate chain structure $(X_1 - X_2 - \cdots - X_n)$. This function would be the f_{chain} as defined in equation(3.1).
- Ω, the set of constraints, makes sure that the node ordering generating to an new one, which has improved fitness score.

Given the statements to this problem, the structure of ChainACO is illustrated in Algorithm 10. Algorithm 11 describes the process of how to select the first node.

Algorithm 12, is the description of how an ant is able to construct a node ordering with the transition rule once the first node x_0 has been selected. The algorithm can be described by some properties. These properties relate to ChainACO approach described in the following section.

Algorithm 10 Pseudo code of ChainACO algorithm

```
1: Initialize pheromone; number of ants: m
2: Initialize heuristic information.
3: Initialize: each ant ant_i is positioned on a selected (or randomly selected) staring
   node X_0.
4: while (terminate=false) do
5:    for i ← 1, m do
6:       Each ant ant_i applies a state transition rule to incrementally build a solution
         S_i.
7:       A local pheromone updating rule is applied to S_i.
8:    end for
9:    A global pheromone updating rule is applied to S_max.
10: end while
11: Implement K2 algorithm on best solution to learn the best structure.
```

3.3 ACO with K2 Greedy Search on BN Structure Learning

As a greedy search algorithm, when K2 is used in BN structure learning for each node in the database, it iteratively makes one greedy choice after another, reducing the search space into a smaller one. The algorithm itself never reconsiders its choices so far. This makes the K2 algorithm fail to produce the optimal solution, or produce the unique worst possible solution.

Here we propose a new approach which combines ACO and K2 greedy search algorithm to improve the performance of K2 algorithm on BN learning. We name this approach as K2ACO. In this approach, ACO is introduced to look for the best BN structure when the K2 greedy heuristic search in the node ordering space. The similar method is the K2GA approach described in the last chapter. In K2ACO, the

Algorithm 11 Pseudo code of starting nodes selected for each ant in ChainACO algorithm

NodesTovisit = $\{X_1, X_2, \cdots, X_n\}$;
Initialize pheromone $\tau_{0i}(i = 1, 2, \cdots, n)$;
Initialize heuristic information $\eta_{0i}(i = 1, 2, \cdots, n)$.
for $i \leftarrow 1, n$ **do**
  if $(q \leq q_0)(q \in [0, 1])$ **then**
    $max = -\infty$
    if $(\tau_{0i} * [\eta_{0i}[^{\beta}) > max$ **then**
      $max = (\tau_{0i} * [\eta_{0i}]^{\beta}), X_0 = X_i$
    end if
  else
    $prob[i] = (\tau_{0i} * [\eta_{0i}]^{\beta})$
    $Normaliseprob, X_s = draw(prob), X_0 = X_s$
  end if
end for
Output: the selected node X_0 as the starting node.

initial individuals in the population are randomly created node orderings which are then optimized by a colony of ants in this space until a good ordering is found. During the ACO process, the fitness of each ordering is calculated by running the K2 search algorithm on it. Once the optimization terminates, as with K2GA, the structure corresponding to the best ordering found is automatically obtained.

In K2ACO approach, the specific characteristics of each composition in the tripe (S, f, Ω) are described as follows:

- S, the set of all candidate solutions, is the set of node orderings with all variables in the investigated databases. When the number of variable is n, the size of search space will be $n!$.
- f, the objective function is the function used to score a candidate BN structure B. This function would be the CH score function defined in equation(2.12).
- Ω, the set of constraints, makes sure that the BN structure generating to an new

Algorithm 12 Pseudo code of an ant for the construction of an ordering in ChainACO algorithm

1: $Best\_node\_ordering = \Phi$.
2: $Visitednodes = \{X_0\}; NodesTovisit = \{X_1, X_2, \cdots, X_n\}$.
3: **while** $(|NodesTovisit| > 0)$ **do**
4: **for** $i \leftarrow 1, |NodesTovisit|$ **do**
5: **if** $(q \leq q_0)(q \in [0,1])$ **then**
6: $max = -\infty$
7: **if** $(\tau_{0i} * [\eta_{0i}]^{\beta}) > max$ **then**
8: $max = (\tau_{0i} * [\eta_{0i}]^{\beta}), X_0 = X_i$
9: $Visitednodes = Visitednodes \cup \{X_i\}$
10: $NodesTovisit = NodesTovisit \setminus \{X_i\}$.
11: $Best\_node\_ordering = Best\_node\_ordering \cup \{X_0\}$.
12: **end if**
13: **else**
14: $prob[i] = (\tau_{0i} * [\eta_{0i}]^{\beta})$
15: $Normaliseprob, X_s = draw(prob), X_0 = X_s$
16: $Visitednodes = Visitednodes \cup \{X_i\}$;
17: $NodesTovisit = NodesTovisit \setminus \{X_i\}$.
18: $Best\_node\_ordering = Best\_node\_ordering \cup \{X_0\}$.
19: **end if**
20: **end for**
21: **end while**
22: **Output:**Best_node_ordering

one, which has improved fitness score of the structure.

The Pseudo code of K2ACO is illustrated in Algorithm 13. The process of how to select the first node and how an ant to construct an node ordering with transition rule once the first node X_0 has been selected are described in Algorithm 11 and Algorithm 12 respectively.

Algorithm 13 Pseudo code of K2ACO algorithm

1: Initialize pheromone; number of ants: m
2: Initialize heuristic information.
3: Initialize: each ant ant_i is positioned on a selected [or randomly selected] staring node X_0.
4: **while** (terminate=false) **do**
5: **for** $i \leftarrow 1, m$ **do**
6: ant_i applies a state transition rule to incrementally build a node ordering S_i.
7: Implement K2 algorithm to S_i to learn the corresponding BN structure and the CH score.
8: A local pheromone updating rule is applied to S_i.
9: **end for**
10: A global pheromone updating rule is applied to the best solution S_{max}.
11: **end while**

3.4 Properties of ACO Metaheuristic in ChainACO and K2ACO

In this section, we give a detailed description on the main components of the two ACO based algorithms. It contains the heuristic information, the initial pheromone, the node transitions rules, the pheromone updating rules and the starting nodes selected in node ordering. We mainly describe how these components are defined. Some of these are defined commonly as the definition in TSP, others, however, are only from the specific problem we discuss.

3.4.1 Heuristic Information

In ACO algorithm, normally, the heuristic information is static and calculated before any ants is launched. In the chain structure model problem, the heuristic information that has to be considered when studying the possibility of node i is the sole parent of node j, $\eta_{ij}(i, j = 0, 1, \cdots, (n-1))$. We record all the values of heuristic information η_{ij} as a $n \times n$ matrix:

$$\eta_{ij} = \begin{pmatrix} \eta_{00} & \eta_{01} & \cdots & \eta_{0,n-1} \\ \eta_{10} & \eta_{11} & \cdots & \eta_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ \eta_{n-1,0} & \eta_{n-1,1} & \cdots & \eta_{n-1,n-1} \end{pmatrix}$$

when $i = j$,

$$\eta_{ij} = \eta_{ii} = 0,$$

If $i \neq j$, we assign the value of η_{ij} with the value

$$P(x_i, x_j) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{3.2}$$

because we use the logarithmic format of this equation, it is a negative value. In this case, the value of η_{ij} is set as

$$\eta_{ij} = \frac{1}{|P(x_i, x_j)|} \tag{3.3}$$

3.4.2 Pheromone Initialization

In ACO algorithm, a small amount of pheromone τ_0, is deposited in every link of the graph. Normally, this value is calculated through a quick method, for example, greedy algorithm, before the ants' operating. In our case, the initial amount of pheromone is achieved as

$$\tau_0 = \frac{1}{n|f(G_{K2GA})|}$$

where n is the number of variables, and $f(G_{K2GA})$ is the score of BN structure obtained by K2GA algorithm. The reason of expression $|\cdot|$ for fitness score is that the $f(G_{K2GA})$ always is negative. The closer of this absolute value to zero, the better the score is. The best score of $f(G_{K2GA})$ means the highest initialization pheromone value τ_0. In order to ensure accuracy and fairness, within independent repeated trials in our experiments, we design the initialization pheromone τ_0 as a fixed value.

3.4.3 Node Transitions Rules

Ant in node i moves to node j according to the probabilistic state transition rule,

$$s = \begin{cases} argmax_{j \in J_K(i)}\{\tau_{ij}[\eta_{ij}]^\beta\} & \text{if } q \leq q_0, \\ S & \text{if } q > q_0. \end{cases} \tag{3.4}$$

where τ_{ij} is the pheromone amount deposited on the arc which connects node i with node j. η_{ij} represents heuristic information, which is a weighting function that assigns at each construction step a heuristic value to each feasible solution between any two nodes. q is a random number uniformly distributed in $[0, 1]$, $q_0 (0 \leq q_0 \leq 1)$ is a parameter that determines the relative importance of exploitation versus exploration, β is a parameter which control the relative importance of pheromone verse the probability, and S is a random variable selected according to an normalize probability function, $S = draw(prob)$. This probability given in equation(3.5), here $\alpha = 1$

$$P_{ij} = \begin{cases} \frac{(\tau_{ij})^\alpha [\eta_{ij}]^\beta}{\sum_{l \in u} (\tau_{il})^\alpha [\eta_{il}]^\beta} & \text{if } j \in J_K(i), \\ 0 & otherwise. \end{cases} \tag{3.5}$$

3.4.4 Local and Global Updating Rules

Ants prefer to move to nodes which are connected by high possibility and a high amount of pheromone. Every time that an edge is chosen by one ant, the amount of

pheromone is changed by applying the local updating rule:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0 \tag{3.6}$$

The local updating rule is intended to avoid a very strong edge being chosen by all the ants. Once all ants have completed their tours a global pheromone updating rule (3.7) is applied:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\sigma_{ij} \tag{3.7}$$

where

$$\Delta\sigma_{ij} = \begin{cases} \frac{1}{C(S^+)} & \text{if } ij \in S^+ \\ \tau_{ij} & \text{if } ij \notin S^+ \end{cases}$$

In (3.6) and (3.7), both ρ and α are parameters which control the pheromone evaporation and $C(S^+)$ is the cost associated with the best solution.

3.4.5 Starting Nodes Selected in Node Ordering

The starting node is the root node of a sequence for each ant. It will influence the selection of other nodes on the subsequent steps. If the starting node can be selected in advance, the possible search space for all ants will be narrowed from the set of $n!$ to $(n-1)!$. To select the starting node, we investigate the chain structure model, and suppose it has an empty node as the parent to the first node (Figure 3.3). We note this first node as X_0.

Figure 3.3: Chain mode structure with empty starting node

We can then calculate the value of the heuristic information within the empty parent node X_0 and the first node X_1 . In equation(3.2), if $P_a(X_1) = \phi$, then $q_i = 0$, the product ranges from $j = 1$ to $j = 0$. The standard convention for a product with an

upper limit smaller than the lower limit is that the value of the product is equal to 1. Hence, the heuristic information function for CH score will be

$$\eta_{oj} = \frac{1}{|P(x_i, \phi)|} = \frac{1}{|\log(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i} N_{ijk}!)|}. \tag{3.8}$$

We suppose each node $X_i(i = 1, 2, \cdots, n)$ has the same chance to have an empty parent, and then we can calculate the corresponding value of heuristic information $\eta(0, j)(i = 1, 2, \cdots, n)$ through equation(3.8). We then utilize the node transitions rules to decide which node will be the first one in the node ordering.

3.5 Experimental Methodology

In this section, we design and run a number of experiments that apply with ChainACO and K2ACO approaches on benchmark structures. We also run comparative experiments on ChainGA and K2GA algorithms. One of the objectives is to measure the performance of the two proposed algorithms. These include measuring the computational complexity and the accuracy of the algorithms. We are also interested in the differences between algorithms based on ACO (ACO algorithms) and GA (GA algorithms), algorithms based on Chain (Chain algorithms) and K2 (K2 algorithms) respectively.

As previously described, we know both ChainACO and ChainGA algorithms have two phases. K2ACO and K2GA algorithms, on the other hand, each has a single phase. In this experiment, we set a fixed number of iterations on the ACO and GA processes in the ChainACO and ChainGA approaches. The node ordering generated then transits to the second. For K2ACO and K2GA algorithms, we specify a stopping condition T_0 according to previous experimental results in [Kabli et al., 2007]. We run the ACO and GA iterations under the condition of the fitness of BN structure less than T_0 respectively to construct the best structures and get the best fitness at the same time.

We design two series of experiments according to the different data sampling method utilized. They are experiment **A**, by the likelihood sampling method, and experiment **B**, through probabilistic logic sampling.

For each algorithm, we record and compare the algorithms in terms of the following outcomes:

1. The value of the CH metric (log version) in equation (3.9). We know that this score is associated with the best structure, as the score is the logarithm of a probability, and so it is a negative number. The closer to zero, the closer the probability is to 1. The higher the probability, the better the network.

$$g_{K2}(G, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} [\log(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!}) + \sum_{k=1}^{r_i} \log(N_{ijk}!)] \tag{3.9}$$

2. The number of factor evaluation (**F.E.**). **F.E.** is utilized to evaluate the efficiency of the algorithms. It is defined as being the count of times the term (3.10) is accessed when Formula (3.9) is used. Equation(3.10) is the function used in K2 algorithm. The logarithmic version of this is

$$f_{K2}(X_i, P_{ai}) = \log\{\prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!\} \tag{3.10}$$

   In ChainACO algorithm, the **F.E.** comes from three parts: the evaluations needed to generate initial heuristic information η_{ij} for node transitions rules, the evaluations needed to calculate chain structure scores in phase 1, and the evaluations needed in the K2 search phase for obtaining the structure score. In ChainGA, the **F.E.** score is made up of two parts: the evaluations needed in the chain structure phase and those in the K2 search phase. In K2ACO, the number of **F.E.** includes those needed for the initial heuristic information η_{ij} for node transitions rules and those needed for the CH score evaluations. In K2GA, the **F.E.** score is calculated solely from the CH score evaluations needed.

3. Comparison of structure difference between the learned network and the original network. This includes the number of arcs deleted (**D**), added (**A**) and inverted (**I**), comparing with the original network (Figure 3.4). The total number of $H = D + A + I$, called structural hamming distance (**SHD**) [Tsamardinos et al., 2006b] that counts the number of differences in the completed partially DAG of the true network and the learned one.

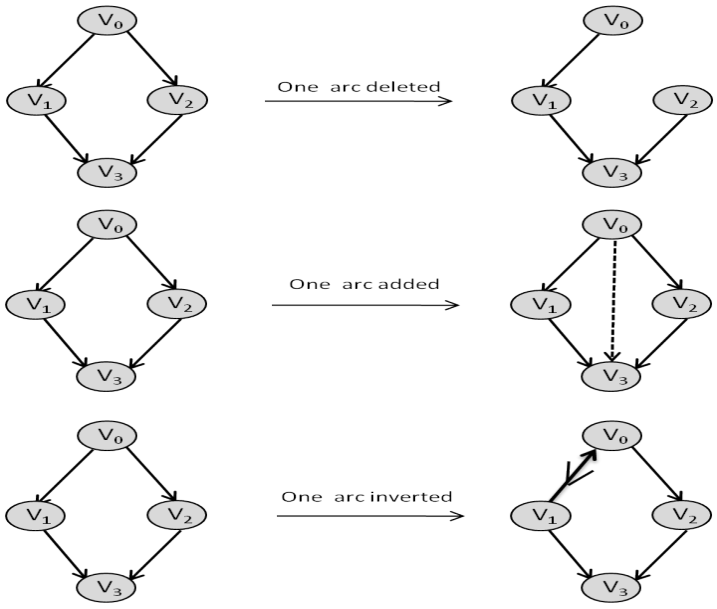


Figure 3.4: Process of one arc deleted, added and inverted

3.5.1 Experiment A

We use three benchmarks, Asia, Car and Alarm, in our experiments. The dataset sizes for Asia, Car and Alarm are 5000, 10000 and 3000 cases respectively. In all cases the scoring metric used to evaluate the search is the CH metric. The parameters used in ChainACO and K2ACO approaches are shown in Table 3.1. The parameter settings of ChainGA and K2GA are shown in Table 3.2. These parameters are the same, for the GA algorithms, as those used in [Kabli et al., 2007].

We have carried out 30 runs of each algorithm for each benchmark problem considered. The best score, **F.E** and their corresponding standard deviation are averaged

Table 3.1: The ACO Parameters Settings

ACO Setting	Asia	Car	Alarm
Number of ants	6	8	8
Number of iterations	30	30	30
q_0	0.8	0.9	0.95
β	2	2	3
ρ, α	0.1	0.1	0.1
Number of runs	30	30	30

Table 3.2: The GA Parameters Settings

GA Setting	Asia	Car	Alarm
Evolution type	Steady State		
Number of offspring	1		
selection	Rank selection		
Crossover	Cycle Crossover, rate:0.9		
Mutation	Displacement Mutation, rate:0.1		
Population size	100	20	10
Number of runs	30	30	30

over 30 runs. Success rates (**S.R.**) are obtained through the comparison of the score of the best structure in each run with the target score. In our experiments, the target is the stopping condition T_0 , for example, in Asia, Car and Alarm networks, $T_0 = -11150.00$, $T_0 = -23175.00$, $T_0 = -30110.00$ respectively. The arcs which we achieved correspond to the best-ever score obtained by each algorithm.

Table 3.3 displays the results obtained for the Asia, Car and Alarm networks, respectively. Table 3.4 displays normalized structural differences from the original network for each algorithm on each benchmark. This includes arcs added (A), arcs deleted (D), and arcs inverted (I). "Normalized" means here that, on each problem, the absolute number of each structural difference measured is divided by the corresponding number for ChainACO. Thus, a normalized value smaller than 1.00 denotes the corresponding approach performs better than ChainACO, whereas a value greater than 1.00 indicates a weaker performance than ChainACO. With the aim of comparing

Table 3.3: EXPERIMENTAL RESULTS

Structures	Approach	Avg.Score	F.E.	S.R.	SHD
Asia	ChainACO	−11167.33 ± 18.89	128.0 ± 4.54	0.50	8
	K2ACO	−11146.22 ± 1.85	1166.3 ± 630.05	0.95	8
	ChainGA	−11153.95 ± 12.16	1133.43 ± 197.24	0.57	8
	K2GA	−11145.99 ± 1.22	2468.73 ± 15.33	**1.00**	8
Car	ChainACO	−23186.77 ± 18.04	2083.1 ± 263.17	0.30	23
	K2ACO	−23163.83 ± 4.07	2707.83 ± 1784.1	0.60	20
	ChainGA	−23270.00 ± 126.75	964.83 ± 198.46	0.25	20
	K2GA	−23163.17 ± 7.52	1774.63 ± 468.07	**1.00**	20
Alarm	ChainACO	−29814.20 ± 90.99	2480.07 ± 1252	**1.00**	54
	K2ACO	−30073.3 ± 094.99	7225.07 ± 3405.1	0.57	68
	ChainGA	−30443.6 ± 220.96	1235.93 ± 205.85	0.10	68
	K2GA	−30087.4 ± 182.31	2596.97 ± 497.15	0.53	63

overall computational efficiency, Table 3.5 displays the success rates per 10000 **F.E.** for each algorithm. Figure 3.5, Figure 3.8 and Figure 3.11 plot the best score values over all runs for each algorithm for the Asia, Car, and Alarm networks respectively. The best structure graphs got by ChainACO, K2ACO, ChainGA and K2GA on Asia and Car networks are present in Figure 3.7a– 3.7d and Figure 3.10a– 3.10d respectively.

We now make detailed observations on the results for each benchmark problem:

3.5.1.1 Asia Structure

For the Asia dataset, we carried out a one way ANOVA test using the Bonferroni correction with the four algorithms to compare the averaged score [Field, 2009]. The results indicate that there is a significant difference between the ChainACO approach and the other three approaches, $F(3, 116) = 22.819, P < 0.05$. In spite of the K2GA obtaining the best average structure score, the result of ANOVA test indicates that there is not a significant difference between it and the K2ACO approach. Figure 3.5 shows the scatter points of these values. The Box plots of the best scores found for the four algorithms across all experiments are displayed in Figure 3.6. This figure shows

Table 3.4: Normalized structural differences within four algorithms to three benchmarks model

		ChainACO	K2ACO	ChainGA	K2GA
Asia	A	1.00	1.00	1.00	1.00
	D	1.00	1.00	1.00	1.00
	I	1.00	0.25	0.25	0.25
Car	A	1.00	0.50	0.63	0.50
	D	1.00	0.50	1.00	0.50
	I	1.00	1.00	1.00	1.25
Alarm	A	1.00	1.25	1.29	1.18
	D	1.00	0.95	1.00	1.00
	I	1.00	1.67	0.67	1.00

Table 3.5: Success Rates per 10,000 F.E.

Approach	Asia	Car	Alarm
ChainACO	**39.06**	1.44	**4.03**
K2ACO	7.72	2.22	0.79
ChainGA	5.03	2.59	0.81
K2GA	4.05	**5.64**	2.04

K2 algorithms have the highest structure score, also a very high number of values on K2 algorithms within a very small segment of the sample.

With respect to the number of **F.E.**, K2GA costs the biggest evaluation, almost twenty times that required by ChainACO. ChainACO is the cheapest one; however, the success rate is the lowest. In order to show the trade-offs between efficiency and success rate, the success rate per 10000 function evaluations is shown across all experiments in Table 3.5. We observe that ChainACO provides the best trade-off for the Asia network.

Comparing the Chain algorithms and K2 algorithms, we can find that the former is cheaper in the consumption of **F.E.**. This result is in accordance with the hypothesis that a chain model is a sufficiently good model to locate node orderings of which a good BN structure can be built. The best structure graphs of each approach achieved

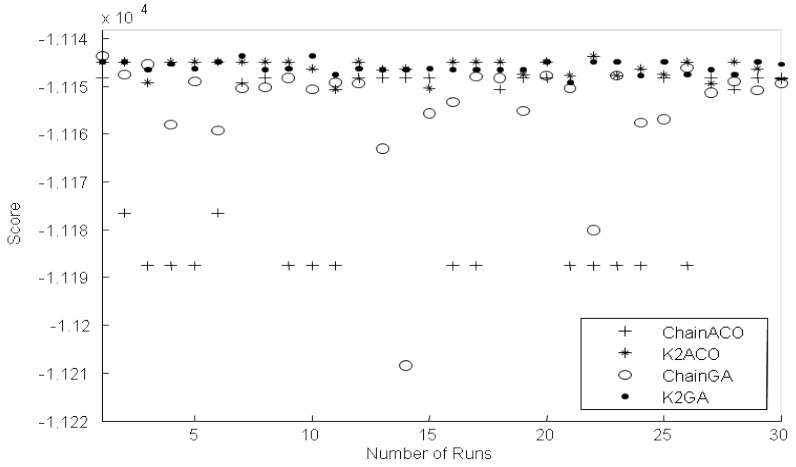


Figure 3.5: Best found Structures at Each Run for all Algorithms to Asia Network

are shown in Figure 3.7a– 3.7d. These graphs indicate that each algorithm finds the same number of arcs. However, the structure achieved by ChainACO is far away from the original one, it has more inverted arcs than the three others. In these figures, an edge with the same direction arrow added means the original arc found by the approach; an edge with the opposite direction arrow represents this original edge found but invested. A dotted arrow is the edge added to the true structure, and the solid directed edges with no central arrow represent edges that occur in the true structure but are not achieved by the corresponding approach. Table 3.4 shows these differences on arcs.

3.5.1.2 Car Structure:

For the Car diagnosis problem, the comparative results shown in Table 3.3 are notably different from those obtained on the Asia network. The one way ANOVA Bonferroni test with the four algorithms to compare the averaged score show that there is a significant difference between the K2 approaches and the Chain approaches, $F(3, 116) = 18.035, P < 0.05$. The former performs better than the latter. Figure 3.8 shows the scatter points of these values. Box plots of the best scores over all runs for each algorithm are displayed in Figure 3.9. This box plot shows that the val-

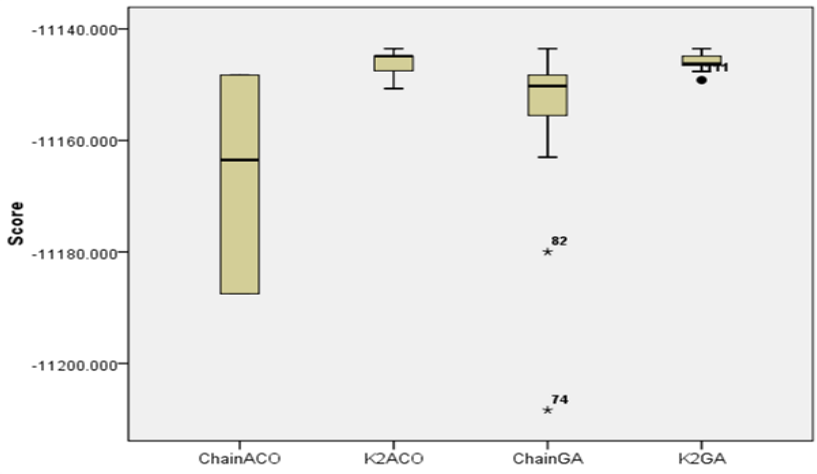


Figure 3.6: Box-plots of the Best found Structure Score for Asia at each run for all algorithms

ues achieved by all approaches other than ChainGA are concentrated within a very narrow range.

The success rates in Table 3.3 are also observably different. Comparing with K2GA, ChainACO and ChainGA have very low success rates. Therefore we might need to re-start these algorithms three and four times respectively to get the same quality of result as K2GA. From Table 3.5, the success rate per 10000 function evaluations indicates that the K2GA performs better than the three others. ChainACO obtained the lowest rate; the value is only 1.44.

The structural arcs found in Table 3.3 and Table 3.4 show that ChainACO has more added and deleted arcs than the other three algorithms. Figure 3.10a– 3.10d show the structures obtained and the difference among these approaches.

From above analysis, we can conclude that the results for the Car problem are that the GA outperforms ACO in computational effort and structure quality.

3.5.1.3 Alarm Structure:

For the Alarm problem, the difference is apparent in the averaged best score, F.E, success rate, and the arc found between ChainACO and other three algorithms. ANOVA

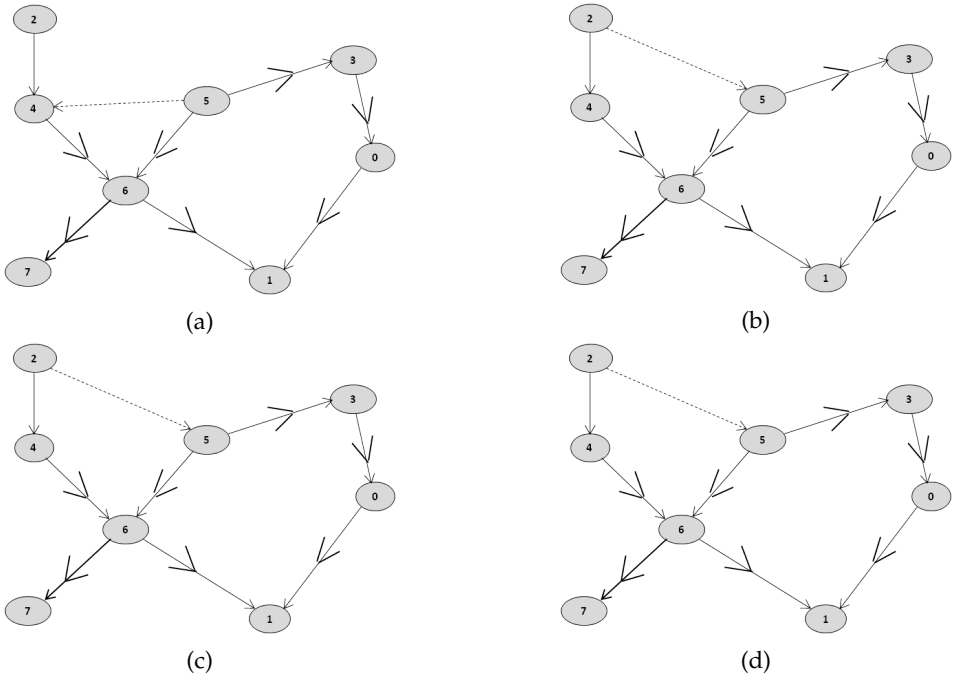


Figure 3.7: Learned Asia Structures with All Algorithms

tests using the Bonferroni correction show there is a significant difference between the ChainACO approach and the other three approaches, $F(3, 116) = 80.726, P < 0.05$. Figure 3.11 shows the scatter points of these values. The comparison of best scores found for the four algorithms are displayed in Figure 3.12. This plot-box differs from the one on Aisa and Car structures. All of them are relatively wider, however, it shows that ChainACO has a significant performance on a highest BN score value, and the distribution of these values with a thinner peak.

From Table 3.4, we can find that the structural errors between ChainACO and the other three algorithms also have significant differences.

Table 3.5 shows that ChainACO obtained the best trade-off. It is interesting to note that K2ACO and ChainGA have significant differences in factor evaluation and success rates but very similar ratios. Both of them are only one fifths of the ChainACO's result. The experimental results for Alarm data also show that the K2 algorithms need more time in factor evaluation than Chain algorithms. The number of **F.E.** in K2ACO

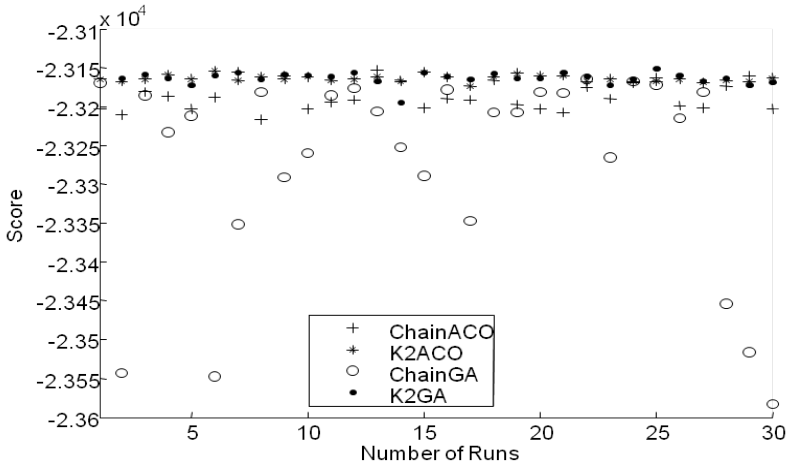


Figure 3.8: Best found Structures at Each Run for all Algorithms to Car Network

is nearly three times that for ChainACO, and in K2GA is nearly twice to ChainGA.

All results in Alarm show that the Chain algorithm with ACO metaheuristic outperforms GA based algorithms.

3.5.2 Experiment B

We select the eight benchmark problems which we discussed in the last chapter in experiment **B**. All the data sets to these benchmarks are sampled from the GeNIe&SMILE [GeNIe and SMILE,]tool, but using the probabilistic logic sampling method. We use the same experimental parameters setting as in experiment A. The data size for Asia is $10,000$, for all other problems, the size are $3,000$.

We have carried out 30 runs of each algorithm to each benchmark problem considered. Main results are shown in Table 3.7 ~ Table 3.9. These results indicate all the values of CH score, **F.E.** with the averaged and corresponding standard deviation. The value in brackets is the best one in each algorithm, and the bold value is the best one within the four algorithms.

We compare the results of the score fitness on learned BN structures first. To investigate these structures, we carried out a one way ANOVA tests using the Bonferroni correction with the four algorithms to compare the averaged score in each structure.

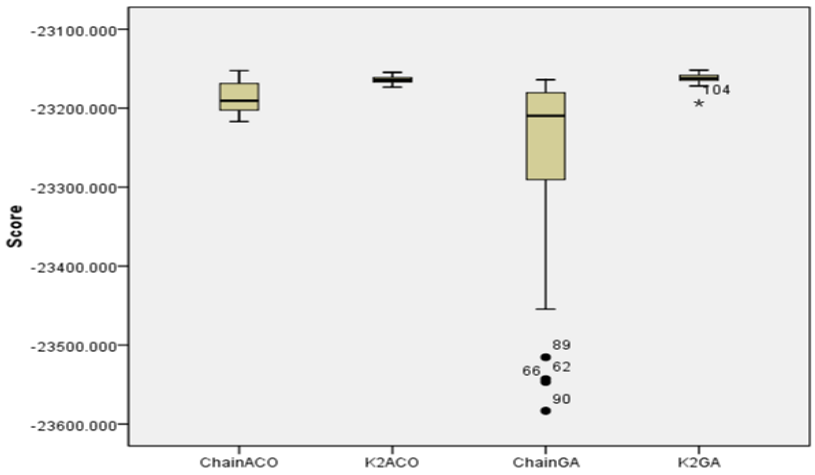


Figure 3.9: Box-plots of the Best found Structure Score for Car at each run for all algorithms

The results indicate, at the 0.05 level, that there is no significant difference between the ChainACO approach and the K2 based approaches in Asia, Car, Boerlaga and Alarm structures. There are significant differences between the ChainGA approach and the K2 based approaches in all structures, except in Car structure there is not significant difference. Within K2 based algorithms, the results show there is no significant difference between ACO algorithms and GA algorithms. In structures Credit, Tank, B and Insurance, the ANOVA results indicate that there is significant difference between Chain based algorithms and K2 based algorithms. K2ACO and K2GA algorithms perform better in recovering the structures. The values of these ANOVA testing are present in Table 3.6.

Table 3.6: The values of bonferroni ANOVA testing on benchmarks

Structure	Asia	Credit	Tank	Car	B	Boerlaga	Insurance	Alarm
$F_{0.05}$	9.127	16.402	20.009	0.692	21.162	8.078	26.525	8.350

A very clear performance from comparing **F.E.** in all structures that can be found is that the chain structure model algorithms are cheaper than K2 based algorithm. The values of **F.E.** within four algorithms have a significant difference. With the increasing

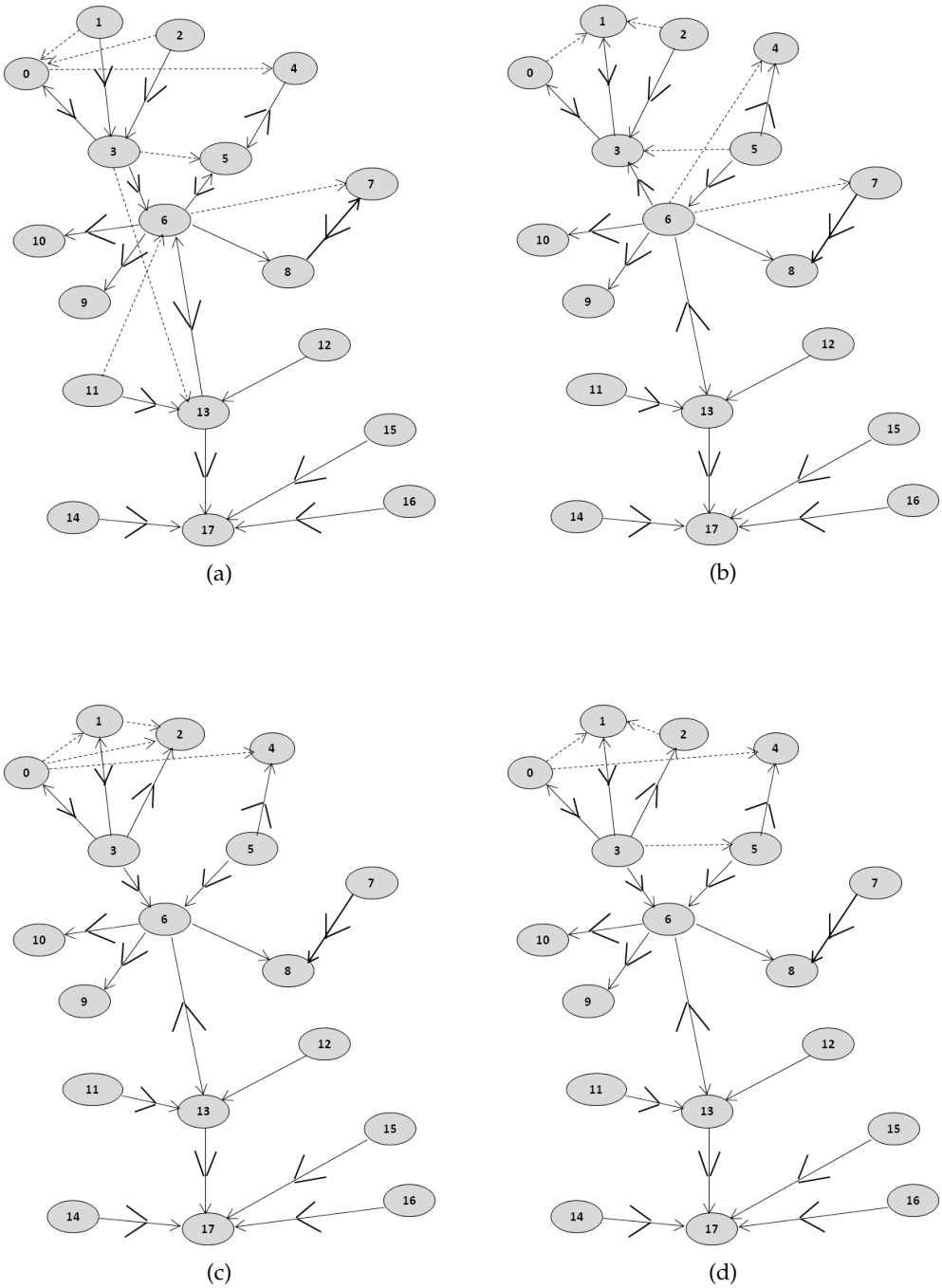


Figure 3.10: Learned Car Structures on All Algorithms

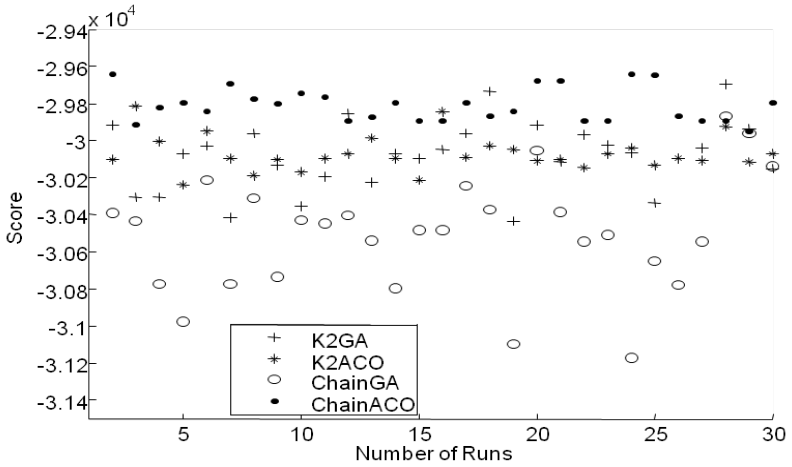


Figure 3.11: Best found Structures at Each Run for all Algorithms to Alarm Network

of nodes in structures, this difference becomes larger. The amount in K2 algorithms is 50 times larger than that in the Chain algorithms with Car and Boerlaga problems. Within all chain based algorithms, ACO performs better than GA in all structures. Within K2 based algorithms, in most of these problems, ACO takes more time than GA in time consumption. In all these structures, ChainACO is the cheapest approach and K2ACO is the most expensive one in computing consumption. Figure 3.13 shows F.E. to the comparison of all algorithms with the eight benchmarks.

We do not intend to give the detailed graphs of the structure learned by each approach. Instead of these we present the values of structural hamming distance (SHD), which includes the arcs deleted (D), invested (I), and added (A) in Table 3.7 ~ Table 3.9.

We append the ANOVA testing results and the Box-plot graphs on each problem in appendix A.

3.6 Summary

In this chapter, we present two new search and score algorithms for BN structure learning. Each of them utilizes the ACO metaheuristic to guide the search of node

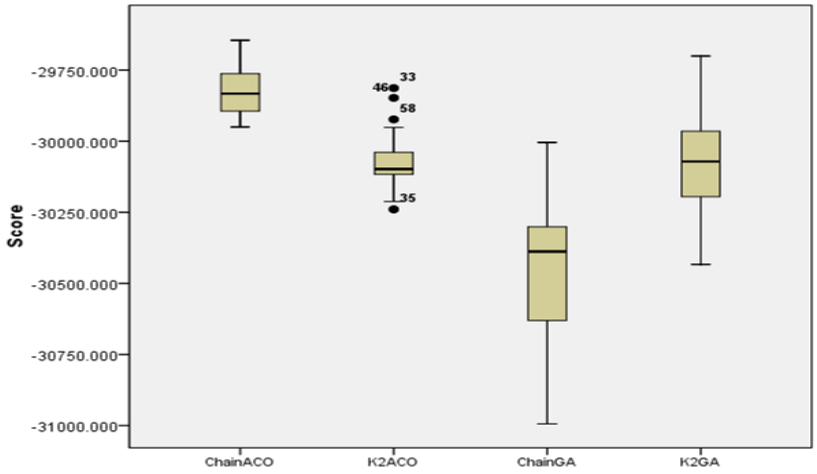


Figure 3.12: Box-plots of the Best found Structure Score for Alarm at each run for all algorithms

orderings. ChainACO uses ACO evolving on the chain structure models. This reduces the computational expense of the search but with a penalty on the success in retrieving structure on known benchmarks. K2ACO also searches node orderings using the K2 algorithm directly to construct and score locally optimized structures. K2ACO is more computationally expensive but in general has a higher success rate in recovering the known structures.

We applied our approaches to eight different benchmark problems of varying complexities, which simulated through two different kinds of sampling methods. We also compared our methods to two existing approaches which also search node orderings but utilizing GA.

Comparing the results in experiments both A and B, one obvious result is there is not significant difference between likelihood sampling and probabilistic logic sampling. The performance of each algorithm, such as in score fitness, **F.E.**, is not affected by the sampling methods. We drew some conclusions from these experiments as follows:

These results indicate that there is a high degree of problem dependency both in the effectiveness and the efficiency of the approaches used. Both the choice of

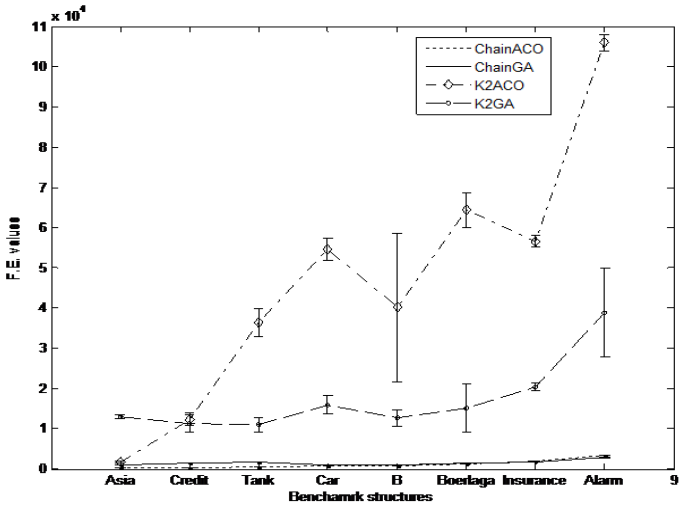


Figure 3.13: Comparing of F.E in each structure for all algorithms

metaheuristic (ACO and GA) and the choice of scoring method (Chain scoring and CH scoring) can significantly affect performance. ACO with Chain model, is the cheapest one in all algorithms. ACO with K2 greedy search becomes very expensive, although sometimes can achieve a better BN score. The reason for it is that mostly the K2 is a heuristic search algorithm, it needs large numbers of iterations in the process of operation. GA is relatively cheap comparing to ACO when they are applied with the K2 greedy search algorithm, but the difference in fitness score is not significant. To all structures, the chain scoring based algorithms are always more computationally efficient, but with a penalty on the success in retrieving structure. The results also indicate in some situations the chain scoring approach is likely to be unsuccessful in producing high quality structures (for example, in Credit, Tank, B and Insurance structures, there are significant difference between the Chain based approaches and the K2 based approaches to score values), and so the relative benefit of reduced computational time is lost. However experiments on the overall distribution and success of the four algorithms showed that the scoring approach, as opposed to the choice of metaheuristic, is the dominant influencing factor. This motivates analysis

of the interaction between scoring metric and problem structure.

We also conclude that, for some problems, repeated restarts will be beneficial. This suggests that one way forward is to develop hyperheuristic approaches that adapt themselves to a particular problem as they learn. An important aspect of this will be to understand how particular structural features interact with particular heuristics to either promote or inhibit learning.

There are some parameters provided for ACO based algorithms in our experiments. We understand that the results achieved from these experiments are related to the setting of these experiments in some degree. In the future research, we should consider these different parameters.

The pheromone initialisation parameters provided for ChainACO and K2ACO are based on a single prior run of K2GA. This was done for convenience and consistency between experiments but raises questions about the independence of ChainACO and K2ACO from K2GA. In fact, other approaches exist to generate the initial pheromone value τ_0 that we could equally well have used. In particular, in [de Campos et al., 2002], the K2SN algorithm is used to generate initial pheromone values. The possible alternative method to set this value is to use the greedy search algorithm, K2 algorithm before starting the ACO process.

The results in this chapter illustrated that both the choice of metaheuristic (ACO and GA) and the choice of scoring method (Chain scoring and CH scoring) can significantly affect performance. In the following chapters, we will investigate how the pheromone depends on each problem for specific benchmark structures.

Table 3.7: EXPERIMENTAL RESULTS(Asia, Credit, Tank)

Structure	Approach	A	D	I	SHD	Avg.Score	F.E.
Asia	ChainACO	2.4 ± 1.42 (0)	0.36 ± 0.48 (0)	3.66 ± 1.13 (1)	6.43 ± 2.10 (1)	−22531.83 ± 15.43 (−22518.55)	143.2 ± 3.45 (138.0)
	ChainGA	4.26 ± 2.58 (1)	0.6 ± 0.67 (1)	4.00 ± 1.32 (2)	8.9 ± 1.32 (4)	−22559.68 ± 39.36 (−22523.94)	886.96 ± 6.82 (876)
	K2ACO	0.7 ± 0.64 (0)	0 ± 0 (0)	2.6 ± 0.8 (1)	3.25 ± 1.41 (1)	−22522.37 ± 3.11 (−22518.55)	1705.7 ± 188.83 (1368)
	K2GA	0.2 ± 0.4 (0)	0 ± 0 (0)	2.3 ± 0.64 (2)	2.5 ± 1.02 (2)	−22519.28 ± 1.47 (−22518.55)	12979.3 ± 510.94 (12808)
Credit	ChainACO	7.16 ± 1.79 (3)	0.3 ± 0.45 (1)	6.7 ± 1.50 (4)	14.2 ± 2.84 (8)	−31889.9 ± 76.33 (−31786.71)	342.36 ± 7.64 (308)
	ChainGA	6.6 ± 1.8 (6)	0.1 ± 0.3 (0)	6.8 ± 2.13 (5)	13.5 ± 3.67 (11)	−31935.48 ± 77.31 (−31835.27)	1408.7 ± 7.15 (1395)
	K2ACO	2.0 ± 0.63 (1)	0.5 ± 0.5 (0)	4.1 ± 1.44 (3)	6.6 ± 1.8 (4)	−31768.31 ± 6.54 (−31756.14)	12306.9 ± 1596.00 (13523)
	K2GA	2.3 ± 0.9 (1)	0.4 ± 0.48 (0)	3.8 ± 1.53 (2)	6.5 ± 2.5 (3)	−31775.5 ± 11.59 (−31756.96)	11322.6 ± 2152.68 (8970)
Tank	ChainACO	13.83 ± 3.65 (7)	2.43 ± 1.76 (1)	8.3 ± 1.77 (7)	24.56 ± 5.65 (15)	−10092.69 ± 102.3 (−9952.59)	505.46 ± 15.35 (472.0)
	ChainGA	15.86 ± 4.49 (6)	3.2 ± 1.55 (0)	7.0 ± 2.08 (5)	26.06 ± 6.58 (11)	−10202.19 ± 194.3 (−9944.71)	1719.5 ± 20.60 (1705)
	K2ACO	8.7 ± 1.87 (6)	0.55 ± 0.66 (0)	7.3 ± 2.66 (5)	16.55 ± 3.1 (11)	−9954.61 ± 13.70 (−9941.73)	36300.75 ± 3539 (30285.0)
	K2GA	7.9 ± 2.16 (6)	0.4 ± 0.66 (0)	6.1 ± 1.92 (4)	14.4 ± 3.46 (10)	−9953.09 ± 9.57 (−9942.09)	10917.5 ± 1768 (10882)

Table 3.8: EXPERIMENTAL RESULTS(Car, B, Boerlaga)

Structure	Approach	A	D	I	SHD	Avg.Score	F.E.
Car	ChainACO	9.8 ± 1.2 (8)	3.45 ± 0.97 (2)	4.95 ± 0.86 (3)	18.2 ± 2.61 (13)	−7126.05 ± 2.32 (−7120.61)	787.1 ± 9.61 (732)
	ChainGA	8.1 ± 1.91 (7)	5.5 ± 2.06 (3)	5.8 ± 2.18 (3)	19.4 ± 4.9 (13)	−7148.24 ± 74.35 (−7099.57)	948.3 ± 41.51 (841)
	K2ACO	9.1 ± 2.66 (5)	3.5 ± 1.28 (2)	3.1 ± 1.02 (2)	15.7 ± 3.63 (9)	−7124.67 ± 4.61 (−7115.68)	54538.0 ± 2786.64 (52236)
	K2GA	8.23 ± 1.56 (6)	3.75 ± 1.34 (2)	4.67 ± 1.2 (3)	16.9 ± 2.98 (11)	−7130.84 ± 4.36 (−7124.10)	15866.8 ± 2236.93 (12578)
B	ChainACO	0.63 ± 0.48 (1)	27.3 ± 1.79 (22)	4.46 ± 1.66 (2)	32.46 ± 3.0 (25)	−72350.0 ± 110.6 (−72092.47)	638.23 ± 23.39 (707)
	ChainGA	0.43 ± 0.49 (0)	27.86 ± 2.1 (24)	4.4 ± 1.2 (2)	32.7 ± 2.76 (26)	−72405.6 ± 128. (−72214.48)	903.5 ± 32.58 (862)
	K2ACO	0.7 ± 0.45 (1)	24.1 ± 1.3 (22)	2.8 ± 1.4 (2)	27.6 ± 2.76 (24)	−72145.9 ± 33.51 (−72090.68)	40151.5 ± 18517 (39596.0)
	K2GA	0.6 ± 0.48 (1)	24.5 ± 1.68 (22)	2.5 ± 0.92 (2)	27.6 ± 2.49 (24)	−72162.8 ± 56.4 (−72090.68)	12638.3 ± 2038.3 (14693)
Boerlaga	ChainACO	7.8 ± 1.85 (3)	8.47 ± 1.11 (8)	11.7 ± 2.7 (8)	27.9 ± 3.33 (19)	−30600.8 ± 19.9 (−30550.94)	1268.87 ± 30.45 (1185.0)
	ChainGA	9.76 ± 2.36 (9)	8.6 ± 0.71 (9)	13.6 ± 1.89 (10)	31.96 ± 3.4 (28)	−30629.8 ± 22.9 (−30575.01)	1444.93 ± 32.08 (1467)
	K2ACO	6.6 ± 2.416 (4)	8.4 ± 0.79 (9)	8.9 ± 1.51 (9)	23.9 ± 3.36 (22)	−30582.01 ± 17.0 (−30553.92)	64385.7 ± 4271.08 (55406.0)
	K2GA	7.3 ± 2.23 (4)	8.7 ± 0.9 (7)	11.8 ± 1.93 (7)	27.8 ± 4.33 (18)	−30598.61 ± 9.9 (−30579.91)	15139.1 ± 5972.74 (14456)

Table 3.9: EXPERIMENTAL RESULTS(Insurance, Alarm)

Structure	Approach	A	D	I	SHD	Avg.Score	F.E.
Insurance	ChainACO	27.7 ± 2.3 (25)	19.2 ± 1.8 (17)	17.3 ± 2.7 (18)	64.2 ± 4.8 (60)	−42009.57 ± 286. (−41710.11)	1957.9 ± 22.86 (1959.0)
	ChainGA	22.93 ± 4.0 (16)	20.36 ± 3.0 (18)	14.36 ± 2.0 (11)	57.6 ± 8.4 (45)	−42351.1 ± 369.1 (−41472.39)	1719.96 ± 126.76 (1641)
	K2ACO	15.8 ± 0.7 (16)	16.0 ± 1.2 (14)	9.7 ± 1.18 (7)	41.5 ± 2.1 (37)	−41391.1 ± 114.0 (−41311.72)	56582.1 ± 1408.41 (53717)
	K2GA	14.2 ± 3.2 (13)	16.5 ± 1.2 (17)	10.1 ± 1.9 (8)	40.8 ± 5.1 (38)	−41509.95 ± 92.4 (−41333.71)	20271.5 ± 976.28 (19819)
Alarm	ChainACO	23.5 ± 4.99 (13)	3.83 ± 1.0 (2)	12.5 ± 3.5 (7)	39.8 ± 8.2 (22)	−30248.03 ± 353. (−29788.09)	3428.0 ± 81 (3373)
	ChainGA	36.56 ± 6.71 (23)	5.3 ± 1.57 (4)	21.2 ± 3.3 (21)	63.1 ± 9.3 (48)	−30793.42 ± 340. (−30047.64)	2817.63 ± 151.82 (2514)
	K2ACO	24.4 ± 3.32 (23)	2.75 ± 0.7 (2)	18.8 ± 2.8 (10)	45.9 ± 4.3 (35)	−30096.2 ± 102. (−29875.9)	105938. ± 2112.20 (107376.0)
	K2GA	29.6 ± 3.7 (31)	3.53 ± 0.9 (3)	19.6 ± 3.2 (16)	52.8 ± 5.3 (50)	−30196.2 ± 29. (−29817.09)	38827.37 ± 10980.0 (29987)

Chapter 4

Comparative Analysis of Search and Score Metaheuristics using Node Juxtaposition Distributions

In this chapter we focus on further investigating the information gained in previous chapters about the property of problem-dependency in search and score algorithms, the chain model approaches and K2 greedy approaches. All these algorithms search on the space of node orderings to the specific problem. The effectiveness and the efficiency of these approaches are very different connecting each problem. The objective of this analysis is to study the relationship between problem features and algorithm performance, for benchmark problems with known structures.

We will design experiments around the proposed algorithms involving four combinations of search (GA or ACO) with scoring (K2 or Chain) approaches. We relate node juxtaposition distributions over a number of runs to the known problem structure, the algorithm performance and the detailed algorithmic processes. The experimental results explore, to some extent, that both the choice of metaheuristic and the choice of scoring method can significantly affect performance.

Finally, we simply describe the application of ACO based algorithms to learning

BN models of RIG operation in the gulf of Mexico, and also perform analysis on node juxtaposition distributions to investigate the relationship on different approaches. This provides us with some comparison on empirical experiment results and a real problem application to study the approaches proposed. Some results of this chapter have been published in paper [Wu et al., 2010a] [Fournier et al., 2010].

4.1 Node Juxtaposition Distribution

The aim of our experiments is to investigate the behaviour of GA and ACO meta-heuristics searching the space of node orderings using chain and K2 evaluation methods. We know from the last chapter that both chain and K2 evaluation methods evolve and utilize the best node ordering as the input topological ordering (Figure 4.1) for inducing the BN structure through K2 greedy heuristic algorithm. The structure and the quality of the node ordering have the most important effect to the final structure learned. Therefore, we try to explore the relationship between arcs derived from node juxtapositions in the best orderings found and arcs in the original structure by all the approaches.

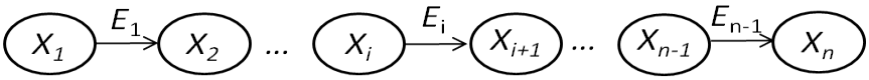


Figure 4.1: Chain structure on best node ordering with arcs

A $n \times n$ matrix $\mathbf{E}_{n\times n} = (e_{ij})_{n\times n}$ for any n-variables node ordering S is defined to indicate the arcs in this structure. Where:

$$e_{ij} = \begin{cases} 1, & \text{if } P_{a_j} = i, (i, j = 1, \cdots, n) \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

to express the situation of where arcs exist within the chain model.

We know, for one specific chain structure model, the number of $e_{ij} = 1$ is $n - 1$, and all these values of 1 lie in different rows and different columns. Table 4.1 is an

example that shows the node juxtaposition distribution of one best node ordering (3-5-8-7-6-4-1-2) in the Asia structure. Here $n = 8$, and the bold number in the first row and the first column represents the node in the Asia network.

Table 4.1: Node Juxtaposition Distribution for one node ordering on Asia

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0
4	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1
6	0	0	0	1	0	0	0	0
7	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	1	0

4.2 Experimental Design

In this section, we design and run a series of experiments to study the proposed task. The framework of the study considered in this chapter is described in Figure 4.2.

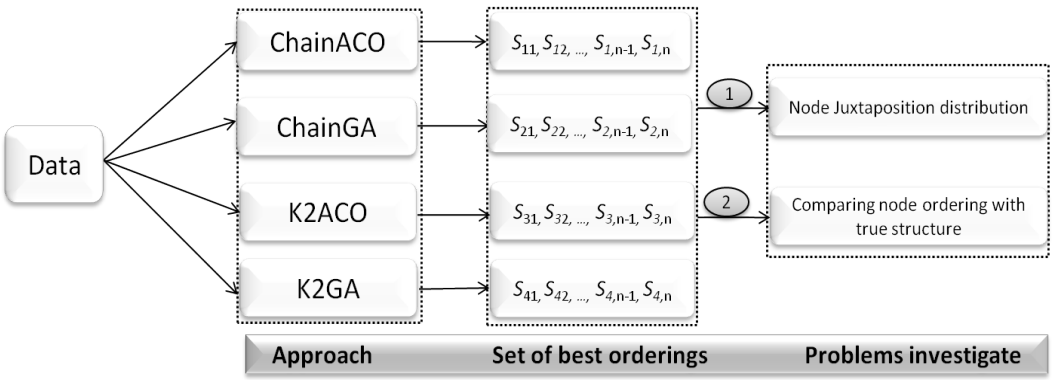


Figure 4.2: Experimental framework with best node orderings

We therefore make runs of each metaheuristic with each evaluation method,

ChainACO, ChainGA, K2ACO and K2GA. Note that in these experiments, for obtaining the best node ordering, we only require to run the first phase of ChainACO and ChainGA algorithms, as the search of ordering space ends at this point - the second phase is deterministic. We are not comparing the computational efficiency or the scores of final networks produced as that has been covered in earlier work. These best node orderings obtained can then be used for further analysis in our experiments. Firstly, they will be applied to learn the node juxtaposition distribution among the set of orderings. Secondly, the arcs that exist in the best node ordering will compare to the true structure to explore the relation of the choice of metaheuristic and the choice of scoring method with the structure learned.

A pseudo code representation of node juxtaposition distribution built on best node orderings in chain and K2 evaluation is shown in Algorithms 14.

Algorithm 14 Pseudo code of node juxtaposition distribution built on best node orderings in chain and K2 evaluation algorithms

1: **Input:** A set of n nodes with database D;
2: **Input:** M=Iteration_ number.
3: **Begin algorithm**
4: Initialize $\mathbf{E}_{n\times n} \leftarrow (0)_{ij}$;
5: **for** $i \leftarrow 1, M$ **do**
6: Implement a proposed approach on D to achieve a best node ordering S_i;
7: Construct matrix $(\mathbf{E}_i)_{n\times n}$ from S_i;
8: $\mathbf{E}_{n\times n} \leftarrow \mathbf{E}_{n\times n} + (\mathbf{E}_i)_{n\times n}$;
9: **end for**
10: **Output:** $\mathbf{E}_{n\times n}$.
11: **End algorithm**

The previous eight well known benchmark problems have been selected in the experiment: Asia, Credit, Tank, Car, Boerlaga, B, Insurance and Alarm. The data size for Asia is $10,000$, for all other problems, the size is $3,000$. In all cases, the scoring metric used to evaluate the node ordering is the CH metric. For ChainACO and ChainGA we carry out 200 experimental runs each. For K2ACO and K2GA we

carry out only 50 runs each due to time complexity. The parameters used for ACO and GA based algorithms in the experiments are the same as those used in previous experiments in chapter 3.

4.3 Results and Discussion

We present the experimental results of four algorithms on all benchmark problems in this section. In each benchmark, we record the corresponding matrix $\mathbf{E}_{n\times n}$ from the four approaches and use the gray scale visual table to intuitively describe the Node Juxtaposition Distributions. We then execute statistical analysis to study these results to explore the potential relations between proposed approaches and problems. We describe these results according to each benchmark problem as follows:

4.3.1 Asia Benchmark Structure

First of all, we consider the Asia structure. Table 4.2a– Table 4.2d presents the distribution of node juxtapositions recorded from the best ordering found in all algorithms on the Asia network. In these tables, the row index indicats the first node in a juxtaposition. The column index indicates the second. The run number of ChainACO and ChainGA is 200, and 50 for K2ACO and K2GA. For example Table 4.2a shows that in 65 of the runs, the node juxtaposition 1-2 appeared 65 times, and the node juxtaposition 2-1 appeared 135 times. This means that, in all runs of ChainACO on Asia, nodes 1 and 2 were juxtaposed in the best ordering found, with a 135:65 preference for node 2 preceding node 1. In all of these cases, the ordering will have been evaluated using a chain structure inserting a directed edge between these nodes. It is not of course necessary that any particular juxtaposition will appear in all experimental runs. Each ordering found will contain $n-1$ juxtapositions where n is the number of nodes. The sum of entries in the ordering distributions table will therefore in general be $r\cdot(n-1)$,where r is the number of runs. In this case, the entries sum to $1400 = 200\times 7$.

Table 4.2: Node Juxtaposition Distribution for 200 runs of ChainACO, ChainGA, and 50 runs of K2ACO, K2GA on Asia Structure

(a) ChainACO

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	65	0	135	0	0	0	0
2	135	0	19	21	9	7	0	0
3	10	76	0	3	12	4	0	4
4	44	0	4	0	0	145	0	0
5	0	9	65	0	0	0	2	31
6	0	3	0	37	0	0	158	2
7	0	0	0	0	0	42	0	158
8	0	2	3	0	155	0	40	0

(b) ChainGA

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	100	6	74	5	2	0	4
2	100	0	17	23	24	10	2	10
3	11	20	0	23	48	22	2	13
4	66	15	24	0	16	22	5	10
5	9	17	38	16	0	8	12	51
6	4	16	19	22	7	0	85	25
7	1	2	1	1	15	105	0	72
8	1	14	15	14	43	21	92	0

(c) K2ACO

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	1	5	0	6	5	17	16
2	0	0	1	0	0	2	4	26
3	18	4	0	4	2	5	14	3
4	6	0	21	0	3	5	8	2
5	3	3	2	32	0	8	2	0
6	5	6	6	4	27	0	0	1
7	13	23	8	1	2	1	0	2
8	2	13	1	5	0	0	2	0

(d) K2GA

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	14	6	5	7	0	7	11
2	0	0	4	4	1	2	1	3
3	4	6	0	6	8	8	9	3
4	17	1	6	0	9	1	6	3
5	6	3	6	5	0	14	14	1
6	7	1	9	14	8	0	8	2
7	6	4	4	5	2	2	0	27
8	7	21	5	8	5	2	2	0

It is noticeable from Table 4.2a– Table 4.2d that the distribution of node juxtapositions are very different within the four approaches. Table 4.2a shows that in ChainACO the distribution of node juxtapositions is concentrated on a relatively s-

mall subset of possible node juxtapositions. This indicates that ChainACO is highly consistent in the node orderings it produces and suggests a strong convergence property of the search. Table 4.2b– Table 4.2d, however, show the distribution of node juxtapositions are scattered over the whole space of the possible node juxtapositions. These clearly reveal that the corresponding approaches have weak convergence property of the search.

In Figure 4.3, we present a visual representation of the node distributions produced by all four algorithms on Asia. Here, the instance counts have been replaced by a normalized grayscale representation running from white (juxtaposition occurs on 0% of runs) through to black (juxtaposition occurs on 100% of runs). The results are arranged gridwise according to the scheme (Chain, K2) × (ACO, GA) to align visual contrast with variation in search and score approaches. It is easy to observe from Asia that there is a marked difference in distribution between ChainACO (top-left) and K2GA (bottom right). ChainACO produces a high contrast image consisting of mostly very dark or very light pixels whereas the K2GA image is much more diffuse. It is hard to visually detect much of a difference in contrast between K2ACO and ChainGA other than that they lie somewhere in between the other two. Moreover the dark areas for ChainACO do not particularly coincide with those for K2GA. However results in the previous section show that each algorithm reliably reproduces the Asia network.

To properly understand the node distributions, we present a quantitative description on node juxtaposition frequencies for all four algorithms on the Asia problems as Box-plot in Figure 4.4. These frequencies come from all the arcs with node juxtaposition, so in the Asia network, the size of these values for each approach is $8 \times 7 = 56$. Here the effect of the ChainACO approach manifests as a low median frequency (which is almost zero) for most possible juxtapositions with three high frequency outliers, which are not included between whiskers. ChainGA approach has the same median score as the two K2 approaches, and also three high frequency outliers. We

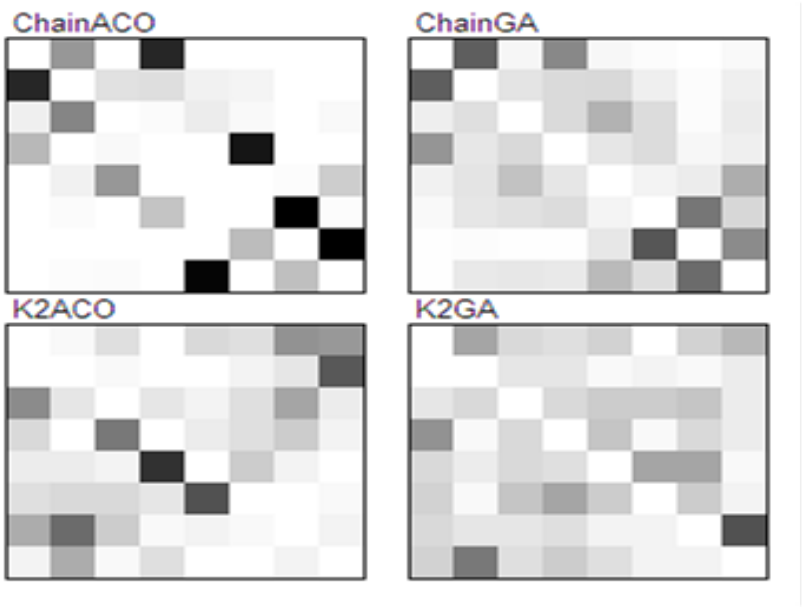


Figure 4.3: Grayscale Grids of the Edges Occurrences in Asia Network within the Four Algorithms

are interested in the outliers, because in this kind of problem more outliers indicate that the corresponding approach produced more arcs which have higher frequencies. Comparing median and outliers on chain approaches and K2 approaches from the table, we can conclude that the former perform better on concentrating than the latter.

Figure 4.5 is a diagram of the known true structures for Asia, annotated with the best orderings found by each approach. Figure 4.5a– Figure 4.5d corresponds to ChainACO, ChainGA, K2ACO and K2GA respectively. For each node juxtaposition occurring in the best ordering that corresponds with an edge in the true structure, an arrow is added in the middle of the edge in the direction of the node juxtaposition. If there is no edge in the true structure corresponding to the node juxtaposition, a dotted arrow is added to the diagram. Solid directed edges with no central arrow therefore represent edges that occur in the true structure but are not represented by node juxtapositions in the best ordering found. We also annotate each node juxtaposition with the overall percentage of runs in which it appeared in the best ordering for that run.

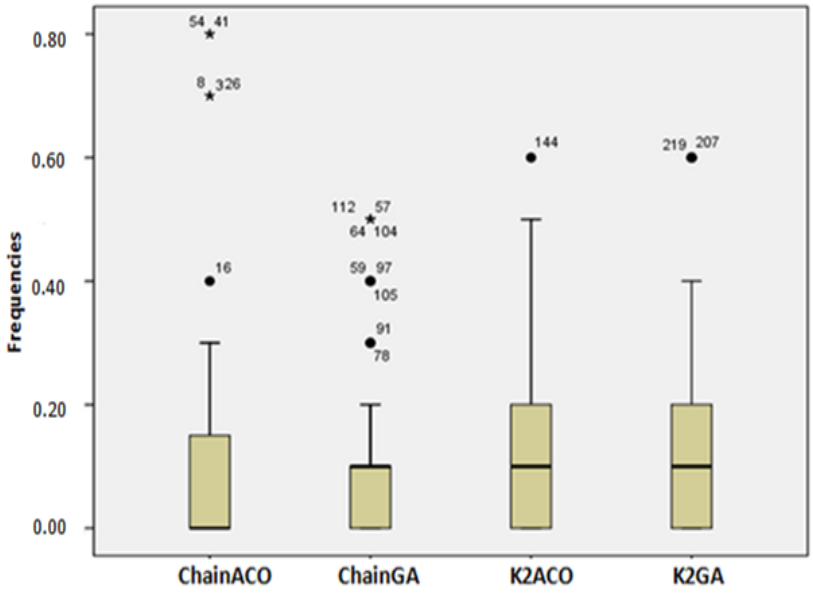


Figure 4.4: Comparison of frequencies of each edges found in Asia network within four approaches

In Figure 4.5a, the best ordering is 2-4-7-6-5-3-0-1. The diagram shows that all but two of the node juxtapositions in the best ordering found by ChainACO for Asia coincide with true arcs. Of these three out of six are correct and three are reversed. The solution adds one node juxtaposition corresponding to a spurious arc and omits two arcs. The percentage of the three correct arcs are 0.185, 0.22 and 0.325. These are relatively high values to ensure that the true structure is learned by K2 in the second phase of the algorithm. In Figure 4.5b, the best node ordering is 2-5-6-7-3-0-1-4. It shows that four node juxtapositions in the best ordering found by ChainGA for Asia are correct arcs comparing to Asia structure, and three added node juxtapositions are produced. In this case, three true arcs are deleted. We can find that the percentage of the four correct arcs are higher than in ChainACO. They are 0.425, 0.36 , 0.33 and 0.50. These results mean it is stable for generating the best node orderings in K2 phase and the true BN structure is hard to produce. We also investigate Figure 4.5c and Figure 4.5d. The best node orderings in these two diagrams are 4-5-2-3-6-0-7-1 and 4-5-6-3-2-0-1-7 respectively. A very clear result is in K2ACO. The number of

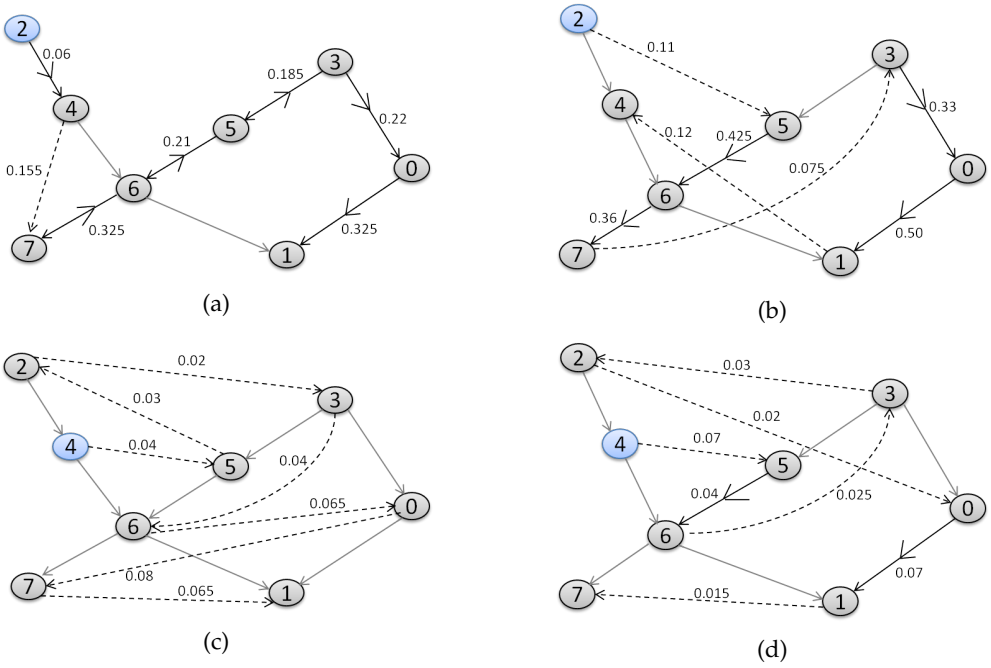


Figure 4.5: Asia Structures annotated by best node orderings on four algorithms

correct node juxtapositions is zero and the deleted arcs is seven. K2GA has two correct arcs produced and five omitted. The nodes corresponding to these deleted arcs in both algorithms are chaotic sequence. These will give K2 a disordered node ordering and make K2 poorly in BN structure learning. The above analysis shows that it is possible to create chains closely aligned to the structure. This explains why ChainACO and ChainGA perform well on Asia. Table 4.3 is the arc statistics on Asia with four approaches.

Table 4.3: Node Juxtaposition Distribution for one node ordering on Asia with four approaches

	Correct	**Reversed**	**Added**	**Omitted**
ChainACO	3	3	1	2
ChainGA	4	0	3	4
K2ACO	0	0	7	8
K2GA	2	0	5	6

4.3.2 Credit Benchmark Structure

From Credit structure onwards, in the following description on other benchmarks we do not intend to display all the data as in Table 4.2a– Table 4.2d. We present the visual table which has equivalent data for all approaches in Figure 4.6. Here, the image of K2ACO looks more marked. There exist 7-8 nodes with heavy color. This means the ordering has concentrated distributions on the ACO metaheuristic with K2 greedy search. From the images, both ChainACO and ChainGA algorithms have similar distributions, but the best node orderings in them are not clear. K2GA, however, is the most uniform on gray distribution. Comparing the experimental results in chapter 3, K2ACO has the best performance on Credit either in fitness score or structure hamming distance. Figure 4.6 can explain how the best node ordering from the K2ACO generated, which is very important for K2 heuristic search algorithms. In spite of failing to produce the best node orderings in K2GA, the approach got a satisfactory result in our previous experiments. Figure 4.7 presents the quantitative

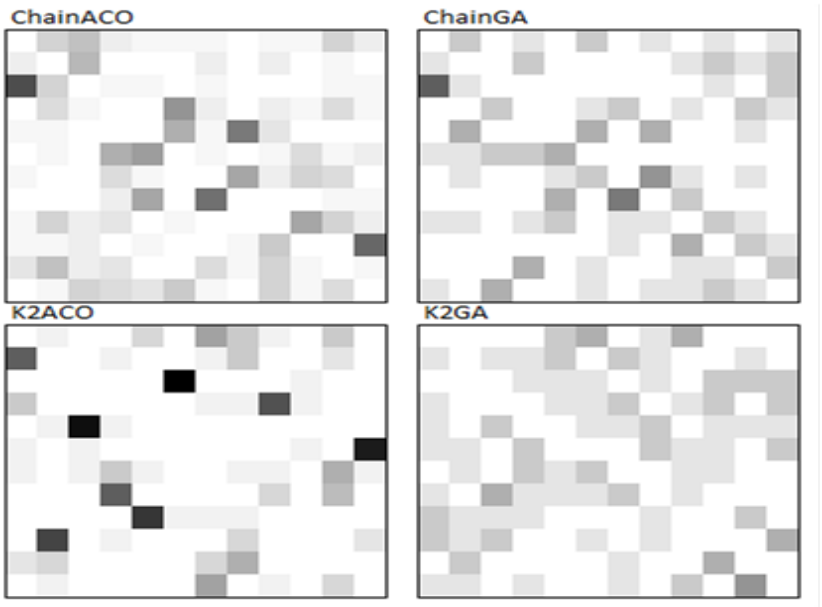


Figure 4.6: Grayscale Grids of the Edges Occurrences in Credit Network within the Four Algorithms

description on node juxtaposition frequencies for all four algorithms on the Credit problem. K2ACO in this case has the most and the furthest outlier compared to other three approaches. This is consistent with the visual table in Figure 4.6. These outliers correspond to the arcs found in chain orderings. Three frequencies of these outliers are very close to 1.00. This can give an absolute result of best arcs both in node ordering and in the final BN structure.

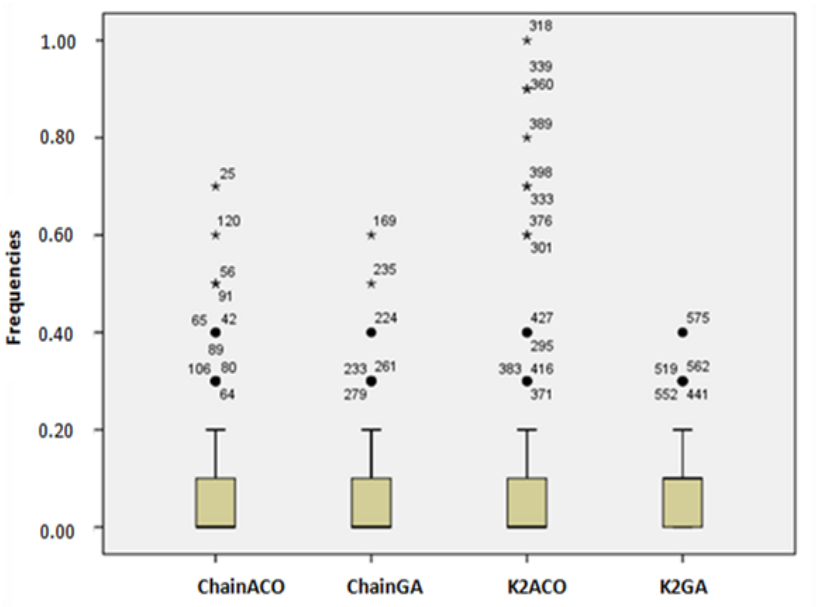


Figure 4.7: Comparison of frequencies of each edges found in Credit network within four approaches

Table 4.4: Node Juxtaposition Distribution for one node ordering on Credit with four approaches

	Correct	**Reversed**	**Added**	**Omitted**
ChainACO	5	2	4	5
ChainGA	2	4	5	6
K2ACO	1	0	10	11
K2GA	2	2	7	8

Figure 4.8 depicts the best node ordering generated in K2ACO compared to the true Credit and Table 4.4 indicates arcs statistics on Credit with the four approaches.

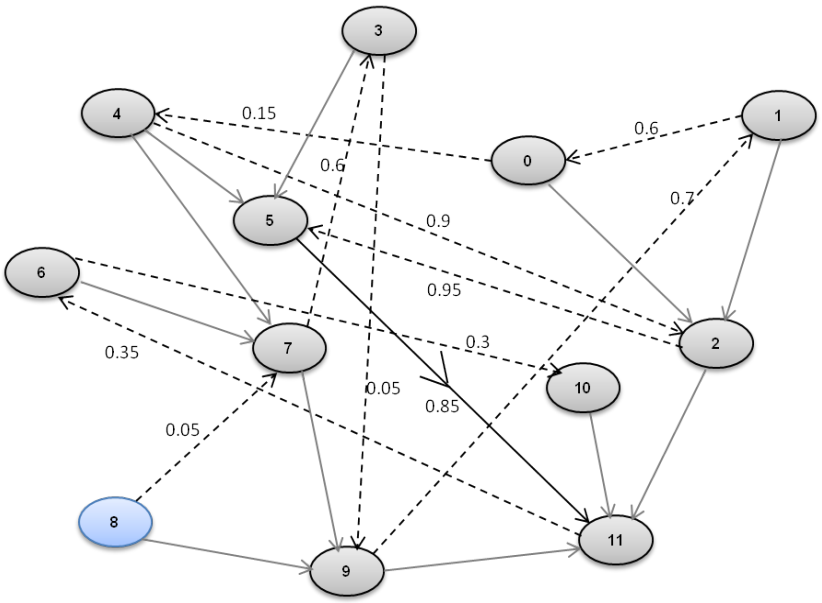


Figure 4.8: Credit- annotated by best K2ACO ordering 8-7-3-9-1-0-4-2-5-11-6-10.

These include the arcs deleted, added and reversed. From Figure 4.8 we can find that only one correct edge can be found by this node ordering. However, from Table 4.4, we know ChainACO achieved five correct edges. These explore the complexity of K2 based search and score algorithms. The node ordering which has more true edges is not the topological node ordering for K2 searching.

4.3.3 Tank Benchmark Structure

The visualization is very clear in Figure 4.9 on Tank structure with four approaches. This is consistent to the performance of K2ACO and K2GA in our previous experiments. We notice the best node orderings in these two approaches are not fixed in the same nodes. This is because the best node orderings of the structure are not unique.

Figure 4.10 presents the quantitative description on node juxtaposition frequencies for all four algorithms on the Tank problem. ChainACO in this problem achieved the most and the farthest outlier compared to the other three approaches. However, the median is nearly zero. K2GA also has a similar median but the number of outliers is

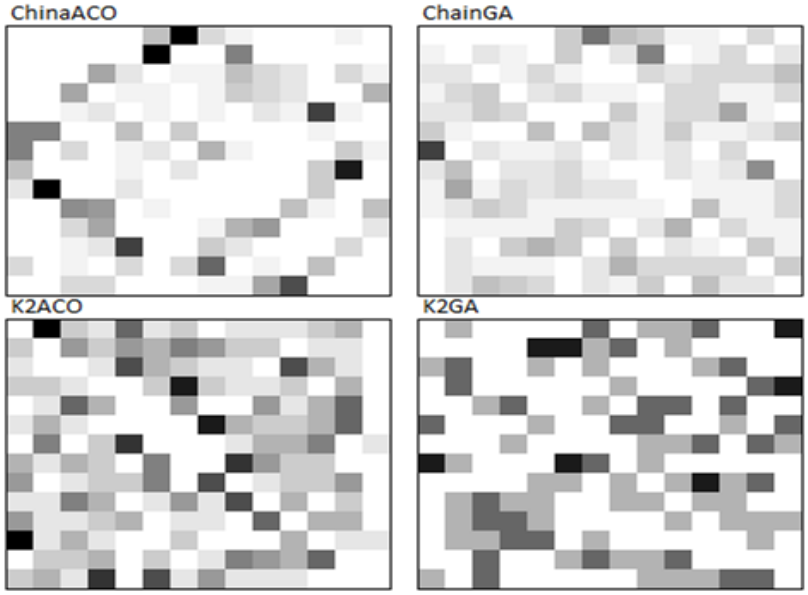


Figure 4.9: Grayscale Grids of the Edges Occurrences in Tank Network within the Four Algorithms

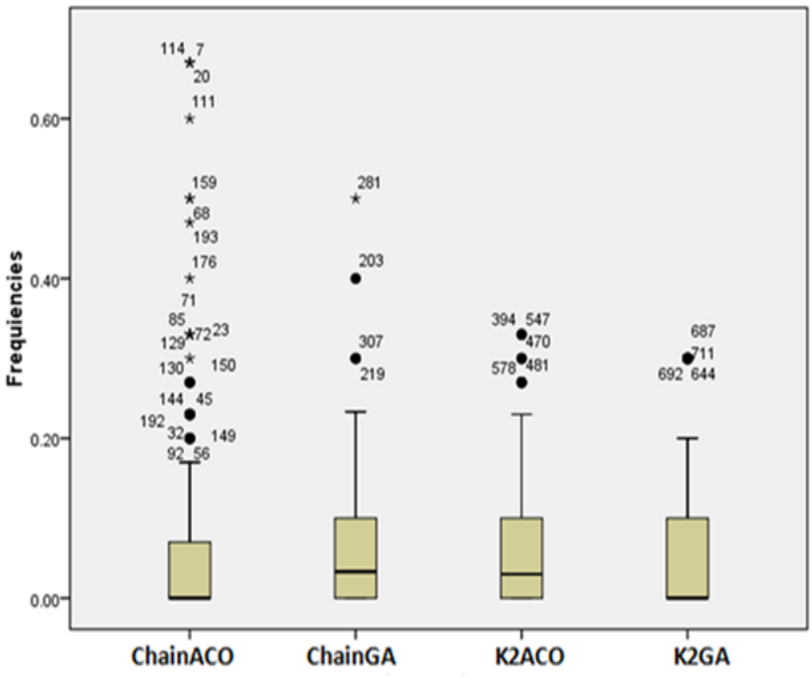


Figure 4.10: Comparison of frequencies of each edges found in Tank network within four approaches

only one. Both ChainGA and K2ACO have higher medians and a smaller number of outliers. Notably, the ChainACO approach got a large amount of outliers relating to specific edges in the best node ordering, but the performance of ChainACO was poor from previous experiments. To explain this performance, we draw the best edges on ChainACO from Figure 4.9. These edges are $0 \rightarrow 6$, $1 \rightarrow 5$, $1 \rightarrow 8$, $3 \rightarrow 2$, $7 \rightarrow 12$, $8 \rightarrow 1$, $11 \rightarrow 4$,$12 \rightarrow 7$, $13 \rightarrow 9$ and $13 \rightarrow 12$. Four of these edges are the correct arcs in the true structure. They are $0 \rightarrow 6$, $1 \rightarrow 5$, $1 \rightarrow 8$ and $7 \rightarrow 12$. Four of them are added to the true structure and two are reversed ($8 \rightarrow 1$ and $12 \rightarrow 7$). Table 4.5 indicates arcs statistics on Tank for the four approaches. We know, in the Tank problem, for a node ordering, the number of arcs within variables is 13. This table shows the difference in number of correct, reversed and added arcs for each approach is not significant within four approaches.

Table 4.5: Node Juxtaposition Distribution for one node ordering on Tank with four approaches

	Correct	**Reversed**	**Added**	**Omitted**
ChainACO	3	1	8	16
ChainGA	1	1	11	18
K2ACO	2	2	9	16
K2GA	3	1	9	16

4.3.4 Car Benchmark Structure

We present the visual table which has equivalent data for all approaches in Figure 4.11. Here there are more possible node juxtapositions and the visual contrast is more marked. In order from highest to lowest visual contrast, the images are ordered ChainACO, K2ACO, ChainGA and K2GA. This ordering is consistent with a hypothesis that both the Chain scoring approach and the ACO metaheuristic result in more concentrated distributions than the K2 scoring approach and the GA metaheuristic respectively.

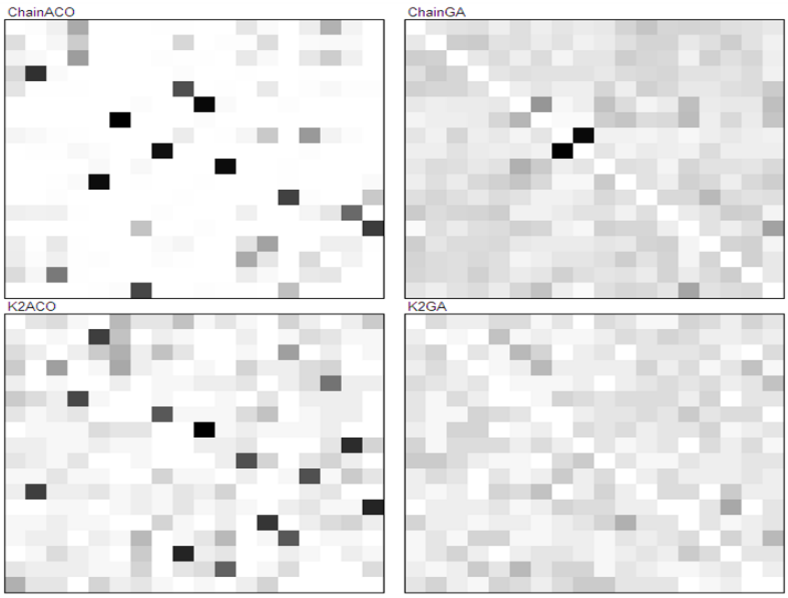


Figure 4.11: Grayscale Grids of the Edges Occurrences in Car Network within the Four Algorithms

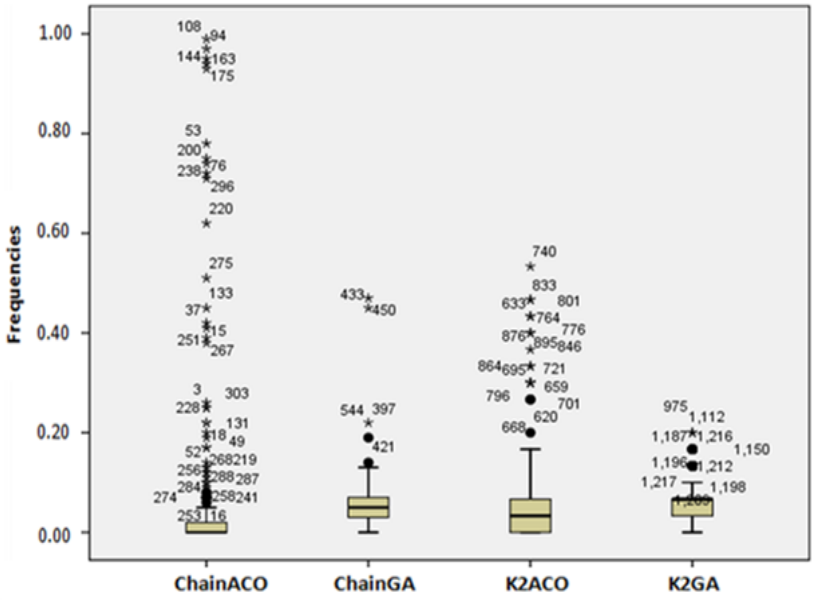


Figure 4.12: Comparison of frequencies of each edges found in Car network within four approaches

The statistical analysis with Box-plot are shown in Figure 4.12. In this case, the data size in each approach is $18*17 = 306$. It is noticeable that ChainACO achieved the

lowest median with most outliers while K2ACO is the second. K2 based algorithms both have a higher median and a small number of outliers. This means that all the possible node juxtapositions distributions are relatively even. The possibility to converge on a node ordering becomes very low in this condition. The box-plot results give a quantitative description of each approach and verify the performance of the chain based algorithm with ACO.

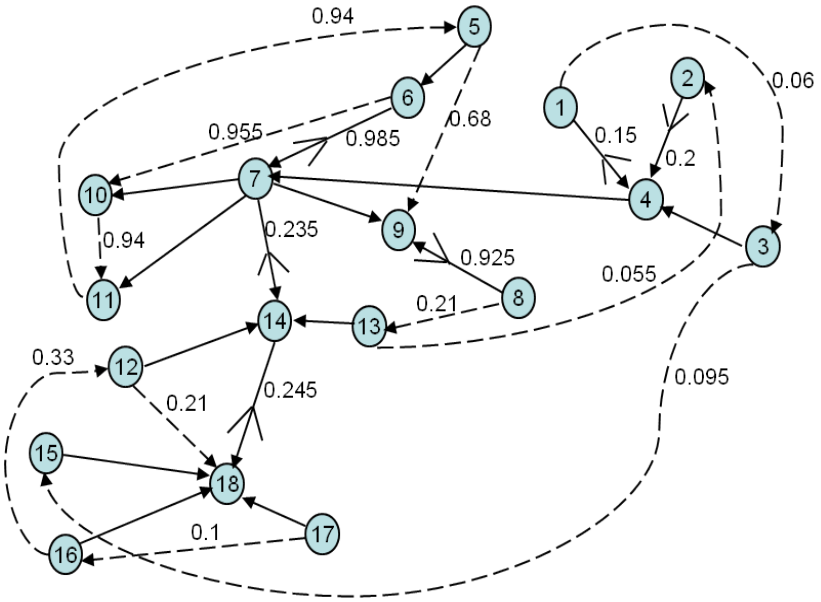


Figure 4.13: Car - annotated by best ChainACO ordering 17-16-12-18-14-7-6-10-11-5-9-8-13-2-4-1-3-15.

Figure 4.13 shows that only six of the seventeen node juxtapositions in the best ordering found by ChainACO for Car coincide with true arcs. Of these only one out of six is correct and five are reversed. The solution adds eleven node juxtapositions corresponding to spurious arcs and omits ten arcs. It is noticeable that the true Car structure contains nodes such as 4, 7, 14 and 18, each of which is a hub for a cluster of tightly-bound nodes. The binding between these clusters is loose. Such a topology is not amenable to the construction of chains where many node juxtapositions correspond to a true arc from independent parents. For example, only one node can be positioned before node 18, which immediately excludes at least three true arcs

in any ordering. Therefore in this case, the chain approach finds spurious arcs that can coexist as node juxtapositions in a single ordering and give a better score than orderings that include correct node juxtapositions. This inherent difficulty in aligning chains of node juxtapositions with the true structure explains why ChainACO and ChainGA perform poorly on Car. Table 4.6 indicates the arcs statistics for all four approaches with the Car structure.

Table 4.6: Node Juxtaposition Distribution for one node ordering on Car

	Correct	**Reversed**	**Added**	**Omitted**
ChainACO	1	5	12	12
ChainGA	1	4	13	13
K2ACO	0	0	18	18
K2GA	1	0	17	17

4.3.5 B Benchmark Structure

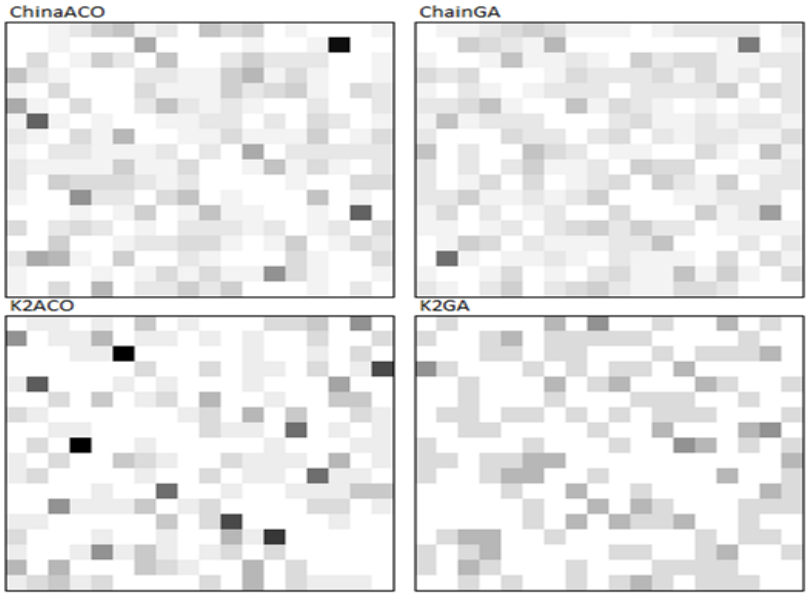


Figure 4.14: Grayscale Grids of the Edges Occurrences in B Network within the Four Algorithms

Figure 4.14 gives the visual table with equivalent data for all approaches. In

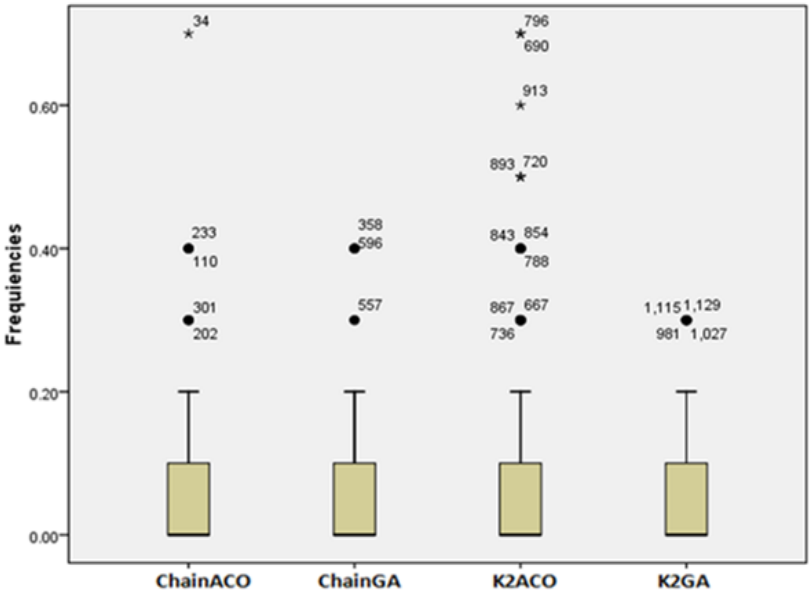


Figure 4.15: Comparison of frequencies of each edges found in B-network within four approaches

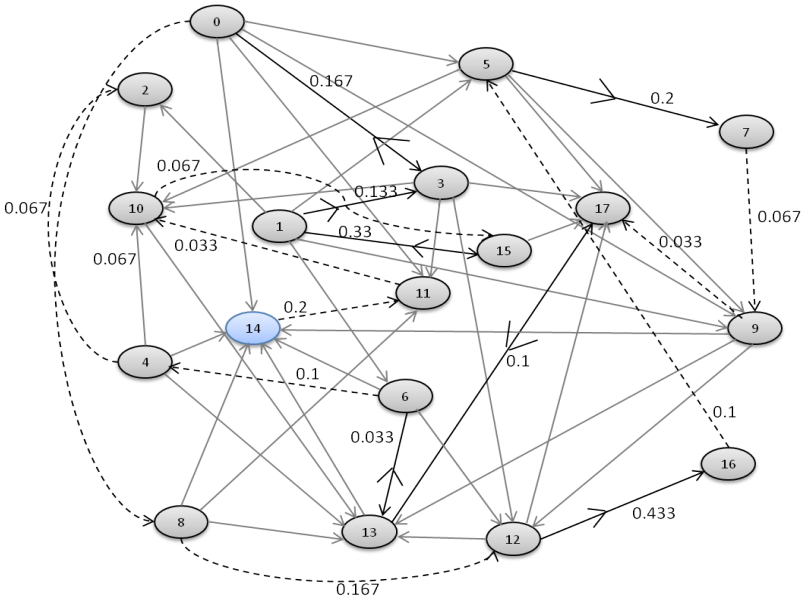


Figure 4.16: B- annotated by best BK2ACO ordering.

this table, K2ACO is the highest visual contrast. The other three approaches have a similar lower visual contrast. This matches the performance of K2ACO on B structure

in previous experiments. The best node ordering produced in K2ACO leads to the best BN structure found in K2 algorithm. In Figure 4.15, the quantitative description on node juxtaposition frequencies for all four algorithms on the B problem shows that K2ACO got the most and farthest outliers compared to the other three approaches. In this problem, all approaches have the same median, all close to zero. Figure 4.16 indicates the structure learned in comparison to the true B problem by K2ACO. There are seven correct arcs found in this 17 nodes long best node ordering. In spite of this K2ACO get the best node ordering, the values of frequency to correct edges are relatively low. The highest one is 0.433. Most of these are less than 0.10.

4.3.6 Boerlaga Benchmark Structure

We know in Boerlaga structure, the ChainACO has the best performance in all four approaches. The best orderings to four approaches are reflected in the visual table in Figure 4.17. This is a very significant result of the comparison of the edges corresponding to the black points in these tables. The ChainACO here is the highest contrast in comparison to the other three. With the same display in other structures discussed above. This ordering can show that both the chain scoring approach and the ACO metaheuristic result in more concentrated distributions than the K2 scoring approach and the GA metaheuristic respectively.

The Box-plot graph in Figure 4.18 is significantly different to the above structures. The median in ChainACO is zero, and the width of box is also zero. This means that except for the 8 outliers, all other values of probability, including the smallest and median, are all very small. We investigate these 8 outliers. The edges they represent are $2 \rightarrow 18$, $4 \rightarrow 10$, $8 \rightarrow 4$, $9 \rightarrow 11$, $12 \rightarrow 13$, $13 \rightarrow 12$, $17 \rightarrow 22$, $18 \rightarrow 2$. Out of these edges, 5 is the correct arcs, 3 is the reversed to the true structure. It shows the ability of chain structure with ACO metaheuristics in searching the best node ordering. Three other approaches also have the same median, but the number of outliers are lesser and lower possibility values. Figure 4.19 shows through the chain structure model

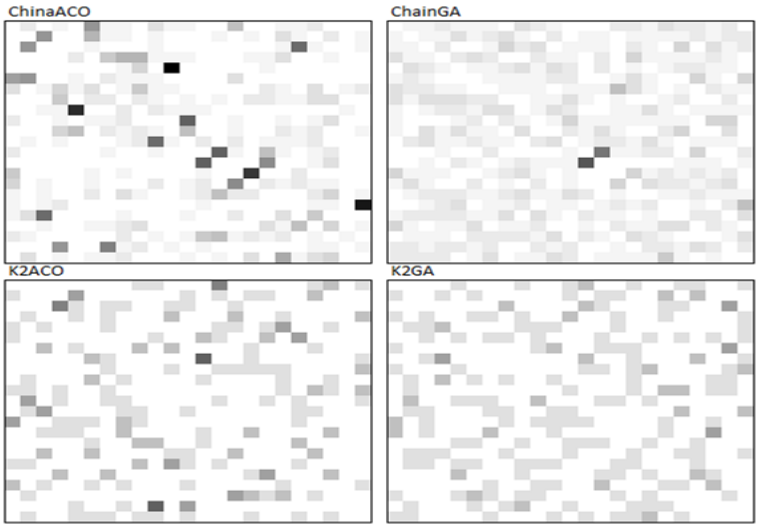


Figure 4.17: Grayscale Grids of the Edges Occurrences in Boerlaga Network within the Four Algorithms

produced, the ChainACO achieved 9 correct edges and 5 reversed edges. The values of probability of these edges are larger; the biggest one is 0.8. These relatively large number of edges of topological node orderings will give the following K2 algorithm very valuable information to generate the BN structure.

4.3.7 Insurance and Alarm Benchmark Structures

Both Insurance and Alarm have a larger number of variables. The equivalent diagrams for them are shown in Figure 4.20 and Figure 4.21. In both of them, as they have many more possible node juxtapositions, the diagram has a finer granularity but the same effects are observable. ACO based algorithms produce contrasting grayscale. Although other approaches have some marked points, the number of these points is clearly less than in ChainACO approach.

We present node juxtaposition frequencies for all four algorithms for the Insurance and Alarm problems as Box plots in Figure 4.22 and Figure 4.23. These essentially show the same information. Here in both Figures, the effect of the ChainACO approach manifests as a low median frequency for most possible juxtapositions with a

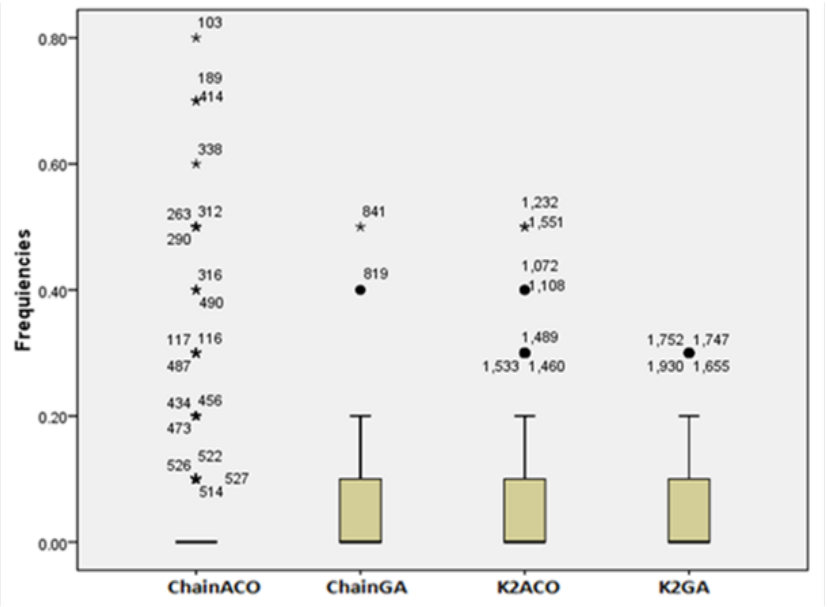


Figure 4.18: Comparison of frequencies of each edges found in Boerlaga-network within four approaches

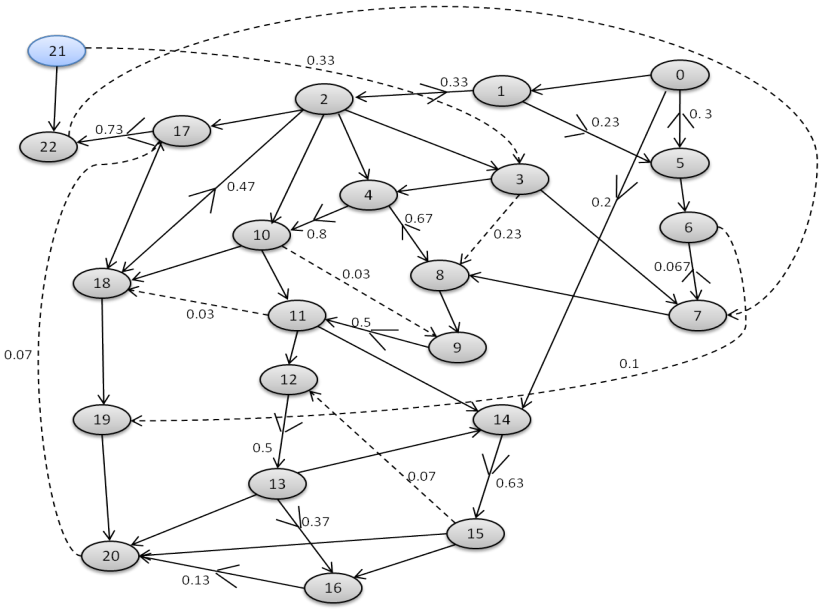


Figure 4.19: Boerlaga- annotated by best ChainACO ordering 21-3-8-4-10-9-11-18-2-1-5-0-14-15-12-13-16-20-17-22-7-6-19.

large number of high frequency outliers. Conversely, for K2 approaches, and K2GA in particular, there is a higher median node distribution frequency and a small dis-

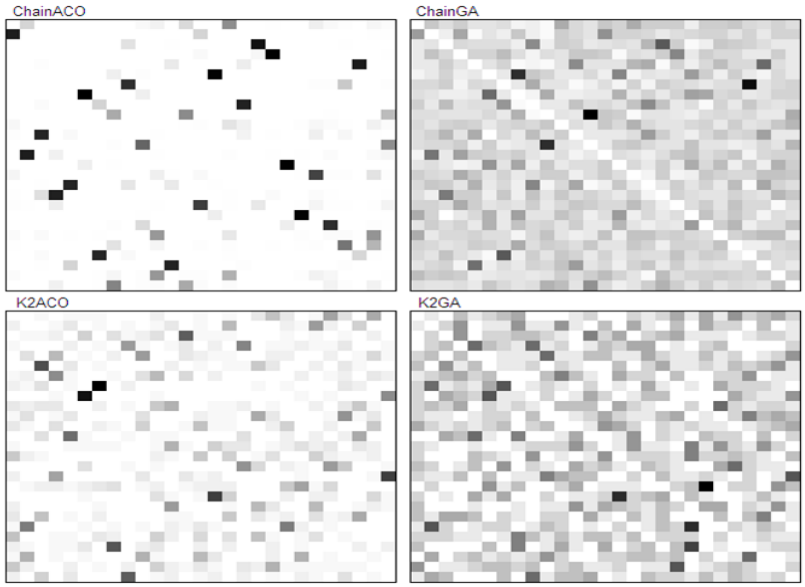


Figure 4.20: Grayscale Grids of the Edges Occurrences in Insurance Network within the Four Algorithms

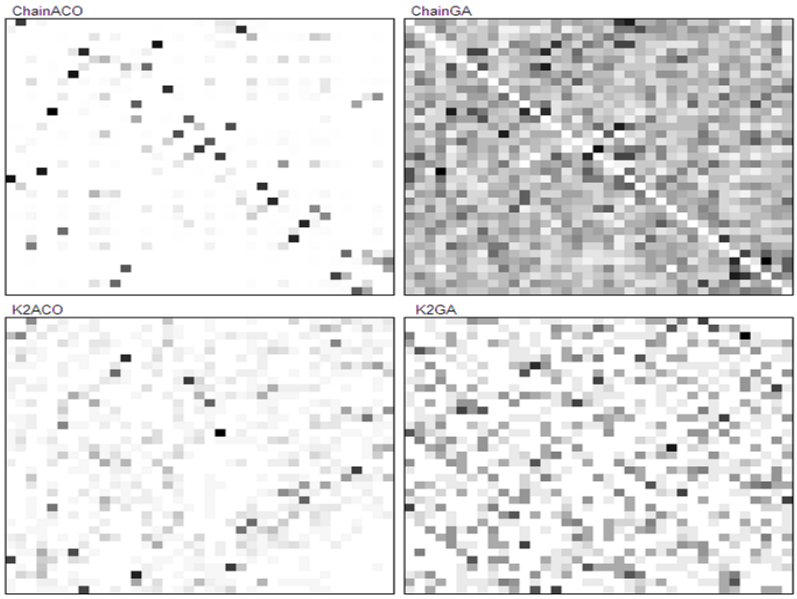


Figure 4.21: Grayscale Grids of the Edges Occurrences in Alarm Network within the Four Algorithms

tribution of frequencies in the interquartile range, corresponding to the more diffuse visual pattern observed earlier. However, we can notice that in Alarm, the highest

frequency value in ChainACO is nearly 1.0. This is a very big number in distribution. It means, nearly every run, this fixed edge definitely emerged. The highest frequency in Insurance is only 0.2 in 200 runs. This can explain why both in Insurance and Alarm, ChainACO got the best node orderings, but compared to the BN structure learning in K2 stage, it performs worse in Insurance, better in Alarm. GA based algorithms in Alarm acquired a very small number of outliers. The frequency of these points are also very low.

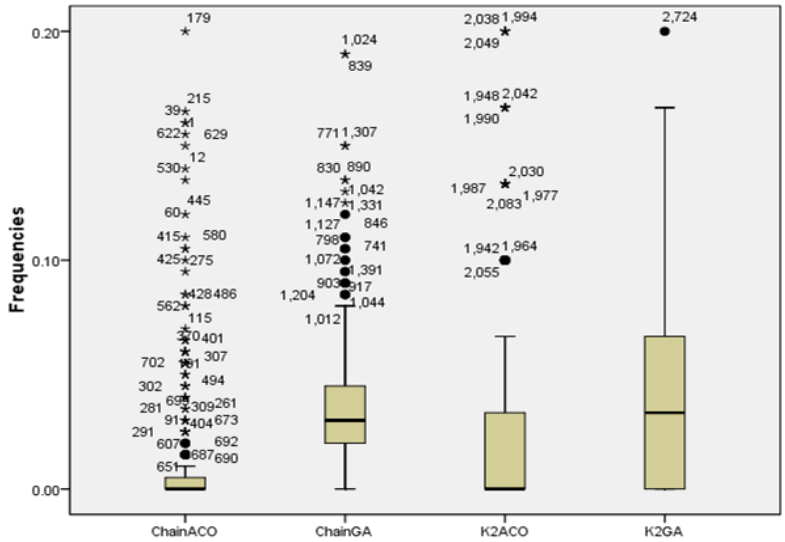


Figure 4.22: Comparison of Frequencies of Each Edges Found in Insurance Network within four Algorithms.

Overall, there appears to be a small reduction in variability of the final ordering produced deriving from the use of ACO rather than GA, but the dominant difference in behavior derives from the choice of scoring approach. We observe that the chain scoring approach concentrates the search on a smaller set of node juxtapositions, and hence node orderings, than the K2 scoring approach does. This is because, for any particular ordering, chain only inserts edges between juxtaposed nodes whereas K2 may insert an edge between any two nodes. Thus it is possible to discover valuable interactions from a wider range of orderings with K2 than with chain. Conversely, it

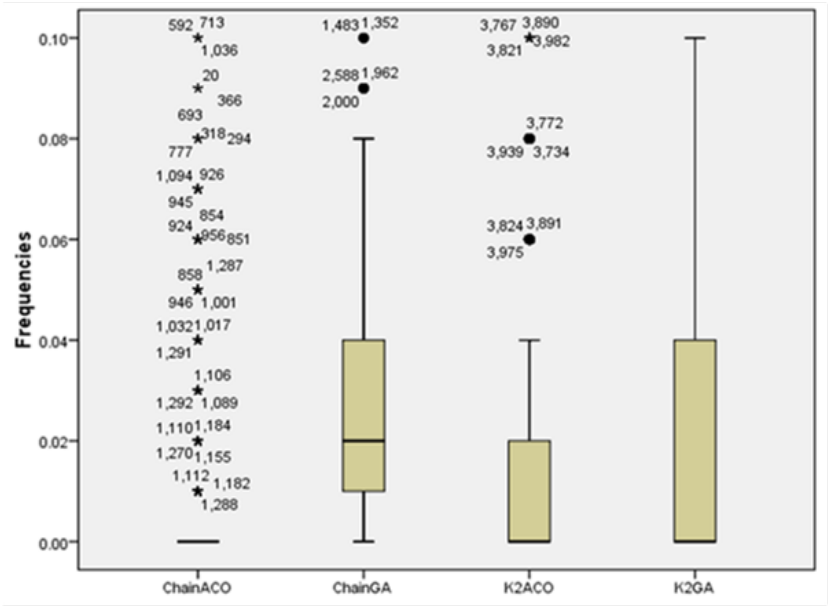


Figure 4.23: Comparison of Frequencies of Each Edges Found in Alarm Networks within four Algorithms.

takes longer to evaluate orderings with K2 because a large number of possible edges have to be considered in turn for each ordering. Therefore, the relative merits of Chain and K2 for any particular problem lie in how amenable the dependencies in the data are to discovery using the Chain approach.

4.4 Search and Score Algorithms to Learning Evolved Bayesian Network Models of Rig Operations in the Gulf of Mexico

The operation of drilling rigs is highly expensive [ODS-Petrodata,]. It is therefore important to be able to identify and analyze variables affecting rig operations. Fournier et al. [Fournier et al., 2010] have applied the ACO based approaches we proposed with other search and score algorithms to study BN structure model with the data achieved from the Gulf of Mexico used by ODS-Petrodata Ltd. [ODS-Petrodata,]. The research is interested in exploring the relative performances of different search and score heuristics and consider trade-offs between best network score and computation

time from an industry standpoint. We describe the main results below:

4.4.1 Performance of the Algorithms

The mean structure scores (log format of CH score) and the significance tests on all pairs of means for each algorithm are presented in Table 4.7 and Table 4.8 respectively. Mean run times for each algorithm are presented in Table 4.9.

Through comparing the results in Table 4.7, K2GA produces on average significantly better scoring structures than all of the other algorithms. Although significantly different, we can see that the results from K2ACO and ChainACO are much closer to K2GA than ChainGA, and they also benefit from a smaller standard deviation, showing their stability compared to ChainGA's. Table 4.8 confirms that K2GA, K2ACO and ChainACO are much closer to each other, in term of scoring, than ChainGA. The difference in the mean score of all pairs formed from K2GA, K2ACO and ChainACO is less than 1000 while all pairs involving ChainGA have a difference in mean score around 7000. It is to be noted that, as discussed in [Fournier et al., 2010] and [Kabli et al., 2007], the performance of ChainGA relating to K2GA appears to be highly problem-dependent. In terms of mean run times, from Table 4.9, the ChainGA runtime is about a quarter of the K2GA runtime. K2ACO requires a significantly different but closer time to ChainGA. However, ChainACO completes with run times divided by a factor of 10 when compared to K2ACO or ChainGA and by a factor of 40 when compared to K2GA.

Table 4.7: Means and Standard Deviations of Best Individuals CH Scores

	N	**Mean Score**	**Standard Deviation**
K2GA	45	-56197.44	205.2
ChainGA	45	-66434.34	1237.7
K2ACO	41	-56265.43	297.8
ChainACO	40	-56556.41	254.7

The score of ACO-based algorithms being much closer to K2GA than ChainGA,

Table 4.8: Paired *t*-test of best Individuals CH score across all runs

Pair	N	Paired Mean Score	Paired Standard Deviation	P
K2GA-ChainGA	43	7721.6793	954.3604	< 0.0005
K2ACO-ChainACO	40	308.3900	109.7554	< 0.0005
K2GA-K2ACO	41	410.3674	298.7311	< 0.0005
ChainGA-ChainACO	40	-6885.6652	653.7462	< 0.0005
K2GA-ChainACO	40	694.2759	234.9186	< 0.0005
ChainGA-K2ACO	41	-7220.7185	658.1467	< 0.0005

Table 4.9: Time Statistics per run over all runs

	Average	Standard Deviation
K2GA	42h28m	5h9m
ChainGA	11h01m	01h11min
K2ACO	11h50m	0h41m
ChainACO	1h39m	0h5m

the loss of quality compared to the gain of time is statistically significant, but smaller than the loss of gain obtained by ChainGA. The long computation times required on this problem are in a large part due to the number of distinct values taken by many of the variables. Considering the vast amount of data available to us, K2GA might not be feasible in some cases for building larger models whereas ChainACO will allow us to build a model taking into account more of the data available.

4.4.2 Expert Evaluation of the Model

Figure 4.24 illustrates the best BN models learned from data using the algorithms. In this figure, as in [Fournier et al., 2010] we can see some matching relationships formed in the models created by K2GA, ChainGA, K2ACO and ChainACO.

We can find that all the algorithms discovered interactions between Rig Capabilities, Rig Types and Water Depth nodes. Experts highlighted that those are linked because specific rig types typically operate at a specific range of water depth. Another group of interactions is identifiable between Well Result, Well Status and Well

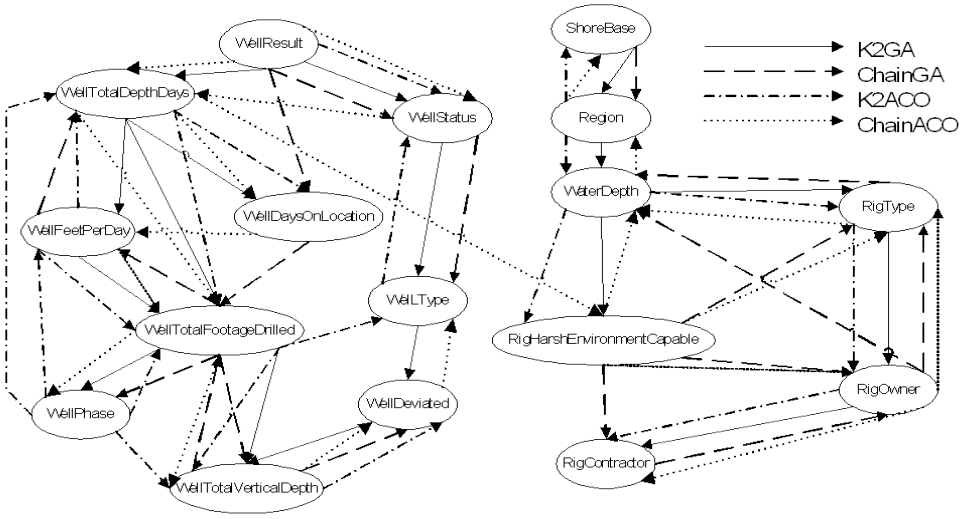


Figure 4.24: Superposition of networks

Type. Only ChainACO omitted that link; however, as the search is non-deterministic, another run of ChainACO might find it.

4.4.3 Analysis of the Node Juxtapositions

Figure 4.25 represents the occurrences of node juxtapositions as a grayscale grid. The vertical axis represents the first node; the horizontal axis represents the second node. The shade is darker proportionally to the number of occurrences of node juxtapositions within the best ordering of each run for all four algorithms.

Figure 4.25 shows that K2GA explores the search space more broadly without focusing on any specific link. This explains why it finds better solutions, but this is an expensive behaviour. ChainGA seems to focus the exploration on the most likely chains. However, its score is lower than that of other algorithms. K2ACO reduces even further the thoroughness of the search but performs better than ChainGA. ACO based algorithms on this problem, seem to be more stable than ChainGA, and also focus more quickly onto the most important part of the ordering compared to GA-based algorithms. ChainACO clearly focuses on some important nodes, converging quickly and consistently towards a good solution.

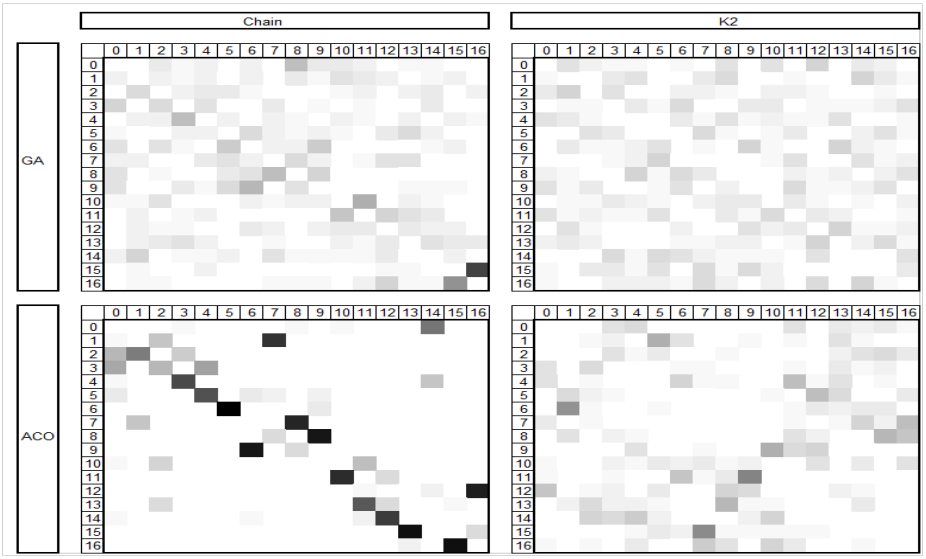


Figure 4.25: Grayscale representation of node juxtapositions

4.5 Summary

In this chapter we have conducted experiments to investigate the behaviour of GA and ACO metaheuristics searching the space of node orderings using the Chain and K2 evaluation methods. We have explained problem-dependent performance trade-offs between cost and structure quality in terms of the relationship between the Chain scoring mechanism, which relies on ordering node juxtapositions, and true arcs in the original structure. In all problems investigated, the Chain scoring approach focused the search on a narrower range of orderings than did the K2 scoring approach. A lesser effect was also observed in that ACO-based methods appeared to concentrate the search more than GA-based methods.

Through the above discussion on all benchmarks, there exist different effects between Chain scoring function and K2 scoring on producing the best node orderings. In most of these benchmarks, for example in Asia, Tank, Car, Boerlaga, Insurance and Alarm, the ChainACO got best node orderings, the distribution of frequency are more dispersed in these benchmarks. However, ChainACO performs poorly in Tank and Insurance. One possible reason is, to some extent, this kind of best node ordering can

not be guaranteed to be the topological node orderings for K2 algorithm to search the best BN structure. In some case, the chain model of best node ordering send insufficient nodes relationships between variables to the K2 greedy search algorithm. This leads to the poor BN structure.

A generalization of the Chain scoring approach that could detect shorter series of well-aligned node juxtapositions combined with a coarse-grain version of the K2 algorithm could potentially assemble high quality structures at reduced cost, even in situations where a single ordering would not admit a set of well-aligned arcs. ACO is a promising approach for more generalized construction approaches.

The comparative results of both GA and ACO algorithm based on node orderings with different approaches to explore and assess methods for the discovery of BN from rig operations data in this chapter indicate the variances among these algorithms. From the perspective of industry experts, the four algorithms have found credible network structures. Comparison of node juxtapositions in the best orderings showed that, for different reasons, the choice of the Chain function as a scoring mechanism or ACO as a metaheuristic, tended to focus the search on a narrower set of orderings. This is consistent with the conclusion drawn from the previous benchmark problems. This proved beneficial for this problem but may not do so in general. However, the computational effort required for ChainACO is a fraction of the effort needed for K2GA.

Chapter 5

Fitness Landscape Analysis of Bayesian Network Structure Learning

We have seen over the previous chapters that search-and-score algorithms using nature-inspired metaheuristics are an important strand of BN structure learning research. However, performance is variable and strongly problem-dependent both in the effectiveness and the efficiency of the approaches used. Both the choice of metaheuristic and the choice of scoring method can significantly affect performance. In all structures, the chain based algorithms are always more computational efficient, but with a penalty on the success in retrieving structure. The results also indicate, in some situation, that the chain scoring approach is likely to be unsuccessful in producing high quality structures (for example, in Credit, Tank, B and Insurance structures, there are significant differences between the chain based approaches and the K2 based approaches to score values), and so the relative benefit of reduced computational time is lost.

In this chapter we mainly focus on studying the performance of the two score functions in search-and-score approaches. We aim to understand to which degree the

choice of different fitness functions affect the effectiveness of the BN structure learning algorithms. The main contribution is to show that landscape analysis around multiple optima (topological orderings) relates structural properties of particular problems to the relative performances of algorithms using chain and CH scoring. We conduct fitness landscape analysis to explain previously observed performance differences between particular search-and-score algorithms. By analyzing the characteristics of each fitness landscape over a range of benchmark structures, we will explore existing empirical performance comparisons with further potential for understanding the relative difficulty of benchmark problems and the robustness of particular algorithms.

We introduce fitness landscape analysis to explain empirically-observed performance differences between particular proposed search-and-score algorithms. We define random walk and fitness functions in our specific problem. We investigate the average landscape discovered by random walks around optimal points in the space of BN node orderings. Differences in algorithm performance are explained in terms of these landscapes, which in turn are related to properties of the BN structures. These initial findings suggest that fitness landscape analysis is a promising approach for explaining existing empirical performance comparisons with further potential for understanding the relative difficulty of benchmark problems and the robustness of particular algorithms. In the next chapter, we will use the conclusions achieved from fitness landscape analysis for determining how important selecting chain scoring or CH scoring is to a particular structure.

5.1 Fitness Landscape Analysis

5.1.1 Definition of Fitness Landscape Analysis

Fitness landscape analysis is a useful tool for representation analysis, either in empirical methods or theory-based analysis [Manderick et al., 1991] [Rothlauf, 2006]. It focuses on investigating the structure from the search space, to visualize the relation-

ship between search space and the solution space in evolutionary algorithm [Wright, 1932]. The search space is regarded as a landscape. An algorithm is seen as a navigating search on the landscape in order to find the highest peak. The points which the algorithm finds represents the dynamics of this algorithm, which is used to show how the algorithm performs its search in the space and how the difficulty of the corresponding landscape. Therefore the technique of fitness landscape analysis can help to analyze and understand the performance of specific algorithms to search problems. It also can provides information for designing an evolutionary algorithm and predicting the performance of the algorithm.

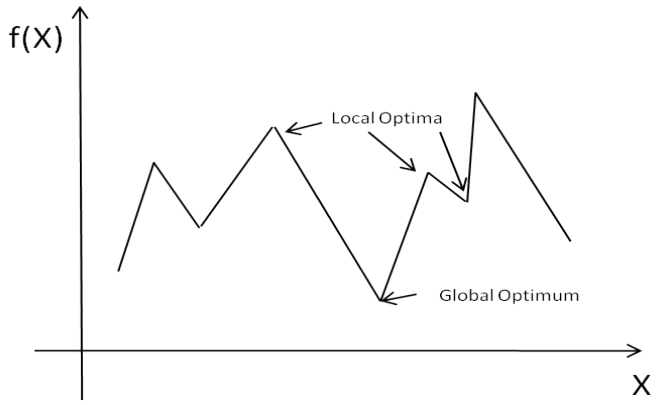


Figure 5.1: Fitness landscape

To construct the fitness landscape for a specific problem, it requires that the global optimum or a very near-global optimum solution is known. It also needs to define a search space first. Every point in the search space has a fitness value assigned. Then some standard mutation operations (random walk) are used to generate a continuous fitness distribution. Typical mutation operations include: bitflip mutation, integer mutation and uniform mutation. The distance between the known optimal solution and the candidate solutions increases with the number of mutation operations increasing.

5.1.2 Measure for Fitness Landscape Analysis

There are several properties for a fitness landscape. They are very important factors to evaluate the performance of corresponding algorithms. For example, the distribution of the fitness function, the number of local optima, landscape ruggedness and the correlation between neighboring points in the search space [Hordijk, 1996] [Stadler, 1995] [Merz and Freisleben, 2000] [Slaný and Sekanina, 2007]. There are several statistical measures for evaluating these properties. For example, if the landscape has several peaks, and if it lacks correlation from two neighbor points, it is said to be rugged. We describe two main related statistical methods proposed to measure the properties of a fitness landscape [Jones and Forrest, 1995] [Hordijk, 1996].

5.1.2.1 Fitness-distance Correlation (FDC).

We suppose $\hat{s}$ is the global optimal solution for a given problem, and s is a local minimum solution. For each local optimum solution s_i, the distance of s_i to the nearest global optimum is defined as d_i. FDC measures the degree to which the distance from s_i to $\hat{s}$ is a predictor of the fitness s. The correlation coefficient of FDC value is computed as:

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})}{S_f S_d} \tag{5.1}$$

$\bar{f}$, $\bar{d}$, S_f and S_d are the mean and the standard deviations of F and D. $F = \{f_1, f_2, \ldots, f_n\}$ is a set of fitness values and $D = \{d_1, d_2, \ldots, d_n\}$ is the corresponding distance to the global optimal. In [Jones and Forrest, 1995] a classification of problem difficulty based on FDC coefficients is suggested. When $r = -1.0$, it represents that the fitness and the distance to the optimum are perfectly related. When $r = 1.0$, it represents the fitness and the distance to the optimum are not related at all. We also know, $r \geq 0.15$ indicates that the fitness tends to increase with distance from the global optimum, $-0.15 < r < 0.15$ indicates there is little correlation between fitness

and distance from optimum, $r \leq -0.15$ indicates that the fitness tends to increase as the global optimum is approached.

5.1.2.2 Landscape Correlation Length (LCL).

A landscape with a few local peaks and small fitness differences between neighboring points in called smooth. A landscape with a lot of local peaks and large fitness differences is called rugged [Hordijk, 1996]. Different landscapes perform differently in their ruggedness. Landscape correlation length (LCL) presents the ruggedness of a fitness landscape. The ruggedness of the landscape to a given problem measures the degree of correlation between neighboring states in a random walk [Hordijk, 1996]. LCL is computed as:

$$l = -\frac{1}{\ln|(\rho(1))|} \tag{5.2}$$

Where

$$\rho(1) = \frac{1}{\sigma^2(f)(m-1)} \sum_{t=1}^{m-1} (f_t - \bar{f})(f_{t+1} - \bar{f})$$

is used to measure the degree of correlation between consecutive points on the landscape [Angel and Zissimopoulos, 1998]. In this equation, $\sigma^2(f)$ denotes the variance of the fitness, m denotes the length of the random walk, f_t is the fitness score in time t, and $\bar{f}$ denotes the mean fitness. The degree depends on the difference between the fitness values of the points. For example, similar fitness values make the landscape smoother and the search for a specific algorithm easier. The higher of the value of LCL, the smoother the landscape is; the lower the value of LCL, the more rugged the landscape. From performance of value of LCL, the fitness landscape structure can reflect how easy or difficult it is for a search algorithm to find a good solution [Manderick et al., 1991] [Reeves, 2000] [Mitchell et al., 1992].

5.2 Topological Node Ordering on Bayesian Network Structure

From our previous results, we know that search-and-score approaches using nature-inspired metaheuristics (GA and ACO) have been successfully applied to BN structure learning. However, the difficulty of structure learning makes the choice of suitable algorithms and scoring functions for certain problems play an important role on the design of approaches. We have investigated both the chain model and K2 greedy search based algorithms with node juxtaposition distribution on benchmark structures. The results indicate that the chain approach is able to yield high quality solutions with significantly less computational effort than the K2 approach in problems where the true structure of the data is amenable to alignment of node juxtapositions in a single ordering. In problems where such alignment is not possible, the chain scoring approach is likely to be unsuccessful in producing high quality structures and so the relative benefit of reduced computational time is lost. When we have a benchmark BN structure, the topological node orderings of this structure can be generated according to the Kahn or Tarjan algorithms. These kind of orderings are the best node orderings for a K2 algorithm to induce the BN structures. We regard these orderings as the best solutions to the specific studied problem. The fitness score according to these orderings then become the global optimum to the structure. The topological sorts give us the way to achieve the global optimum in BN learning. In spite of this problem is a NP hard problem. So in our research, we take them to yield topological node orderings which will be applied to investigate the fitness landscape analysis.

5.2.1 Kahn and Tarjan Topological Node Ordering on Car Structure

We use both Kahn and Tarjan algorithms to generate topological node orderings on benchmark structures. The CH and chain score based on these node orderings will explore the different performances of each search and score algorithm on related

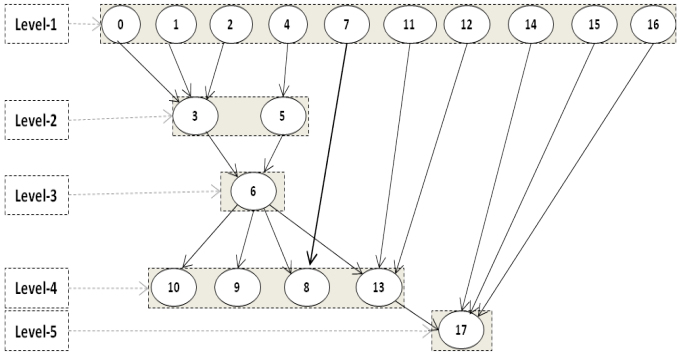


Figure 5.2: The true structure and group levels in Car network

structures. As an example, we describe the details of topological node orderings on Car benchmark structures below.

From the previous chapter, we know that when K2 algorithm is used to induce the BN structure, a node ordering on the variables is assumed. In the benchmark examples we consider here, the true structure of the network is already known. In this section we consider orderings that are consistent with this structure in the sense that, starting with the ordering, K2 is capable of recovering the true structure. We will refer to such an ordering as a true ordering. In general, there are many distinct orderings consistent with the true structure. The key property is that K2 is able to add each correct edge. In other words, if there exists an edge from node i to node j, then i must precede j in the ordering (though not necessarily as an immediate predecessor). This is precisely the property that the ordering is a topological sort of the nodes based on structural dependence. Figure 5.2 shows the true structure and level groups of Car network [Kabli et al., 2007]. We note that Car, an 18-node network, has a level-set size of 10 and 5 levels of dependency. In Figure 5.3, we show a random breadth-first(Kahn) representation node ordering of the Car network. In Figure 5.4, it is a random depth-first (Tarjan) representation node ordering of the Car network. In Figure 5.3 and Figure 5.4, the dotted lines indicate the true edges to the original structures.

In Figure 5.2, Level 1 contains all independent nodes. Subsequently, for $n > 1$,

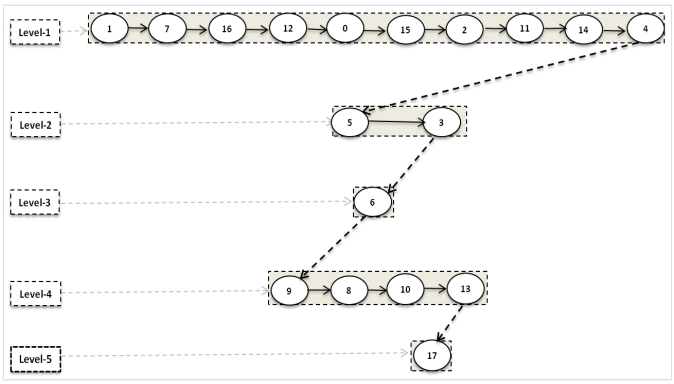


Figure 5.3: Topological node ordering in Car network on Kahn Algorithm

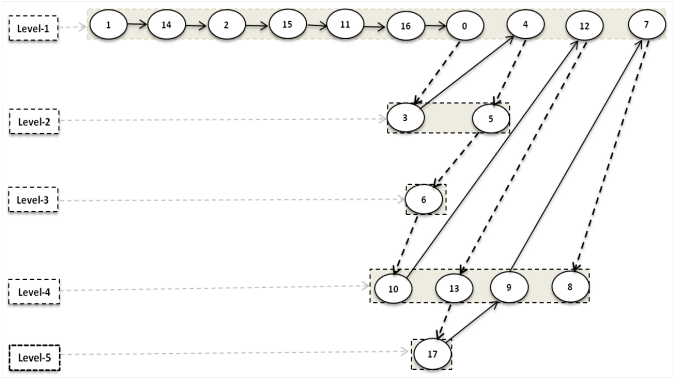


Figure 5.4: Topological node ordering in Car network on Tarjan Algorithm

Level n contains all nodes that become independent only when nodes of level $l < n$ are removed. Any ordering that internally preserves the order of node level groups is automatically a true ordering. In Figure 5.3, when the Kahn algorithm is applied, a node ordering is produce. This sort is 1-7-16-12-0-15-2-11-14-4-5-3-6-9-8-10-13-17. In the Kahn algorithm, we note that any node exchange within each of the level groups 0-1-2-4-7-11-12-14-15-16, 3-5, and 10-9-8-13 will remain true orderings. For example, node 1-0-2-4-7-11-12-14-15-16-3-5-6-10-9-13-8-17 comes from the swapping of any two pairs of nodes in level 1 and level 4. K2 will induce the same BN structures and return the same CH score. This indicates plateaux in the fitness landscape for K2 around breadth-first sorted true orderings. The position for chain is more complicated. To evaluate a chain we force an edge between juxtaposed nodes. Clearly within level

groups this inserts edges between independent nodes and so low scores will be associated with such arcs. A chain structure imposes a total order on the nodes whereas a topological sort imposes a partial order on the nodes. Search methods based on chain score would therefore be expected to perform well when the network structure admits topological sorts "close to" total orderings.

In Figure 5.4, when the Tarjan algorithm is applied, one node example is the ordering 1-14-2-15-11-16-0-3-4-5-6-10-12-13-17-9-7-8.This node is not limited to the same level any more as it exceeds from each level to another one. A significant property in this case is that it has more edges that correspond to edges in the true structure (0-3, 4-5, 5-6, 6-10, 12-13,13-17,7-8). In this sense an ordering produced by the Tarjan algorithm, which we term a Tarjan ordering, is likely to be more amenable to chain-based search-and-score methods. We notice that Car is broad and relatively narrow as a structure and so the breadth-first orderings produced by Kahn are more likely to dominate.

5.2.2 Kahn and Tarjan Topological Node Ordering on Alarm Structure

Figure 5.5 is another example that shows the true structure and level groups of the Alarm network [Beinlich et al., 1989]. Alarm by contrast is a 37-node network with a larger level-set size of 12 and 11 levels of dependency. We note that Alarm is relatively narrow and deep in comparison to Car. Intuitively we expect that the ratios of level-set size and number of levels to the overall number of nodes in the network and to each other will relate significantly to the performance of K2 and chain based search-and-score algorithms. In Alarm, we can get node orderings using the Kahn algorithm as in Car structure. The possible maximum number of true edges in all these orderings is 10; this is because the structure has 11 levels. However, the Tarjan algorithm can easily achieve a larger number of true edges. For example in the random ordering 31-20-37-36-35-33-32-34-11-30-3-29-28-24-26-25-23-17-16-27-22-21-19-18-15-14-13-12-10-4-1-2-9-8, the true edge number now is 13 (35-33,30-3,29-

28,26-25,25-23,17-16,27-22,22-21,21-19,19-18,18-15,15-14,14-13,13-12,12-10,9-8).

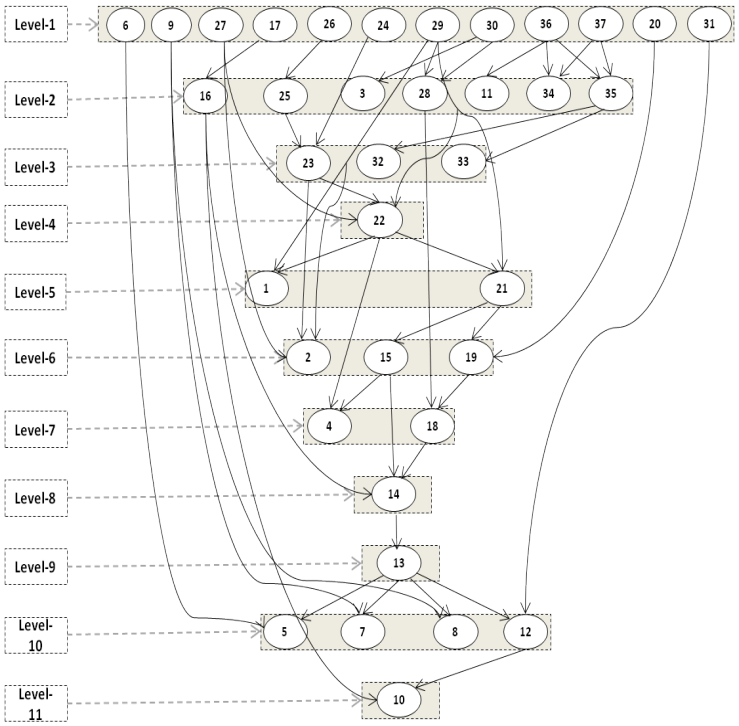


Figure 5.5: The true structure and group levels in Alarm network

From analyzing in this section, we notice that the different values of width and depth to each structure will affect the Kahn and Tarjan type orderings. We also expect that the total number of true edges in different original structures will lead to the different performance of landscape to each ordering. To understand the structures and generate topological node orderings for our experiments in a later section, we describe properties to all eight benchmark structures we studied previously in Table 5.1. These structures are: Asia, Credit, Tank, B, Boerlaga, Insurance, Car and Alarm. The properties in Table 5.1 include the number of width, depth, true edges, the ratio of width to true edges and depth to edges, and the percent values of true arcs produced by Kahn and Tarjan algorithms. We got these percent values through randomly generating 1000 node orderings on Kahn and Tarjan algorithms respectively, comparing the number of trues edges that emerged in these orderings to the total number of edges

Table 5.1: Properties of Group Level on Benchmark Structures

Structure	N	E	W	D	W/E	D/E	True Arcs	
							Kahn	Tarjan
Asia	8	8	3	4	0.37	0.50	0.31	0.52
Credit	12	12	7	4	0.58	0.33	0.13	0.35
Tank	5	20	14	4	0.25	0.20	0.14	0.39
Car	18	18	10	5	0.55	0.27	0.14	0.27
B	18	39	5	6	0.12	0.16	0.20	0.52
Boerlaga	23	36	3	14	0.08	0.39	0.41	0.71
Insurance	27	52	5	10	0.09	0.18	0.18	0.44
Alarm	37	46	12	11	0.26	0.24	0.13	0.34

in the original structures. These values show the difference between each algorithm on producing the true edges. In Table 5.1, N represents the nodes number of each structure, E is the number of original edges in the structure, W indicates the width, and D shows the depth of the specific structure.

5.3 Experimental Methodology

We carry out experiments to conduct fitness landscape analysis on the two score functions of BN structure learning. The two previously discussed two basic topological sorts, Kahn and Tarjan, are utilized to each benchmark problem for producing fitnesses. Some details of the implementation and parameters are provided below.

5.3.1 Random Walks with Bubble Swapping with Topological Node Orderings

The search space in our problem is composed of all node orderings. These include the Kahn and Tarjan topological node orderings. We take both Kahn and Tarjan

node orderings as starting node orderings as they are the optimal solutions for each approach. A standard mutation with random walk is defined for constructing the fitness landscape in the search space. Since swaps of consecutive nodes in a topological sort retain the property of being a topological sort, consecutive node swaps are selected as our minimal random mutation operator. As consecutive item swaps are well-known in the Bubble-sort algorithm, we here term them bubble-swaps for brevity. Figure 5.6 shows a sequence of bubble-swaps on topological sorts that produce a path from the global node ordering. In Figure 5.6, X_0 represents an optimal solution, and $X_1, X_2, ..., X_t$ are a series of node orderings after bubble swapping. $d_1, d_2, ..., d_t$ indicate the number of bubble swaps in this process.

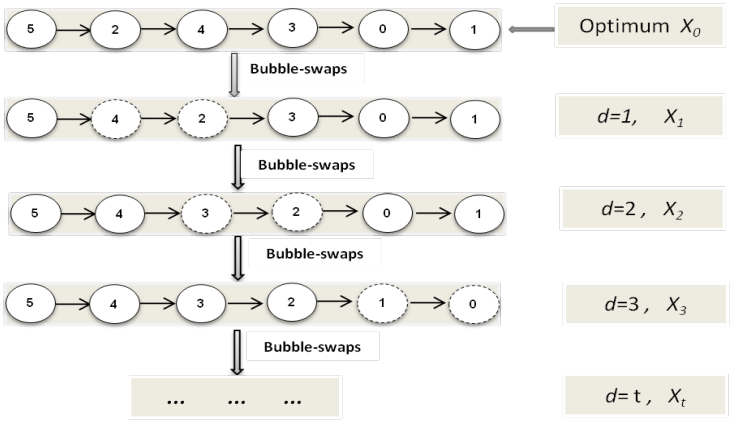


Figure 5.6: Bubble-swaps and distance to a topological sort ordering

5.3.2 Experimental Design

To structure fitness landscapes within this space, firstly, we randomly generate topological sorts, both Kahn and Tarjan, for each structure. All these node orderings will be regarded as the global optima solutions X_0 to the fitness landscape in the space of orderings for a particular BN structure problem. Afterwards, random swaps of consecutive nodes in each topological sort will be executed to get a series of node orderings $X_1, X_2, ..., X_t$, corresponding fitness score (chain score and CH score) achieved

to these sorts. After a certain number of swappings, fitness landscapes are then constructed according to the walks from the global solutions. A general template of random walk with swapping is shown in Algorithm 15.

Algorithm 15 Pseudocode of Random Walk with Random Bubble Swapping

1: **Begin Random Walk**
2: $S \leftarrow$ Optimal node ordering (length=n) from Kahn or Tarjan topological orderings for each approach.
3: $d \leftarrow 0$, d is distance to optimal solution.
4: Chain score, CH score on (S) and d.
5: **for** $i \leftarrow 1$, MaxNumberofWalk **do**
6: Select a random position $j(0 < j < n)$ on S.
7: $S \leftarrow$ bubbleSwap (S, j) with S.
8: $d++$.
9: Record Chain score, CH score on (S) and d.
10: **end for**
11: **Output:**Chain score and CH score
12: **End Random Walk**

To construct and measure the performances of fitness landscapes, we record some metrics. These metrics are:

CH and chain scores. We will use CH and chain fitness scores to each topological orderings X_0 and the series of orderings generating through bubble swapping $X_1, X_2, ..., X_t$ to construct the fitness landscape to each structure respectively.

Distance from the global optimum. We define the distance d in relation to the number of Bubble-swaps from the global optimum. The value of d equals the number of random swaps to the optimal node ordering. Greater values of d means a longer distance from the optimum.

In our experiments, we set the total walk length (MaxNumberofWalk in Algorithm 15) as 5 times the length of each node ordering. This is mainly because of the time consumption. From previous results, once the random walk step is large, it is very expensive for score function operations, especially for CH score function. For

example, for the Alarm network, the node number is 37 so the steps for swap will be 185. So the minimum distance is $d_{min} = 1$, and the maximum distance is the total swapping number $d_{max} = 185$. In all cases, we carry out 100 experimental runs each to get the averaged results.

5.4 Results and Discussion

To compare the fitness landscape on two score functions, we describe all the results in this section. Figures 5.7a– 5.7h and Figures 5.9a– 5.9h indicate the fitness landscapes for chain score and CH score to all benchmark structures respectively. In each of these figures, the fitness landscapes come from Kahn type orderings and Tarjan type orderings respectively. From Figure 5.7a to Figure 5.7h and Figure 5.9a to Figure 5.9h, each figure indicates the results of Asia, Credit, Tank, Car, B, Boerlaga, Insurance and Alarm respectively. In these figures, the X axis represents the walk distance (steps of random Bubble swaps) in the search space, and the Y axis expresses the chain score and CH score values.

We also give properties of FDC and LCL to these fitness landscapes. These properties can help us to distinguish the landscapes and understand the difference of Kahn and Tarjan node orderings on chain and CH score functions. We compare the values of FDC and LCL in Figure 5.8a– 5.8b and Figure 5.10a– 5.10b respectively. The histogram in these figures indicate the coefficient of FDC and LCL to each algorithm and the labels '1'–'8' in each figure represents the structure Asia, Credit, Tank, Car, B, Boerlaga, Insurance and Alarm.

5.4.1 Fitness Landscape on Chain Score Function

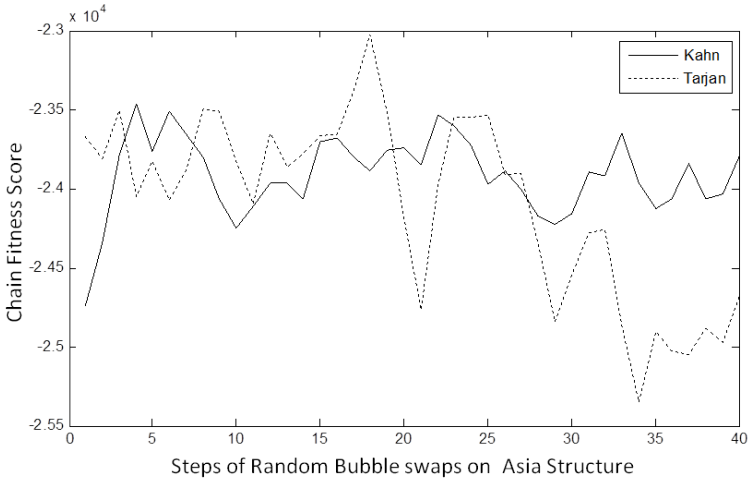
We consider the fitness landscape of Kahn orderings first. We find in Figures 5.7a– 5.7d and Figure 5.7g that the graphs look really flat as the random walk moves away from their corresponding optimal orderings. These results indicate that algorithms using chain score will struggle to converge optimally in these structures. In Figure 5.7f

and Figure 5.7h, the graphs have a clear gradient, these explain that the algorithms using chain score in these two structures will tend to promote small changes that lead toward the optimal. However, the graph in Figure 5.7e is increasing as the distance to global optimal increases. It indicates that chain score to Kahn ordering will mean that it is impossible to converge on optimal in B structure.

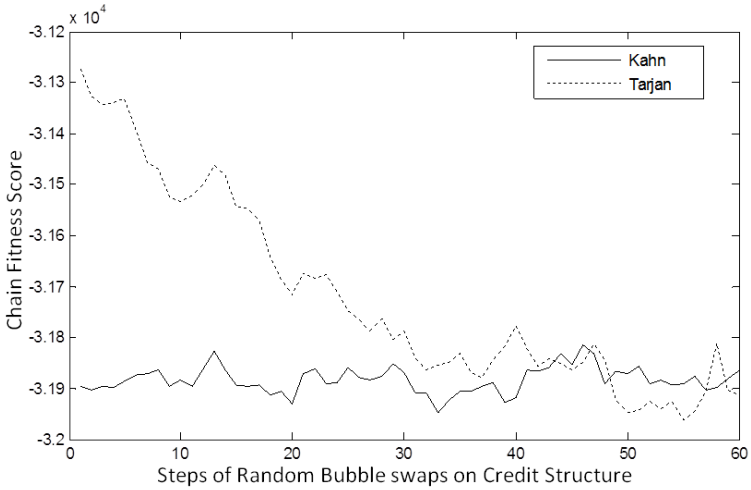
We then investigate the graphs of Tarjan orderings on chain score to benchmark structures. In spite of some structures, such as Asia and Car, where the graphs express fluctuation, all the graphs in these figures are clear gradient as the distance to the optimal increases. This performance is very different from the Kahn algorithm on chain score. It indicates that when the Tarjan algorithm is applied to generate node orderings they will give more attractive than Kahn orderings in close proximity to the optima.

Figure 5.8a indicates the difference of FDC between Kahn and Tarjan node orderings to each benchmark structure on chain score. We can see that in structure B, the values of FDC on both Kahn and Tarjan are very large and positive. This means that the problem is misleading and the algorithms on chain score will be led away from the global optimum. This result is explanatory of the observed performance of B in Table 3.8. Both chain and K2 based algorithms have poor performance in recovering the true structure. The FDC value on Kahn algorithm in Credit and Tank structures, are also positive. The value on Tarjan algorithm are negative, so we know the Tarjan algorithm should perform better than Kahn algorithm. Comparing values of FDC on other structures, the common performance is that all values are negative and the values for the Tarjan algorithm are better than for the Kahn algorithms, except on Alarm, where the value for the Tarjan algorithm is a little lower. These can explain why the performance of the chain score function is sensitive to Tarjan algorithm but not to Kahn algorithm in Figure 5.7a– 5.7h.

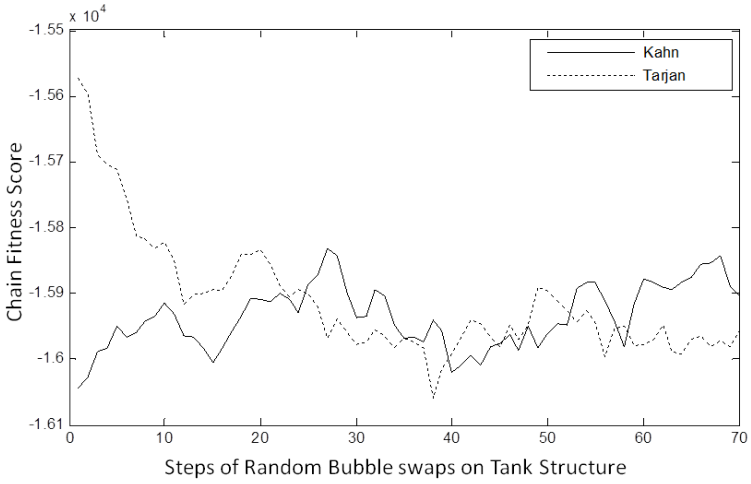
Figure 5.8b indicates the difference of LCL between Kahn and Tarjan orderings to each benchmark structure on chain score. We see that the Kahn orderings have a



(a)

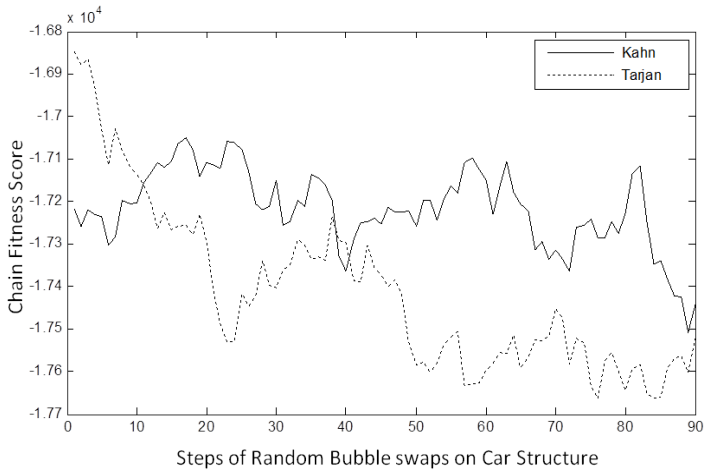


(b)

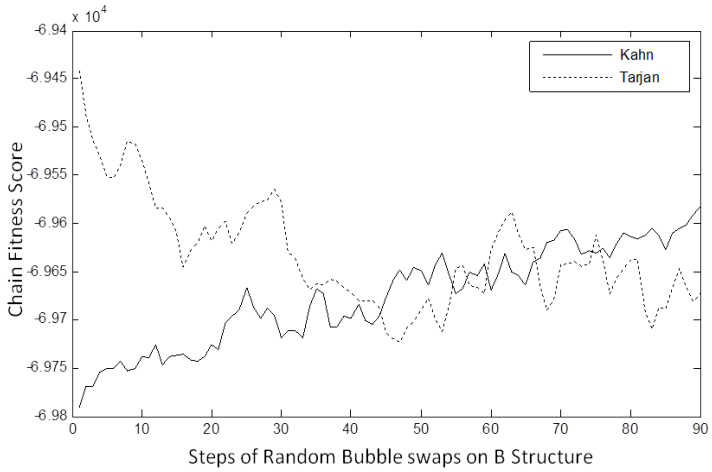


(c)

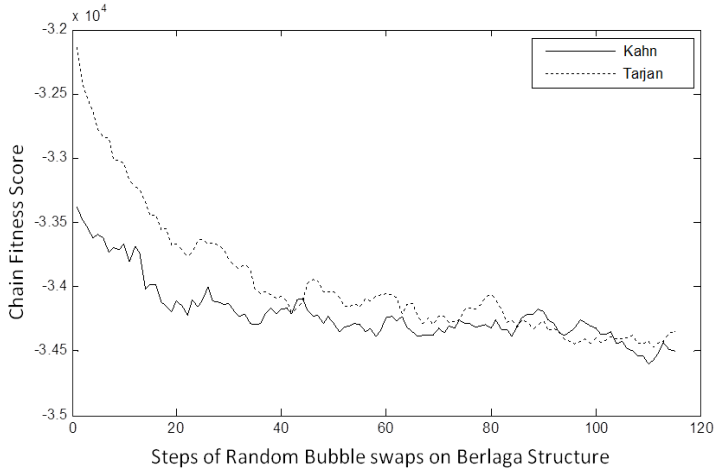
Figure 5.7: Comparative chain score fitness landscapes around Tarjan and Kahn orderings on benchmark structures



(d)

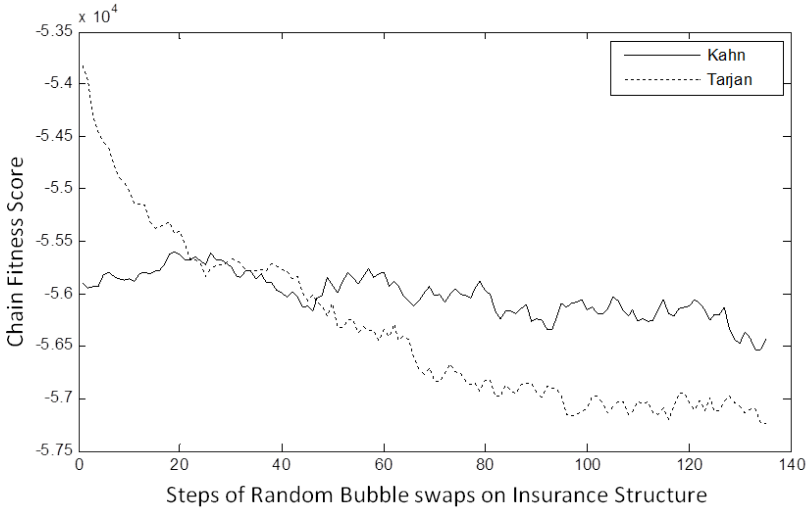


(e)

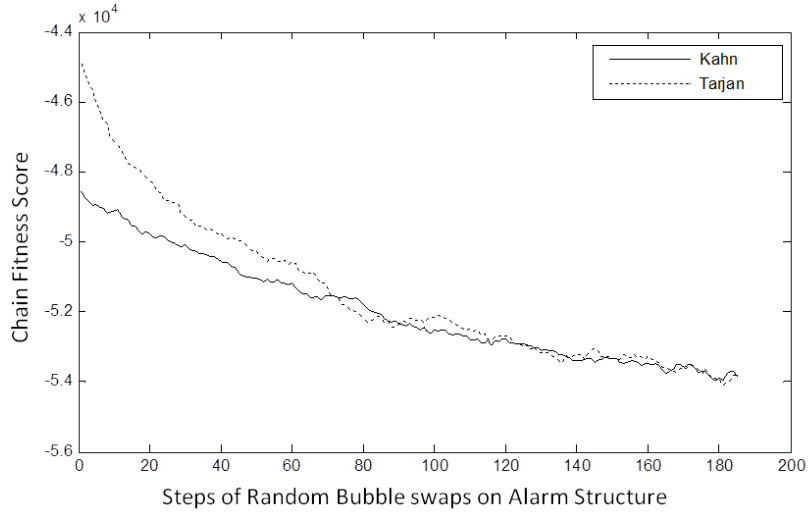


(f)

Figure 5.7: Comparative chain score fitness landscapes around Tarjan and Kahn orderings on benchmark structures

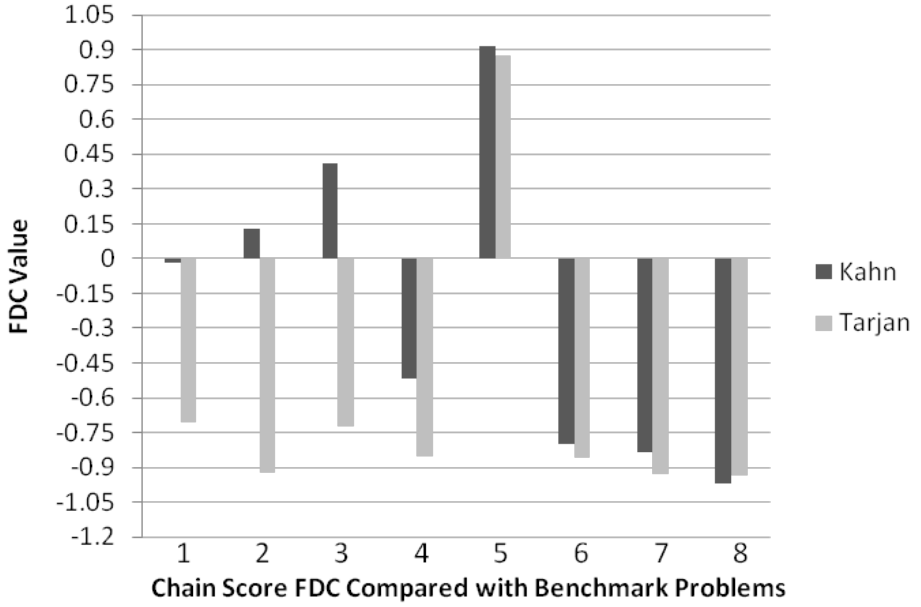


(g)

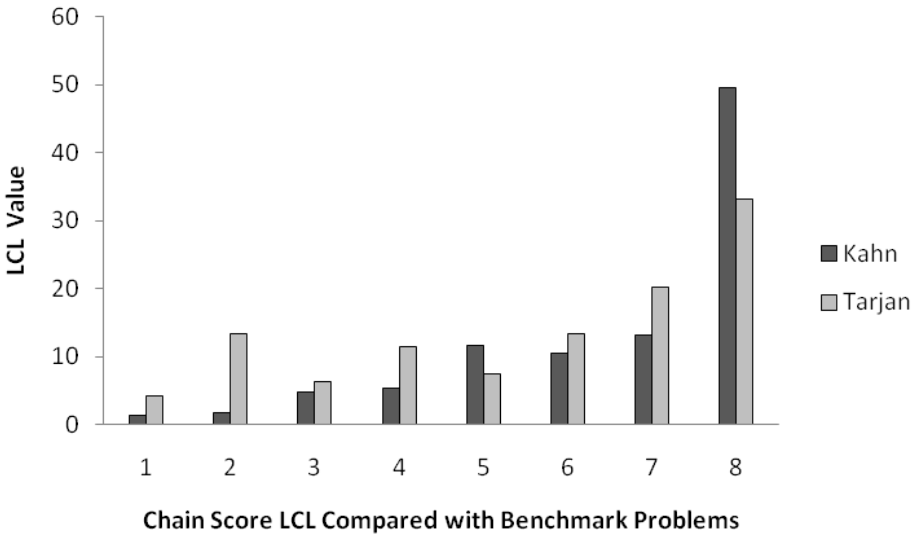


(h)

Figure 5.7: Comparative chain score fitness landscapes around Tarjan and Kahn orderings on benchmark structures



(a)



(b)

Figure 5.8: Comparative of FDC vlaue on chain score and CH score around Tarjan and Kahn orderings on benchmark structures

smoother performance than the Tarjan orderings only on the B and Alarm structures. The values indicate that there are significant differences between the two orderings. In all other structures, the Tarjan orderings perform well and get smoother plots. These properties can demonstrate why the search process is easy in specific node orderings.

Results on Figures 5.7a– 5.7h are explanatory of the observed performances of chain based search and score algorithms on these structures. We tested the algorithms using two different node orderings, Kahn and Tarjan type orderings. The experimental results show that there are clear different performances to chain approaches. Kahn type orderings present uncertainty and problem dependent to chain approaches. Tarjan orderings looks more effective and stable to chain score function approaches. In nearly all structures, the algorithms show strong abilities to converge to optimal solutions. This manifestation will motivate us to search suitable node orderings when we apply chain algorithms for large datasets.

5.4.2 Fitness Landscape on CH Score Function

When we compare all landscape graphs on Kahn orderings, only in Figure 5.9c is the landscape almost flat with the distance to the global optimal increasing. Other graphs, however, have the similar gradient tendency. In the starting node, the graph gets the best. With the increasing of random walk the score gradually decreases clearly. These indicate that algorithms using CH score will tend to promote small changes that lead towards the optima. Out of all landscapes of Tarjan orderings in these figures, only Figure 5.9c shows a very different tendency compared to other graphs. In this graph, the score function increases with the number of swaps increasing. This result indicates the Tarjan orderings using CH score will search randomly with very poor chances of reaching optimal. Other structures demonstrate a similar tendency in spite of some different gradients among them. For example, in Figures 5.9e, 5.9f and 5.9h, the graphs suggest that Tarjan orderings might be slightly more attractive than Kahn orderings for CH score in close proximity to the optima. In Figures 5.9b 5.9d and 5.9g

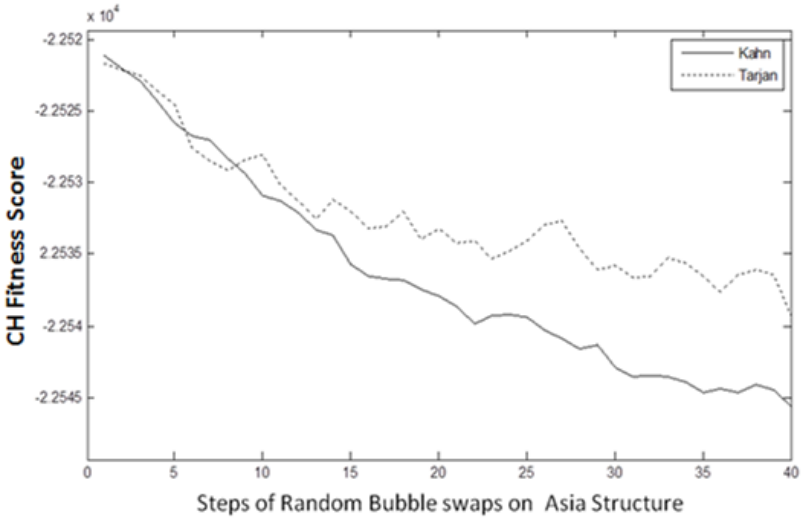
the graphs' tendency on Kahn and Tarjan orderings reveal that in some stage, Kahn orderings are easy for access to the optimal. In other stage Tarjan orderings find it easy to do this.

The comparative analysis of FDC in CH score gave us more clear results. We can see the values on both Tank and B structures are positive and closer to 1.0. In these cases, the fitness score and the distance has strong relation. However, this relation leads the algorithms to perform worse to the optimal solution both in Kahn orderings and Tarjan orderings. This means that the two algorithms find it hard to get good results when using the CH score function. This conclusion is consistent to the experimental results shown in Table 3.7 ~ Table 3.9. The value of FDC on the rest of the structures are all negative, both on Kahn and Tarjan orderings. These large negative fitness distance correlations ($r < -0.15$) tell us that the problem is straightforward and the algorithm will find the global optima with relatively good performance.

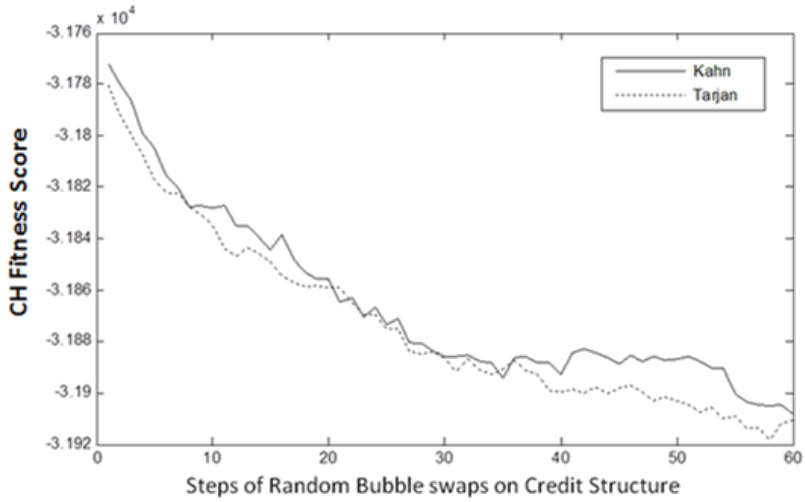
The values of LCL are described in Figure 5.10b on CH score. The comparative results show only in Credit and Tank structures that the Tarjan type ordering achieved a smoother plot. In all other structures, the Kahn type orderings got smoother plots.

The results above show the performance difference of Kahn ordering and Tarjan ordering on chain score and CH score based algorithms. On chain score algorithms, from the FDC results, we know that there is a significant difference between Kahn and Tarjan orderings to specific structures. This result indicates that the selection of different kinds of node orderings in chain score algorithm has a vital affect on the BN structure learning. On CH score algorithm, Fitness Landscape results show both Kahn and Tarjan type orderings have similar performances to all structures. There is not a significant difference between Kahn and Tarjan orderings to any structure. Although they have a slight difference, the Kahn type orderings could make the search algorithm faster and easier from the FDC results. The results suggest CH score based algorithms have more attraction than chain algorithms to any kind of node ordering.

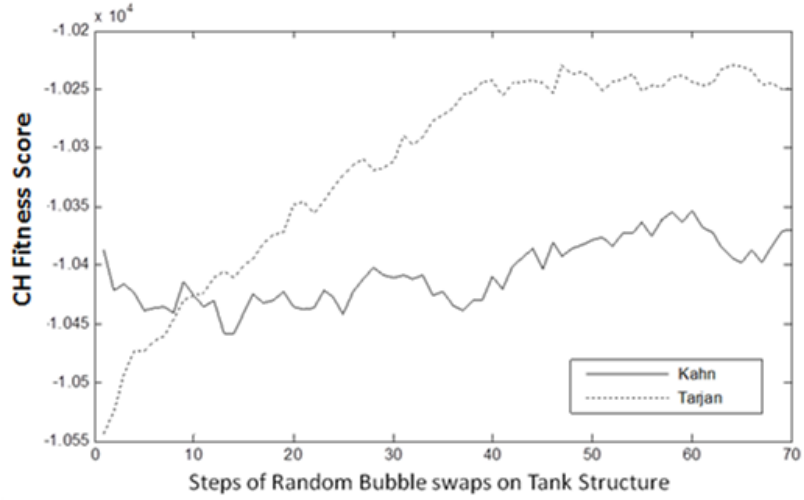
How can these differences be explained? We can explain these based on the



(a)

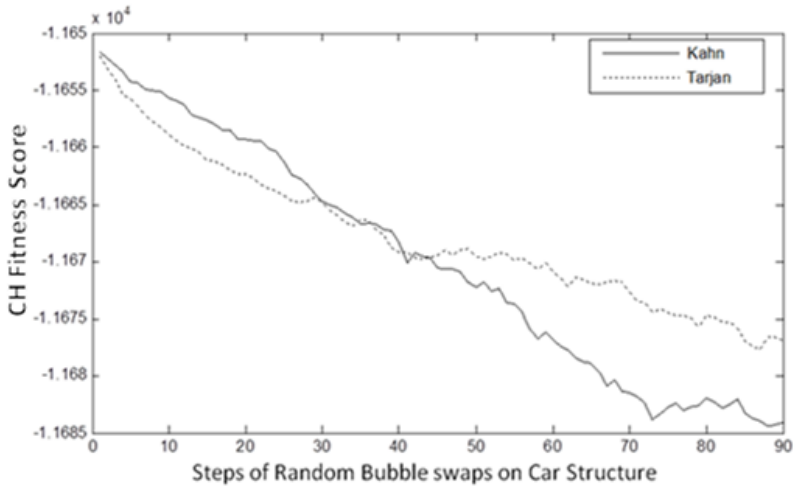


(b)

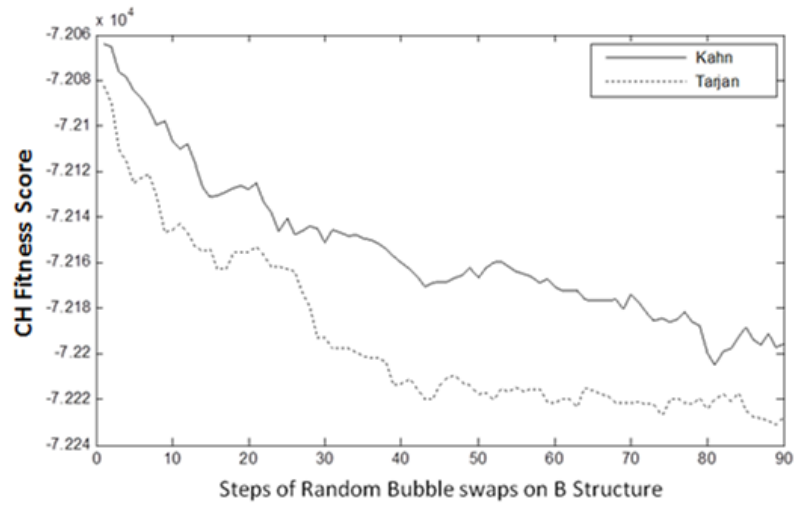


(c)

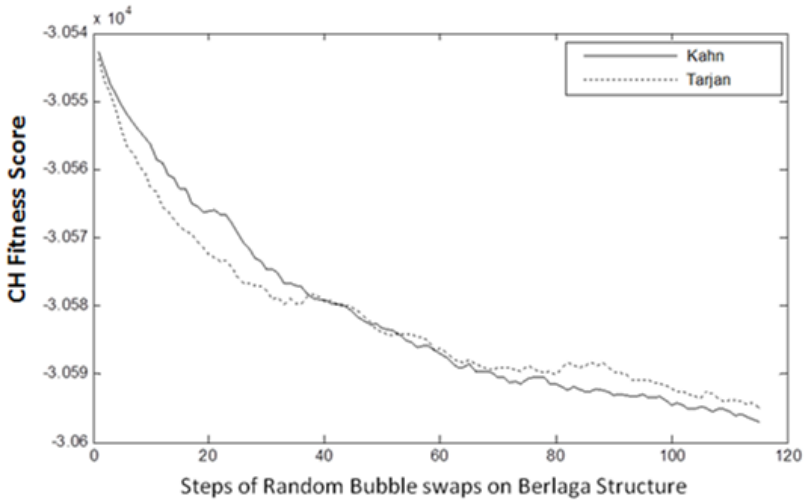
Figure 5.9: Comparative CH score fitness landscapes around Tarjan and Kahn orderings on benchmark structures



(d)

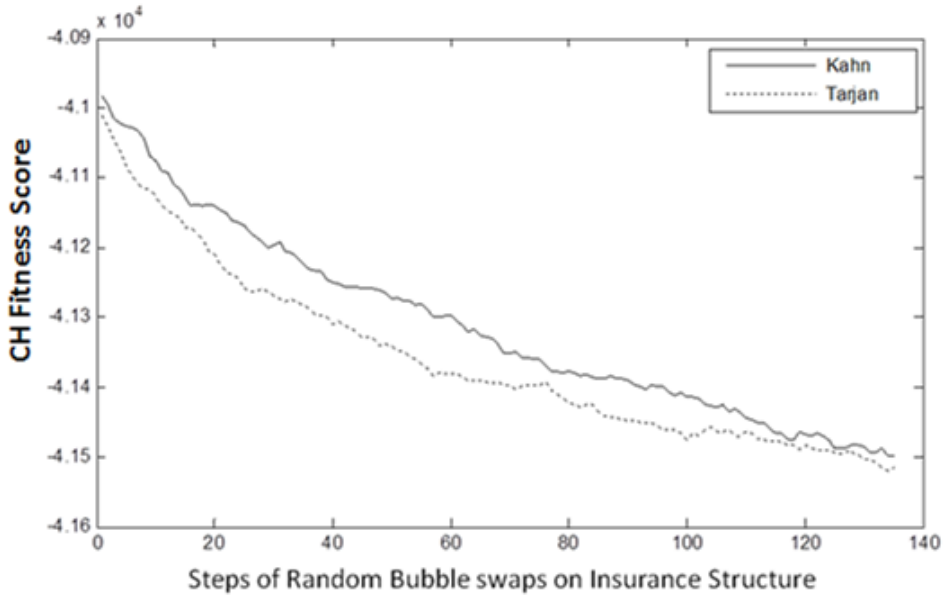


(e)

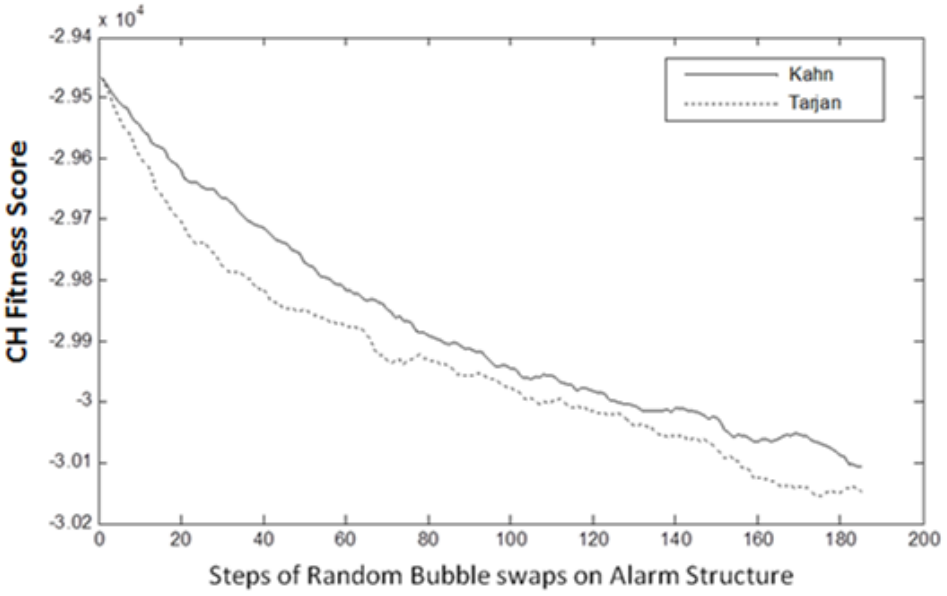


(f)

Figure 5.9: Comparative CH score fitness landscapes around Tarjan and Kahn orderings on benchmark structures

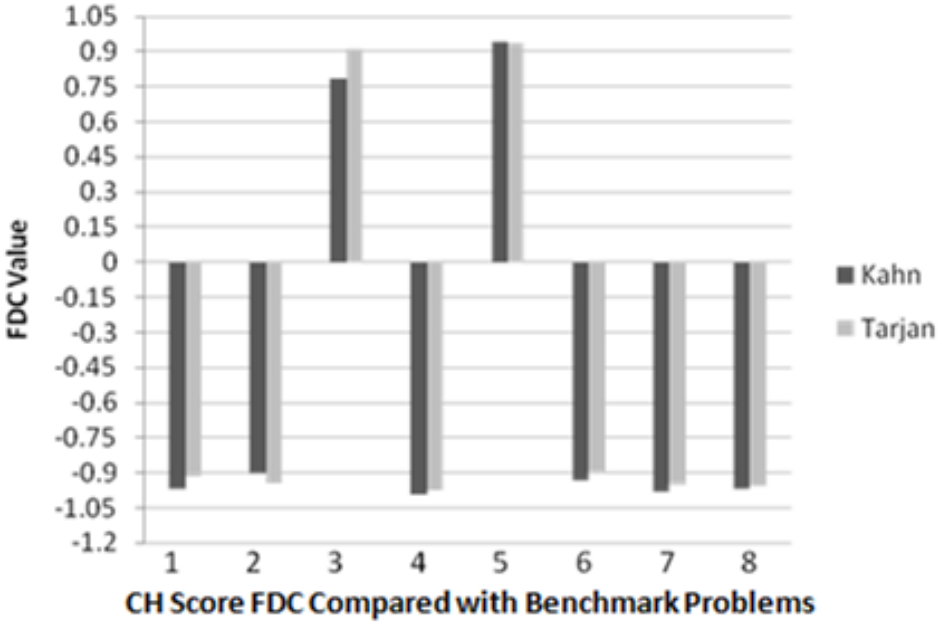


(g)

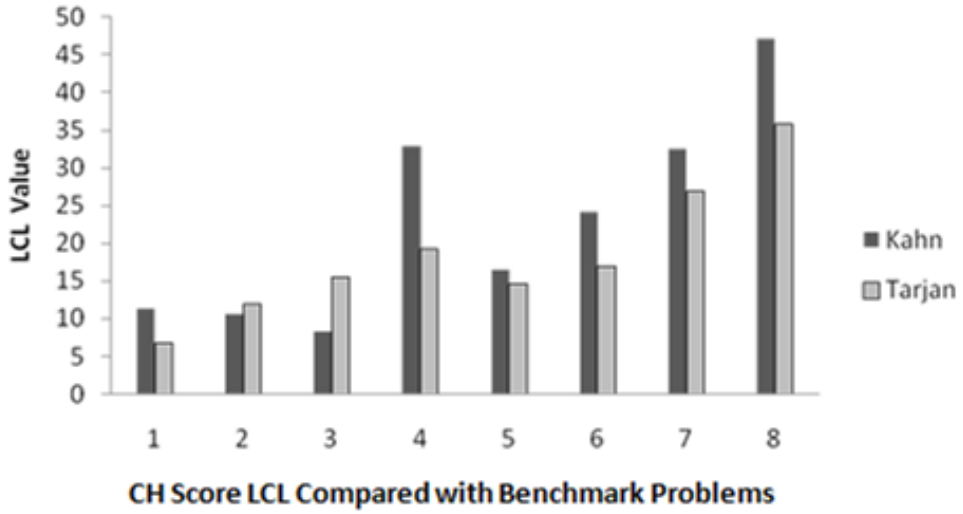


(h)

Figure 5.9: Comparative CH score fitness landscapes around Tarjan and Kahn orderings on benchmark structures



(a)



(b)

Figure 5.10: Comparative of LCL value on chain score and CH score around Tarjan and Kahn orderings on benchmark structures

definition of topological node orderings. From the previous section, we know that both Kahn and Tarjan algorithms generate node orderings by using node groups and levels of the structures. The depth and width of the structure affect the Kahn and Tarjan node orderings. More value of depth means more possibility of true edges produced when generating the Tarjan node ordering. This possibility also depends on the total true edges in specific structure. The value of D/E in Table 5.1 shows the relative results of depth to the total true edge of each structure. We sort these structures on the values from high to low: Asia, Boerlaga, Credit, Car, Alarm, Tank, Insurance and B. Comparing these orderings to results in previous results on each approach, we can find that chain based algorithm (ChainACO) got better results on structures with a higher value of D/E : Asia, Boerlaga, Car and Alarm. This performance can test and explain why in chain score algorithms, the Tarjan node orderings perform better in some structures. The structure of Credit has a higher value of D/E, but is poorer on chain based algorithm. The reason of this is that the value of W/E is relatively higher.

Table 5.1 also tells us that in all benchmark structures, the Tarjan orderings can get a higher percentage of true edges compared to Kahn orderings. When randomly swapping operations are executed to these specific node orderings, the true edge that exists in Kahn orderings is destroyed easily. However, the Tarjan ordering can still keep a higher percent number of the true edges. The change to chain score corresponding is not as dramatic as in the Kahn orderings. Simultaneously, the performance of LCL on CH score indicates that even with a higher number of true edges in the node orderings, the CH score model can't guarantee to get the smoother plot in some structures. This is mostly because of the computational complexity of K2 greedy algorithm.

5.5 Summary

In this chapter we proposed how to construct fitness landscape analysis for chain and CH score function based on two different topological node ordering; Kahn and Tarjan

sorts. We also investigated the performance of landscape around optimal orderings to related search and score algorithm on BN structure learning. The results have shown that empirically-observed differences in performance for search and score algorithms can be understood in terms of the relative breadth and depth of the DAG structure being learned.

Our results found that fitness landscape based on chain and CH score from Kahn-type topological sorts or Tarjan-type sorts have significant differences. Chain score functions can maintain the characteristic of Tarjan sorts when conducting random swapping. This makes the search process smoother and easier. CH score based landscapes, however, on the Kahn node orderings performs better sometimes. However we know, this process is to be expensive on most of the benchmark problems.

The space of node orderings is widely used for BN structure learning and so this technique has the potential to be applied to other search and score algorithms on other benchmarks. Our results also suggest that a characterisation of optima in terms of depth or breadth first topological sort may be useful in understanding the relative attractiveness of particular optima to particular scoring functions. Our results gave us some useful inspiration when learning BN structure using search and score approaches. The results indicate that the structure of node orderings will affect the search and score algorithms to a large extent. The Tarjan type orderings is suggested to be good for both of the algorithms discussed in this paper.

We know that search and score algorithms are popularly applied on BN structure. The feature of NP hard problem on BN structure learning makes it hard to choose a suitable approach for a specific new problem when the topological node orderings are unknown. However, the empirical results in this chapter can provide useful inspiration to us. The parameters of FDC and LCL on different node orderings have different characteristics. This behavior is consistent to the performance of each corresponding approach. For example, to all the chain scoring based algorithms, the values of FDC are negative. Except in B structure where both chain and CH scoring

algorithms perform very poorly. In the next chapter, we will investigate this kind of relationship to study how to make a decision on scoring based approach according to the results achieved in this chapter.

Chapter 6

Off-line Hyperheuristic Bayesian Network Structure Learning on Unseen Problems

In previous chapters, we have investigated the performance of chain scoring and CH scoring algorithms. We have empirically investigated trade-offs between computational cost and solution quality and have shown that the pattern of search is strongly related to the choice of search and score technique. Studied results on benchmark problems have shown that both chain and CH scoring based algorithms are problem dependent. It is therefore problematic to recommend a specific choice of scoring function for an unseen problem where the structure is not known. In the last chapter, we looked into the fitness landscape analysis through introducing topological node orderings on specific benchmark problems. The results have indicated that this concept is a useful tool for testing and explaining the existing empirical performance in search and score algorithms. The results also inspired us to deeply consider with the valuable parameters achieved for more complex research problems.

In this chapter, we propose to use the technique of Receiver Operating Characteristic (ROC) curve to construct a classifier method based on utilizing the fitness

landscape analysis to randomly select the node orderings on specific problems. We aim to use the parameter of FDC on chain score landscapes as a continuous classifier for unseen BN structure learning problems. This will help us to build a selective off-line hyperheuristic algorithm for unseen BN structure learning with search and score algorithms. This is a special case of the widely-recognized difficulty of using performance on benchmark problems as an indicator of performance on unseen problems [Corne and Reynolds, 2011].

The main contribution in this chapter is that we define the process of how to construct ROC curves with parameters of fitness landscape analysis on BN structures. We apply an iterated series of bubble swaps to the randomly generated node orderings to produce the value of correlation between fitness and distance (FDC). We also design algorithms for generating new benchmark problems from previous standard structures to ensure a precise ROC curve is produced.

6.1 Hyperheuristics

This chapter describes a off-line hyperheuristic algorithm which is specifically designed from our previous experimental results on some benchmark structures. The main goal of this work is to develop a hyperheuristic which makes use of the properties of low-level heuristic algorithm (ACO, GA, K2, etc) in BN structure learning. Through investigating the performance with chain and CH scoring algorithms individually and integrating with the parameters of fitness landscape analysis on specific problems, we will create a classifier rule for predicting the algorithm selection to some unseen problems. The hyperheuristic hopes to be successfully applied to a wide range of datasets for BN structure learning. The framework of hyperheuristics in BN structure learning are summarized in Figure 6.1

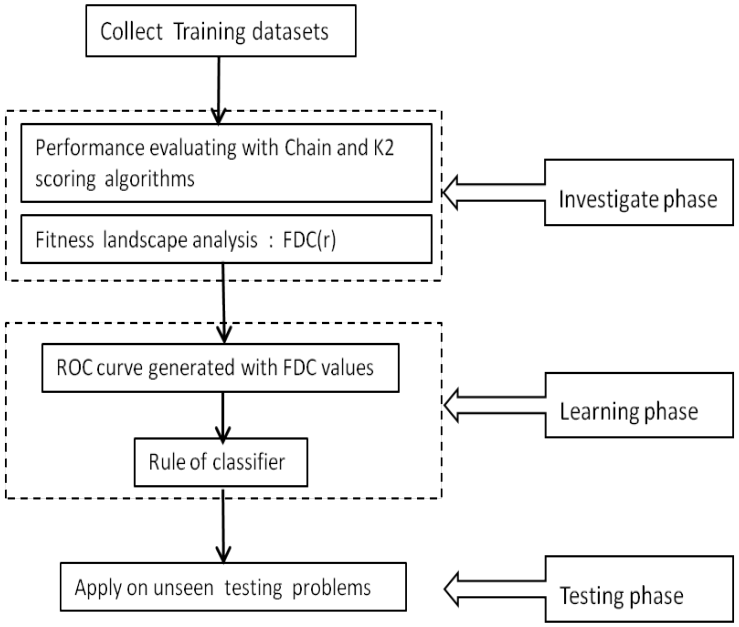


Figure 6.1: The framework of hyperheuristic on BN structure learning

6.2 Previous results of search and score algorithms on Bayesian Network Structure

In this section, we give a short summary on previous experimental results. This mainly includes the performance comparison of chain scoring algorithms and CH scoring algorithms and the results of fitness landscape analysis on these two kinds of algorithms. Table 6.1 provides a detailed comparison from the above aspects.

Table 6.1 presents the results of eight standard benchmark problems. Here '*' means the specific scoring algorithm that gives the best performance in terms of structure recovery. '−' and '+' represent that the value of FDC as negative and positive respectively. The results show the difference of performance between chain and CH scoring algorithms to these structures. The table also demonstrates that fitness landscapes based on chain and CH score algorithms have significant differences. Comparing FDC values to the performance of both chain and K2 algorithms, we observe informally that chain score FDC is consistent with the performance of chain

scoring algorithms to a large extent. For example, chain scoring algorithms perform better on four benchmarks (Asia, Car, Boerlage and Alarm), whose values on FDC are negative both on Kahn and Tarjan optima. In the remaining four problems, except Insurance, the FDC values are positive. In a CH score based landscape, the CH score FDC is inconsistent with the performance of CH scoring algorithms. The FDC values on 6 of the 8 benchmarks are negative. However, CH scoring algorithms perform well on only 2 of them. The FDC values on Tank are positive but, counter-intuitively, CH scoring algorithms perform better on it.

The above investigation on the performance of search and score algorithms and fitness landscape analysis suggest that the FDC measure for chain score algorithm may be the basis for a useful classifier. This classifier is hoped to have the ability of automatically provide decisions on how to select the suitable algorithm for specific BN structure learning problems. Previous experimental results have indicated that chain algorithms are cheap in all benchmark problems. This means that the calculation of FDC feasible for a large number of bubble swaps to benchmark problems. This will contribute to the classifier model constructing easily and expand the applicability of this model. We give the details of related knowledge in the following sections.

6.3 Receiver Operating Characteristic with Bayesian Network Structure Learning

6.3.1 Receiver Operating Characteristic

Receiver operating characteristic (ROC) curves are a popular way for organizing classifiers and visualizing their performance [Martin et al., 1997] [Metz, 1978]. They are commonly used in medical decision making and diagnostic tests [Swets et al., 2000]. In 1989, Spackman [Spackman, 1989] first adopted it in machine learning problems, and demonstrated the value of ROC curves in evaluating and comparing algorithms. After that, ROC was increasingly applied in the machine learning

Table 6.1: Properties of Group Level on Benchmark Structures

	Performance result		Chain FDC		CH FDC	
Structure	Chain	CH	Kahn	Tarjan	Kahn	Tarjan
Asia	*		−	−	−	−
Credit		*	+	+	−	−
Tank		*	+	+	−	−
Car	*		−	−	+	+
B		*	+	+	+	+
Boerlage	*		−	−	−	−
Insurance		*	−	−	−	−
Alarm	*		−	−	−	−

and data mining research fields [Spackman, 1989] [Fawcett, 2008] [Flach and Wu, 2005] [Flach, 2003]. ROC plots the tradeoff between a test's sensitivity and its false positive rate. The analysis has appeared to offer more robust evaluation of the relative prediction performance of alternative models than traditional comparison of relative error [Obuchowski, 2003] [Webb and Ting, 2004].

We assume that ROC analysis is applied to evaluate the expected performance of a model with two classes. These classes are usually denoted as positive *'p'* and negative *'n'* class labels. A classifier model is a mapping from instance to predicted classes. There are four possible outcomes in prediction classes from this binary classifier. They are true positive (TP) if the outcome from a prediction is p and the actual value is also p; false positive (FP) if the actual value is n ; true negative (TN) when both the prediction outcome and the actual value are n, and false negative (FN) when the prediction outcome is n while the actual value is p. The 2×2 confusion matrix with four outcomes is described in Figure 6.2.

Some important numeric measures for a classifier are defined in a given test set once the four above outcomes are achieved. They are *true positive rate*(also as

		Predicted class		Total instance
		p	*n*	
Actual class	*p*	TP	FN	P
	n	FP	TN	N

Figure 6.2: The confusion matrix with four outcomes

sensitivity), $TPR = TP/P$, means the proportion of positive instances that are correctly classified as positive. *False positive rate*, $FPR = FP/N$ defines how many incorrect positive results occur among all negative samples available during the test. $Accuracy = (TP + TN)/(P + N)$, indicates the proportion of instances that are correctly classified. Here, $P = TP + FN$, $N = FP + TN$.

The TPR and FPR of different classifiers on the same test set are often represented diagrammatically by a ROC curve. The ROC curve is obtained by selecting a series of thresholds first, with these thresholds used to produce a series of corresponding FPR and TPR values. The ROC curve is built through plotting the value of FPR on the horizontal axis and the values of TPR on the vertical axis. It depicts relative trade-offs between benefits and costs. Each point in this curve can be written as a pair of values (x, y) indicating that the FPR has value x and the TPR has value y. This point corresponds to a classifier that can be obtained using a given decision threshold. Some important points in ROC are worth concern. The point $(0, 1)$ represents perfect classification. All the points lie in the diagonal line $y = x$ joining the bottom left and top right-hand corners corresponds to random guessing, which mean both the TPR and FPR will be 0.5. In ROC, one classifier is better than another if its corresponding point is to the north-west of the others. An example of a ROC curve with TPR versus FPR is described in Figure 6.3.

One of the most popular measures of the accuracy of a diagnostic test is the area under the ROC [Hanley and McNeil, 1982] [Green and Swets, 1966]. The Area Under

Curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. A test with $AUC = 1.0$ means that it will be perfectly accurate result because the sensitivity is 1.0 when the FPR is 0.0. On the contrary a test with $AUC = 0.0$ is perfectly inaccurate [Hanley, 2008]. Recently, more and more researchers use ROC AUC as a statistic tool for model comparison [Bradley, 1997] .

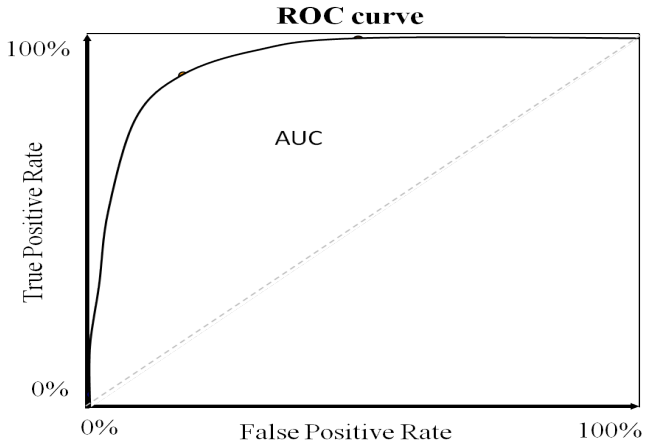


Figure 6.3: An example of ROC curve with TPR versus FPR

6.3.2 Receiver Operating Characteristic with Bayesian Network Structure

It is known that the optimal choice of scoring technique is problem dependent. Chain algorithms on some problems are cheap and can achieve better results. K2 algorithms, however, are expensive and may be poor on structure retrieval. Previous work using fitness landscape analysis on benchmark structures have shown that the values of the landscape that measure FDC are related to the performance of chain scoring algorithms. All these results inspire us to apply fitness landscape analysis techniques to predict the approach chosen for specific unknown structure in BN structure learning.

We achieve this through constructing a classifier method with ROC curves generated from fitness landscape analysis as follows. First we perform random walks from random starts on a training set where we know which particular scoring function

performs best. The random walks yield a set of FDC values r for chain score. We then use these to construct a continuous classifier on r. Finally we develop a hyperheuristic algorithm that selects either chain or CH scoring based on a chosen threshold for r. This is compared against an algorithm that makes blind choices of scoring technique on a test set of unseen BN structure learning problems.

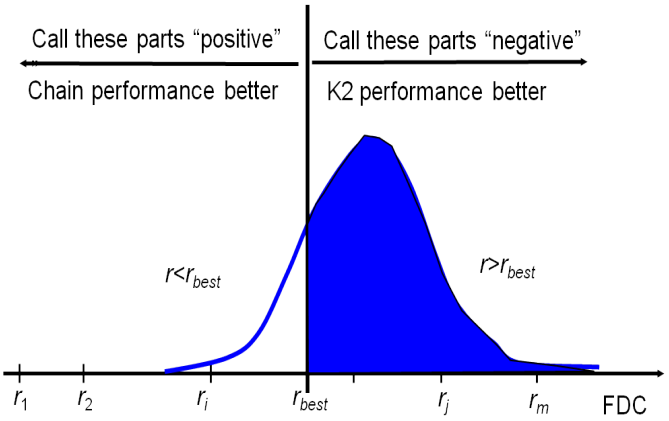


Figure 6.4: Predictive Outputs with best threshold on ROC model

In more detail, we give some hypothesis for constructing the model in this paper. If chain scoring algorithms perform better than CH scoring algorithms for a specific problem, we call this a positive instance and label it as *'p'*. Otherwise it is a negative instance, labeled *'n'*. From the fitness landscape analysis results described above, for a given threshold r_{best}, if the classifier output value r is under the threshold $r < r_{best}$, the classifier then produces a positive prediction *'p'*, else a negative prediction*'n'*. The outcome is true positive (TP) for an actual positive p class, is FP for an actual negative n. Each threshold value produces a different point in ROC space. If we generate a certain number, for example m of FDC values from all problems studied, the size of candidate thresholds is m , the range of a threshold will be from $-\infty$ to $+\infty$ and tracing a curve through ROC space. Figure 6.4 shows a ROC model with some thresholds and the corresponding output's instance.

6.4 Data Preparation and New Benchmark Structures Generating

We used the previously described eight benchmark structures in our experiment. We also derived some new benchmark structures utilizing a mutational technique from these problems for training in this chapter. Some of these new structures are used as a training set to generate a ROC curve and the remainder as an unseen set to test the predictive efficiency of ROC models. To create new problems, we take encoding documents (.xdsl) of each BN structure file from GeNIe&SMILE tool [GeNIe and SMILE,]. The file includes the parameters of conditional probability for the original benchmark structure. We randomly selected each conditional probability P_{ij} at rate P_{select}, and give it slight modification with increment $P_{increment}$. A normalization process is then executed to these modified values, making the sum of all of them is 1.0 to each specific variable. These new conditional probabilities are used to the GeNIeSMILE tool for generating new datasets of specific benchmark problem. The detailed description of how to modify the conditional probabilities for each new test problem is presented as Algorithm 16.

We run and compare both chain and CH scoring algorithms on this new data for BN structure recovering. The performance results of the chain scoring function relative to the CH scoring function based on fitness score of the best solution found on these new generated structures, are presented in Table 6.2. In this table, all the structures are divided into 6 groups, with 7 structures in each group. Each group has its own parameters of P_{select} and $P_{increment}$. Structures in group 1 are added with the former 8 structures for constructing the ROC curve. In order to conveniently describe in the following section, we number the structures, in this case from 9-15. The structures in other groups are used for testing the ROC model later.

Algorithm 16 Algorithm for modifying the parameters in conditional probability distribution

Input: rate P_{select}, mutation rate in conditional probability distribution;
power $P_{increment}$, mutation power in conditional probability distribution;
m, the number of nodes in specific structure;
n, the number of conditional probability of node i.
for $i \leftarrow 1$, m **do**
  $P_{sum} = 0$
  for $j \leftarrow 1$, n **do**
    if $(RandomNumber_{01} < r)$ **then**
      $lower = P_{ij} * (1 - p)$
      $upper = P_{ij} * (1 + p)$
      $P_{ij} = lower + RandomNumber_{02} * (upper - lower)$
    end if
    $P_{sum} = P_{sum} + P_{ij}$
  end for
  for $j \leftarrow 1$, n **do**
    $P_{ij} = P_{ij}/P_{sum}$
  end for
end for
Output: P_{ij}
End Algorithm

6.5 Experimental design

As explained above, the off-line hyperheuristic algorithm for BN structure learning includes the phases of search and score algorithm investigation, FDC and ROC learning phases and Testing phases. To construct it, we designed the experiments below:

Our experiments follow three stages. First, we perform a fitness landscape analysis on benchmark structures to get parameter values of FDC. Second, we construct a ROC curve with these results and known relative performance to get the best threshold. This threshold is used to parameterise a hyperheuristic algorithm that performs an initial random walk analysis before deciding whether to select either the chain or the CH scoring function. Finally we evaluate the hyperheuristic on a test set of unseen problems.

In detail, the first stage is specified as follows: for each benchmark problem, we randomly generate a certain number n of node orderings. Swaps of randomly-selected adjacent nodes are then repeatedly applied T times to each of the node orderings to produce corresponding chain fitness scores and distance. These values are applied to calculate the FDC. An average value r of FDC over all node orderings is then returned for each problem. The pseudocode of this process is presented as Algorithm 17. An example of some steps in a typical random walk is shown in Figure 6.5. We produced a series of FDC values r for all benchmark problems considered. Parameters used in Algorithm 17 are: $n = 100$, $T = 1,000$. We repeat each experiment 20 times to get the averaged value for later use.

Once we obtained all these FDC values, we sort them from smallest to largest. All these values are regarded as the thresholds in this case. With these thresholds and the hypothesis, for all benchmark structures, we can produce all the values of corresponding FTR and TPR. These are used to build the ROC curve. The detailed process of how to build the ROC is depicted as Algorithm 18.

In the final stage of the experiment, we will use some unseen Bayesian structures

Algorithm 17 Algorithm of FDC achieved on fitness landscape analysis to one benchmark problem

Input: T, the number of bubble swaps;
m, the number of random node ordering;
Randomly generate node orderings $S_i(i = 1, 2, , n)$ on search space.
for $i \leftarrow 1$, n **do**
  for $t \leftarrow 1$, T **do**
    Excuting Bubble swaps on node ordering S_i;
    Recording chain fitness score f_t and distance d_t;
  end for
  Calculating FDC value r_i to S_i on f_t and d_t;
end for
Average value of FDC $r = (r_1 + r_2 + + r_n)/n$;
Output r.
End Algorithm

to test and verify the predictive classifier model we have established in the second stage. The details of this experiment are as follows: we evaluate the performance of both chain and CH scoring algorithms to these structures first, simultaneously we apply the best threshold achieved to these structures to predict whether chain scoring or CH scoring is suitable for one unseen structure. Through comparing to what extent the empirical experiment result and the predictive classifier conclusion are consistent, we can get an intuitive evaluation to this ROC classifier on BN structure learning. Algorithm 19 indicates the detailed description of this process on how to test the enhanced algorithm.

6.6 Results and Discussion

Table 6.3 presents the averaged values of FDC achieved from Algorithm 17 over all 15 training problems. We also display the results of chain scoring function performance relative to the CH scoring function based on the CH score of the best solution found. Each r value defines a fixed classifier and so we can construct a ROC curve from this

Algorithm 18 Algorithm for ROC curve generated from binary discrete classifier

Input: $r[m]$, the sorted FDC values of m structures;
$c[m]$, the performance of chain scoring algorithm values of m structures;
N,the number of Y in $c[m]$;
P,the number of N in $c[m]$;
Add point(0, 0) to ROC curve.
for $i \leftarrow 1$, m-1 **do**
  $FP \longleftarrow 0$
  $TP \longleftarrow 0$
  $T_0 = r[i] + (r[i+1] - r[i])/2$
  for $j \leftarrow 1$, m **do**
    if $(r[j] < T_0)$ **then**
      if $(c[j] == Y)$ **then**
        $TP \longleftarrow TP + 1$
      else
        $FP \longleftarrow FP + 1$
      end if
    end if
  end for
  Add point(FP/N,TP/P) to ROC curve.
end for
Add point(1,1) to ROC curve.
End Algorithm

Algorithm 19 Algorithm for testing the predictive efficiency of ROC model.

Input: D_m, the data size of m unseen structures;
r_0,the best threshold in ROC curve;
$FP \longleftarrow 0$,$TP \longleftarrow 0$,$TN \longleftarrow 0$,$FN \longleftarrow 0$.
for $i \leftarrow 1$, m **do**
  Run chain scoring algorithm and CH scoring algorithm on data D_i, get fitness score of BN structure f_{chain} and f_{CH};
  if ($f_{chain} > f_{CH}$) **then**
    instance=Y
  else
    instance=N
  end if
  Execute Algorithm 17 on data D_i to get FDC r_i
  if ($instance = Y$ and $r_i < r_0$) **then**
    $TP \longleftarrow TP + 1$
  end if
  if ($instance = Y$ and $r_i > r_0$) **then**
    $FN \longleftarrow FN + 1$
  end if
  if ($instance = N$ and $r_i < r_0$) **then**
    $FP \longleftarrow FP + 1$
  end if
  if ($instance = N$ and $r_i > r_0$) **then**
    $TN \longleftarrow TN + 1$
  end if
end for
Output: FP,TP,TN.FN;
End Algorithm

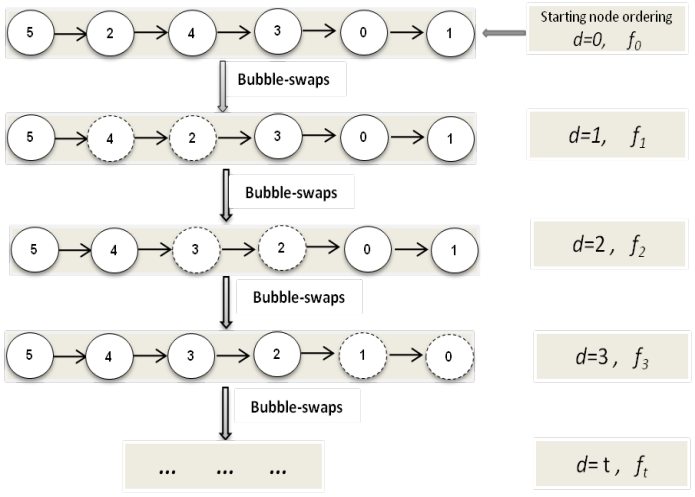


Figure 6.5: Example of Bubble swaps and distance to a topological node ordering

data.

Figure 6.6 shows the ROC curve generated according to Algorithm 18 on all the FDC values. In this problem, the TPR means the proportion of samples that are correctly classified as "chain scoring algorithm better" by the value of FDC. The FPR means the proportion of samples that are incorrectly classified as "chain scoring algorithm better" by the value of FDC. The FPR and TPR are expressed as percentages by multiplying the rate by 100. The curve is a step function curve. It indicates the thresholds and the corresponding classifier to a single point in all the considered problems. The best predictive accuracy value is 0.7333, which exists at point A with coordinates $A(0.3750, 0.8571)$ and the value of AUC, which in this case is 0.720. This value shows a strong relationship between FDC values and the outcomes of the algorithm classification performance on the training set. The true positive rate is 0.8571; this means it has a higher rate to classify positive instances correctly in just under 86% of cases. This is desirable since a correct positive classification means that the cheaper scoring function can be used without penalty (and possibly with benefit) to solution quality. To identify each threshold in this curve, we compare the points in the curve to the sorted FDC values which are predicted. Figure 6.7 shows all values

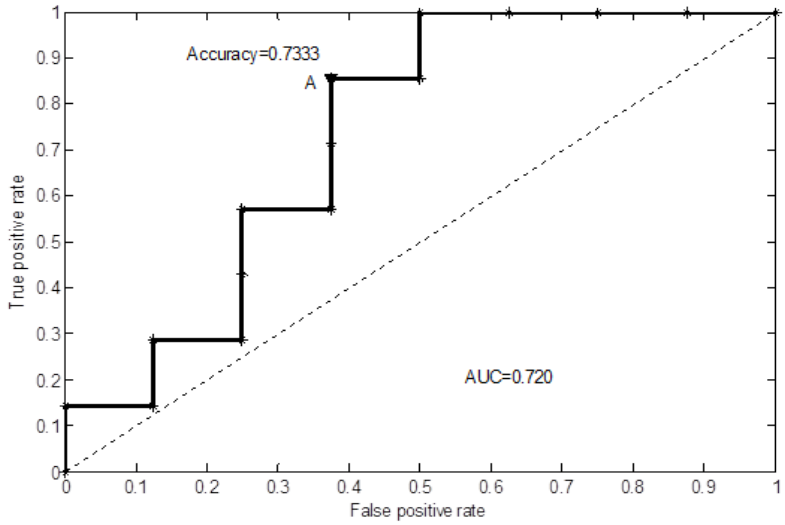


Figure 6.6: Roc curve generated from 15 benchmark structures

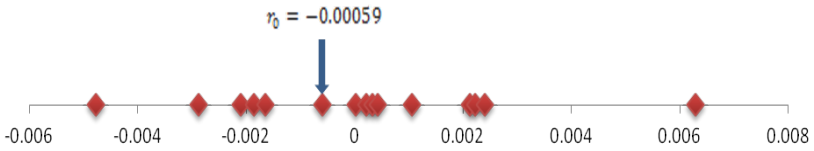


Figure 6.7: Sorted values of FDC in 15 benchmark problems

of r calculated from experiments arranged along a number line corresponding to the continuous classifier represented in the ROC curve. The thresholds are increased from $-\infty$ to $+\infty$ in our discrete classifier problem. We can see the threshold r which has the value corresponding to the best accuracy point A is $r_0 = -0.00059$, which is marked by an arrow in Figure 10.

We run experiments to test the best threshold achieved as described on Algorithm 19 . In this experiment, we employ 35 new problems, which we generated through Algorithm 16 as described above. Table 6.4 depicts the results on 35 new benchmark problems. In this table, the value of FDC (r) achieved from landscape analysis on the specific problem is presented. If the value of FDC (r) is less than the threshold $r_0 = -0.00059$, the chain scoring algorithm performs better. This is indicated as $*$, otherwise, it is space. The values of TN, FP, FN and TP are calculated through

comparing the testing prediction to the corresponding true performance of chain scoring algorithms on specific problems. Table 6.5 shows all the results from the above 5 groups. In this table, T represents the number of true positive classes in each group; N indicates the number of true negative classes in each group, $T + N = 35$. The values of TN, FP, FN and TP are the sum of all the corresponding values of TN, FP, FN and TP in Table 6.4. These are introduced for generating the parameters of FPR, TPR and accuracy corresponding to the hyperheuristic algorithm.

We also run experiments to investigate the values of parameters FPR, TPR and accuracy under the condition of randomly selecting algorithm for specific problems. In this case, we choose either chain scoring algorithm or CH scoring algorithm at random to each structure, and study the relationships to the performance of chain scoring algorithm or CH scoring algorithm in Table 6.2 for producing relatively TN, FP, FN and TP. The values of FPR, TPR and accuracy achieved here are compared to the output through best threshold and the testing algorithm described above. Table 6.6 indicates values of FPR, TPR and the accuracy for the training set, the test set using the classifier experiment and the test set using random selection of scoring function. As expected, classification performance on the test set is poorer than on the training set but is notably better than random selection of scoring function. The enhanced algorithm achieved an accuracy of 0.6285. Given that we are sampling from a very heterogeneous set of unseen problems, this can be regarded as a significant success. In particular, TPR was 0.7222 which means that, in 72% of cases where using the cheaper chain scoring function gave a benefit, the enhanced algorithm selected that option. Given the computational costs of the search, this is a significant result. Conversely, of course, a false positive result means that the chain scoring function is used when it does not provide a benefit. However, at 0.4705, the FPR for the enhanced algorithm is comparable to that of 0.4592 achieved by random selection so there is no real disadvantage here.

6.7 Summary

In this chapter, we have proposed a hyperheuristic BN structure learning algorithm. It uses a fitness landscape analysis-based classifier, learned from a training set of benchmark problems, to select the most suitable scoring function to use on unseen problems based on an initial random walk. We have also described a process for generating new unseen problems from known benchmarks and have generated a test set of 35 new problems. Our experimental results demonstrate that the selective hyperheuristic using the fitness landscape-based classifier has a significant advantage over blind selection of scoring function on unseen problems.

This chapter has also demonstrated that the ROC curve technique is very useful for binary classification in BN structure learning problems. The results in Figure 6.6 intuitively gave us a selected interval of all probable thresholds. This interval depends on the accuracy we need in prediction and the balance of TPR and FPR. In addition this chapter has shown that bubble swapping in fitness landscape analysis can provide valuable information from the specific studied problem. The node orderings for bubble swapping in this chapter are generated at random. This approach is different to the study in the last chapter, where we needed both Tarjan and Kahn node ordering. This means that it is feasible to produce fitness landscape analysis for any problems when we do not know its structure beforehand.

BN structure learning is an important problem. As interest in analyzing large data sets continues to increase, there is a growing need for approaches that can be adapted to new problems with some confidence. These initial findings suggest that fitness landscape measures such as FDC will prove useful in selecting suitable algorithms for unseen structure learning problems.

Table 6.2: Comparison Performance of Chain Scoring algorithms and CH Scoring on Benchmark Structures

	Group 1		Group 2		Group 3		Group 4		Group 5		Group 6	
rate	0.05		0.06		0.07		0.08		0.09		0.10	
power	0.05		0.06		0.07		0.08		0.09		0.10	
	Chain	CH	Chain	CH	Chain	CH	Chain	CH	Chain	CH	Chain	CH
Asia	*			*	*		*		*			*
Credit		*	*		*			*		*		*
Tank		*	*			*	*			*	*	
Car		*		*		*		*	*		*	
B	*		*		*		*		*		*	
Boerlaga		*	*		*			*		*		*
Insurance	-	-	-	-	-	-	-	-	-	-	-	-
Alarm	*		*		*		*			*	*	

Table 6.3: Results of FDC Values on Each Benchmark Problem

Instance	**Chain**	**CH**	$FDC(r)$
1	*		-0.00478
2		*	0.00240
3		*	0.00041
4	*		0.00002
5			-0.00211
6	*		0.00033
7		*	-0.00186
8	*		0.00021
9	*		-0.00059
10		*	0.00221
11		*	0.00212
12		*	-0.00288
13	*		0.00105
14		*	0.00628
15	*		-0.00165

Table 6.4: Results of fitness landscape analysis(FDC value, r) and the predict performance of if chain scoring algorithm is better through comparing r to threshold $r_0 = -0.00059$ achieved on specific structure problems.

	Structure	FDC(r)	Chain	TN	FP	FN	TP
Group 2	Asia	0.00183		1	0	0	0
	Credit	-0.00179	*	0	0	0	1
	Tank	-0.00117		0	1	0	0
	Car	0.00161	*	0	0	1	0
	B	-0.00063	*	0	0	0	1
	Boerlaga	-0.00064		0	1	0	0
	Alarm	-0.00759	*	0	0	0	1
Group 3	Asia	-0.00017		0	0	0	1
	Credit	-0.00320	*	0	1	0	0
	Tank	-0.00072		1	0	0	0
	Car	0.00113	*	1	0	0	0
	B	-0.00483	*	0	0	0	1
	Boerlaga	-0.00470		0	1	0	0
	Alarm	-0.01153	*	0	0	0	1
Group 4	Asia	-0.00247		0	0	0	1
	Credit	0.00237	*	1	0	0	0
	Tank	-0.00035		0	1	0	0
	Car	0.00385	*	1	0	0	0
	B	-0.00063	*	0	0	0	1
	Boerlaga	0.00021		1	0	0	0
	Alarm	-0.00282	*	0	0	0	1
Group 5	Asia	-0.00471		0	0	0	1
	Credit	0.00179	*	0	0	1	0
	Tank	0.00465		0	0	1	0
	Car	-0.00113	*	0	1	0	0
	B	-0.00063	*	0	0	0	1
	Boerlaga	0.00113		1	0	0	0
	Alarm	0.00619	*	0	0	1	0
Group 6	Asia	0.00378		0	0	1	0
	Credit	0.00398	*	1	0	0	0
	Tank	-0.00369		0	1	0	0
	Car	-0.00240	*	0	1	0	0
	B	-0.00573	*	0	0	0	1
	Boerlaga	0.00128		1	0	0	0
	Alarm	-0.00573	*	0	0	0	1

Table 6.5: Comparison of Accuracy on Three Experiments

	TN	**FP**	**FN**	**TP**	**T**	**N**
Group 2	1	2	1	3	4	3
Group 3	2	2	0	3	3	4
Group 4	3	1	0	3	3	4
Group 5	1	1	3	2	5	2
Group 6	2	2	1	2	3	4
Total	9	8	5	13	18	17

Table 6.6: Results of Testing Experiments on 35 Structures

	Accuracy	**FPR**	**TPR**
ROC	0.7333	0.3750	0.8571
Test Set with Hyperheuristic	0.6285	0.4705	0.7222
Test Set with Random Selection	05038	0.4592	0.4944

Chapter 7

Conclusions and Future Work

This chapter outlines the contributions and draws the conclusions of the thesis from the previous chapters. This is followed by the possible future work which could continue from this research.

7.1 General Conclusions

In this thesis, we have developed two novel approaches based on Ant Colony Optimization for Bayesian network structure learning. Both of them evolved optimal structures on the search space with node orderings. The empirical results on eight standard benchmark problems have described that the performance of these two approaches are problem dependent.

We have investigated a number of themes related to these approaches, including the scoring function (chain score and CH score) and the heuristic algorithms (GA and ACO). We have defined node juxtaposition distribution to all topological node orderings with specific algorithms. The application of this technique helps us to gain a deeper understanding of the capability of chain model structure to achieve the final best node ordering.

We have been able to develop a measure of the fitness landscape analysis with two topological node orderings to evaluate the performance of approaches proposed. We

have defined random bubble swapping in search space for producing time sequences of fitness. We can now describe the chain model structure that can produce more Tarjan kind of topological ordering, which can induce a better value of FDC than the Kahn kind of topological ordering. This result has given us inspiration for selecting a suitable algorithm for unseen problem structure learning.

We have extended the application of fitness landscape analysis for randomly generated node orderings. This is a relatively cheap process with chain model structure. We have also been able to build off-line hyperheuristic models for unseen Bayesian network structure learning from the results of fitness landscape analysis. We have utilized the receiver operating characteristic to select the threshold value for controlling the accuracy of prediction and the balance of TPR and FPR.

In general, we have been able to develop novel approaches for Bayesian network structure learning problems. The comparative analysis with node juxtaposition distribution and fitness landscape analysis to these algorithms have verified the characteristics of these new algorithms. These results have provided a strong basis for further research on BN structure learning problems.

7.2 Main Contributions

The main contributions in this research are:

1. **Review of BN structure learning and fitness landscape analysis**: A review of algorithms incorporating ACO probabilistic techniques and fitness landscape analysis model has been presented.

2. **Extension of Chain model BN structure learning with ACO**: The ChainACO approach has been proposed, and experiments have been executed with ChainACO.

3. **Extension of K2 heuristic algorithm for BN structure learning with ACO**: The K2ACO approach has been proposed, and experiments have been executed with

ChainACO.

4. **Comparative analysis between Chain model and K2 heuristic search and score algorithms**: A comparative analysis using node juxtaposition distribution between Chain model algorithms and K2 model algorithms with GA and ACO approaches.

5. **Introduction of fitness landscape analysis with topological node ordering**: An deep investigation has been carried out on Chain model score and CH score with GA and ACO search approaches.

6. **Introduction of receiver operating characteristics**: The ROC technique has been introduced for building classifier models with chain model approaches for prediction of algorithm selection.

7. **Introduction of data generating on modifying conditional probability**: The new data have been generated with minor modification of conditional probability for producing new comparative benchmark problems.

8. **Applications**: Chain model and K2 based algorithms have been applied to a number of benchmark problems and real world problems in oil rig data.

9. **Off-line hyperheuristic Bayesian network structure learning**: This off line hyperheuristic algorithm has been constructed for unseen problem BN structure learning.

7.3 Future Work

The main possible distributions of future work are:

7.3.1 Different score functions

From the descriptions in Chapter 2 and Chapter 3, we know that the K2 algorithm is one of the most successful algorithms for BN structure learning. The basic idea

of K2 algorithm is to calculate the joint probability of a BN structure given data, accompanied by a topology node ordering by maximizing the joint probability. The score function applied for measuring the joint probability of a BN structure with data plays a key rule in K2 algorithm.

In Chapter 2, we reviewed some different score metrics. All these metrics are proposed with some assumptions of the network parameter distribution. Some researchers have studied the performance of these score metrics with K2-like algorithms [Yang and Chang, 2002]. Experimental results show there exist differences among these metrics with K2 algorithm.

Our further research is interested to explore the two novel ACO based approaches with more score metrics. They are MDL [Buntine, 1991], BDeU [Buntine, 1991], BIC [Schwarz, 1978] and AIC [Akaike, 1970] scores. The objective is to investigate (1) Which score metric would be expected to generate the best performance with chain model structure and K2 structure respectively? (2) Is it possible to deduce the relationship with score metrics and data size in benchmark problems?

7.3.2 Deeply investigate Chain model structure on BN structure learning

The results presented in Chapter 3 illustrate that the chain model structure for BN structure learning can achieve better joint probability in some of the benchmark problems. It performs cheaper in all of these cases, both with GA and ACO algorithms. The chain structure model is a kind of topological linear node orderings. Both Tarjan and Kahn node orderings described in Chapter 5 two typical ordering. They imply the best node ordering for K2 greedy search algorithm to learn the best BN.

It would also be interesting to deeply investigate the characteristics of Tarjan and Kahn ordering in standard problems. This would reveal some linkages and correlation between nodes from statistical analysis to known BN structure. The results can give us inspiration on designing the existing chain structure model. We expect it can provide constraints on score function, help us to adapt and update our chain structure model

as a multi-objective optimization problem.

7.3.3 Improving the efficiency of hyperheuristic on BN structure learning

The hyperheuristic algorithm proposed in this thesis has provided us with a theoretical application for unseen BN structure learning problems. This algorithm is based on the characteristics of fitness landscape analysis. In order to improve the efficiency of the hyperheuristic algorithm, we still need to deeply investigate the performance of fitness landscape analysis on BN structures. This investigation can provide a more accurate prediction of problem difficulties. Except for the local properties and global properties we have utilized in this thesis, the further study with fitness landscape analysis will focus on two new properties [Watson, 2000] [Mattfeld and Bierwirth, 1999]:

Distance between solutions: this distance is defined as an operator-independent distance measure. It uses the normalized disjunctive graph measure $D(s_i, s_j)$ to determine the distance between two arbitrary solutions s_i and s_j. The average distance in the solution pool reflects the hard or easy property of the specific problems.

Solutions Entropy: [Mattfeld and Bierwirth, 1999] defined solutions entropy in 1999 for Job-shop problems. The entropy is an another measure in fitness landscape for capturing the solutions commonality. When the entropy $E = 1$, it means a completely random set of solutions and $E = 0$ represents a completely homogeneous set of solutions.

We are able to compare these new properties with the previous properties, to produce balanced evaluation on the performance of each algorithm.

References

[Abramson et al., 1996] Abramson, B.; Brown, J.; Edwards, W.; Murphy, A.; and Winkler, R. L. (1996). "Hailfinder: A Bayesian system for forecasting severe weather". *International Journal of Forecasting*, 12(1), pp. 57–71.

[Acid and de Campos, 2003] Acid, S. and de Campos, L. M. (2003). "Searching for Bayesian Network Structures in the Space of Restricted Acyclic Aprtially Directed Graphs". *Journal of Artificial Intelligence Research*, 18, pp. 445–490.

[Akaike, 1970] Akaike, H. (1970). "Statistical predictor identification". In *Annals of the Institute of Statistical Mathematics*, volume 22(1), pages 203–217.

[Aldrich, 1997] Aldrich, J. (1997). "R. A. Fisher and the Making of Maximum Likelihood 1912-1922". *Statistical Science*, 12(3), pp. 162–176.

[Angel and Zissimopoulos, 1998] Angel, E. and Zissimopoulos, V. (1998). "Autocorrelation Coefficient for the Graph Bipartitioning Problem". *Theoretical Computer Science*, 191, pp. 229–243.

[Arge et al., 2003] Arge, L.; Toma, L.; and Zeh, N. (2003). "I/O-Efficient Topological Sorting of Planar DAGs". In *In Proceedings of the 15th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 85–93.

[Bader-El-Den and Poli, 2008] Bader-El-Den, M. and Poli, R. (2008). "Generating SAT local-search heuristics using a GP hyper-heuristic framework". In *Proceedings of the*

Evolution artificielle, 8th international conference on Artificial evolution, EA'07, pages 37–49, Berlin, Heidelberg. Springer-Verlag.

[Battiato et al., 2010] Battiato, S.; Farinella, G. M.; Giuffrida, G.; Sismeiro, C.; and Tribulato, G. (2010). "Exploiting visual and text features for direct marketing learning in time and space constrained domains". *Pattern Analysis and Applications*, 13, pp. 143–157.

[Beinlich et al., 1989] Beinlich, I. A.; Suermondt, H. J.; Chavez, R. M.; and Cooper, G. F. (1989). "The ALARM Monitoring System: A Case Study with Two Probablistic Inference Techniques for Belief Networks". In *Proc. Second European Conf. Artificial Intelligence in Medicine*, pages 247–256.

[Bianchi et al., 2002] Bianchi, L.; Gambardella, L. M.; and Dorigo, M. (2002). "An Ant Colony Optimization Approach to the Probabilistic Traveling Salesman Problem.". In *PPSN'02*, pages 883–892.

[Bilgin et al., 2006] Bilgin, B.; Özcan, E.; and Korkmaz, E. E. (2006). "An experimental study on hyper-heuristics and exam timetabling". In *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling*, pages 123–140.

[Binder et al., 1997] Binder, J.; Koller, D.; Russell, S. J.; and Kanazawa, K. (1997). "Adaptive Probabilistic Networks with Hidden Variables". *Machine Learning*, 29, pp. 213–244.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Blanco et al., 2003] Blanco, R.; Inza, I.; and Larrañaga, P. (2003). "Learning Bayesian networks in the space of structures by estimation of distribution algorithms". *International Journal of Intelligent Systems*, 18, pp. 205–220.

[Boerlage, 1992] Boerlage, B. (1992). "Link Strength in Bayesi an Networks". *MSc thesis, Department of Computer Science, University of British Columbia, Canada.*

[Borgelt et al., 2009] Borgelt, C.; Kruse, R.; and Steinbrecher, M. (2009). *Graphical Models: Representations for Learning, Reasoning and Data Mining*. Wiley, 2nd edition.

[Bradley, 1997] Bradley, A. P. (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms". *Pattern Recognition*, 30, pp. 1145–1159.

[Brouard et al., 2008] Brouard, T.; A.Delaplace; and H.Cardot (2008). "Evolutionary Methods for Learning Bayesian Network Structure". *Advance in Evolutionary Algorithms*, pages 335–360.

[Buntine, 1991] Buntine, W. (1991). In *Theory Refinement on Bayesian Networks*, pages 52–60. Morgan Kaufmann.

[Burke et al., 2006] Burke, E. K.; Petrovic, S.; and Qu, R. (2006). "Case-based heuristic selection for timetabling problems". *J. of Scheduling*, 9, pp. 115–132.

[Campos et al., 2001] Campos, L. M. D.; Fernández-luna, J. M.; and Puerta, J. M. (2001). "Stochastic local search algorithms for learning belief networks: Searching". In *in the space of orderings, Lecture Notes in Artificial Intelligence 2143*, pages 228–239.

[Campos and Huete, 2000] Campos, L. M. D. and Huete, J. F. (2000). "A new approach for learning belief networks using independence criteria". *International Journal of Approximate Reasoning*, 24, pp. 11–37.

[Castelo and Boutilier, 2001] Castelo, R. and Boutilier, C. (2001). "Improved learning of Bayesian networks". In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 269–276. Morgan Kaufmann.

[Cheng et al., 1997] Cheng, J.; Bell, D. A.; and Liu, W. (1997). "Learning belief networks from data: an information theory based approach". In *Proceedings of the sixth*

international conference on Information and knowledge management, CIKM '97, pages 325–331, New York, NY, USA. ACM.

[Chickering, 1996] Chickering, D. (1996). *Learning Bayesian networks is NP-complete*. Springer-Verlag, New York.

[Cicirello and Smith, 2001] Cicirello, V. A. and Smith, S. F. (2001). "Ant Colony Control for Autonomous Decentralized Shop Floor Routing". In *ISADS*, pages 383–390.

[Cooper and Herskovits, 1992] Cooper, G. and Herskovits, E. (1992). "A Bayesian method for the induction of probabilistic networks from data". *Machine Learning*, 9(4), pp. 309–347.

[Cooper, 1990] Cooper, G. F. (1990). "The computational complexity of probabilistic inference using Bayesian belief networks (research note)". *Artif. Intell.*, 42, pp. 393–405.

[Corne and Reynolds, 2011] Corne, D. and Reynolds, A. P. (2011). "Evaluating optimization algorithms: bounds on the performance of optimizers on unseen problems". In *GECCO (Companion)*, pages 707–710.

[Daly and Shen, 2009] Daly, R. and Shen, Q. (2009). "Q.: Learning Bayesian network equivalence classes with ant colony optimization". *Journal of Artificial Intelligence Research*, 35, pp. 391–447.

[de Campos and Puerta, 2002] de Campos, L. M., G.-J. A. and Puerta, J. M. (2002). "Learning Bayesian networks by ant colony optimisation: Searching in two different spaces". *Mathware & Soft Computing*, 9(2-3), pp. 251–268.

[de Campos et al., 2002] de Campos, L. M.; Fernández-Luna, J. M.; Gámez, J. A.; and Puerta, J. M. (2002). "Ant colony optimization for learning Bayesian networks". *Int. J. Approx. Reasoning*, 31(3), pp. 291–311.

[de Campos et al., 2004] de Campos, L. M.; Fernández-Luna, J. M.; and Huete, J. F. (2004). "Bayesian networks and information retrieval: an introduction to the special issue". *Inf. Process. Manage.*, 40(5), pp. 727–733.

[de Campos et al., 2003] de Campos, L. M.; Fernández-Luna, J. M.; and Puerta, J. M. (2003). "An iterated local search algorithm for learning Bayesian networks with restarts based on conditional independence tests". *Int. J. Intell. Syst.*, 18(2), pp. 221–235.

[de Campos and Puerta, 2001] de Campos, L. M. and Puerta, J. M. (2001). "Stochastic Local Algorithms for Learning Belief Networks: Searching in the Space of the Orderings". In *ECSQARU*, pages 228–239.

[Dempster et al., 1977] Dempster, A. P.; Laird, N. M.; and Rubin, D. B. (1977). "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society, Series B*, 39(1), pp. 1–38.

[Diekert and Gastin, 1995] Diekert, V. and Gastin, P. (1995). "The Book of Traces". World Scientific.

[Dorigo and Blum, 2005] Dorigo, M. and Blum, C. (2005). "Ant colony optimization theory: A survey.". *Theor. Comput. Sci.*, pages 243–278.

[Dorigo and Caro, 1999] Dorigo, M. and Caro, G. D. (1999). "The ant colony optimization meta-heuristic". In *in New Ideas in Optimization*, pages 11–32. McGraw-Hill.

[Dorigo et al., 2002] Dorigo, M.; Caro, G. D.; and Sampels, M., editors (2002). *Ant Algorithms, Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002, Proceedings*, volume 2463 of *Lecture Notes in Computer Science*. Springer.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). "Ant colony system: a cooperative learning approach to the traveling salesman problem.". *IEEE Trans. Evolutionary Computation*, pages 53–66.

[Dorigo and Stützle, 2002] Dorigo, M. and Stützle, T. (2002). "The ant colony optimization metaheuristic: Algorithms, applications, and advances". In *Handbook of Metaheuristics*, pages 251–285. Kluwer Academic Publishers.

[Dorigo et al., 1996] Dorigo, M.; Vittorio, M.; and Alberto, C. (1996). "The Ant System: Optimization by a colony of cooperating agents".

[Du et al., 2005] Du, T.; Zhang, S. S.; and Wang, Z. (2005). "Efficient Learning Bayesian Networks using PSO". In *Proceedings of the 2005 international conference on Computational Intelligence and Security - Volume Part I*, CIS'05, pages 151–156, Berlin, Heidelberg. Springer-Verlag.

[Edwards, 2000] Edwards, D. I. (2000). *Introduction to Graphical Modelling*. Springer, 2nd edition.

[Fawcett, 2008] Fawcett, T. (2008). "PRIE: A system for generating rulelists to maximize ROC performance". *Data Mining and Knowledge Discovery*, 17(2), pp. 207—224.

[Field, 2009] Field, A. (2009). *Discovering Statistics Using SPSS (Introducing Statistical Methods series)*, volume Second Edition. Sage Publications Ltd.

[Flach, 2003] Flach, P. (2003). "The geometry of ROC space: understanding machine learning metrics through ROC isometrics". In *Proc. 20th International Conference on Machine Learning (ICML'03)*, pages 194–201. AAAI Press.

[Flach and Wu, 2005] Flach, P. A. and Wu, S. (2005). "Repairing concavities in ROC curves". In Kaelbling, L. P. and Saffiotti, A., editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 702–707. Springer.

[Fournier et al., 2010] Fournier, F. A.; McCall, J. A. W.; Petrovski, A.; and Barclay, P. J. (2010). "Evolved Bayesian Network models of rig operations in the gulf of Mexico.". In *IEEE Congress on Evolutionary Computation'10*, pages 1–7.

[Friedman et al., 2000] Friedman, N.; Linial, M.; Nachman, I.; and Pe'er, D. (2000). "Using Bayesian Networks to Analyze Expression Data.". *Journal of Computational Biology*, pages 601–620.

[Fung and Chang, 1989] Fung, R. and Chang, K.-C. (1989). "Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks". In *Uncertainty in Artificial Intelligence 5 Annual Conference on Uncertainty in Artificial Intelligence (UAI-89)*, pages 209–219, Amsterdam, NL. Elsevier Science.

[Gambardella et al., 2002] Gambardella, L. M.; Dorigo, M.; Middendorf, M.; and Stützle, T. (2002). "Guest editorial: special section on ant colony optimization.". *IEEE Trans. Evolutionary Computation*, pages 317–319.

[GeNIe and SMILE,] GeNIe and SMILE. "GeNIe and SMILE". `http://genie.sis.pitt.edu/about.html`.

[Glymour and Scheinnes, 2000] Glymour, P. S. C. and Scheinnes, R. (2000). *Causality: Models, Reasoning, and Inference (2nd ed.)*. The MIT Press.

[Goldberg, 2002] Goldberg, D. E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA.

[Green and Swets, 1966] Green, D. M. and Swets, J. A. (1966). *Signal detection theory and psychophysics*. Wiley, New York.

[Guntsch and Middendorf, 2001] Guntsch, M. and Middendorf, M. (2001). "Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic T-SP.". In *EvoWorkshops'01*, pages 213–222.

[Guntsch and Middendorf, 2003] Guntsch, M. and Middendorf, M. (2003). "Solving Multi-criteria Optimization Problems with Population-Based ACO". In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 464–478. Springer.

[Hanley and McNeil, 1982] Hanley, J. A. and McNeil, B. J. (1982). "The Meaning and Use of the Area under a Receiver Operating (ROC) Curvel Characteristic". *Radiology*, 143(1), pp. 29–36.

[Hanley, 2008] Hanley, J. B. M. (2008). "A method of comparing the areas under receiver operating characteristic curves derived from the same cases". *Radiology*, 148 (3), pp. 839–84.

[Hartmann and Runkler, 2007] Hartmann, S. A. and Runkler, T. A. (2007). "Online Optimization of a Color Sorting Assembly Buffer Using Ant Colony Optimization.". In *OR'07*, pages 415–420.

[Heckerman, 1999] Heckerman, D. (1999). "A Tutorial on Learning With Bayesian Networks". pages 301–354.

[Heckerman et al., 1995] Heckerman, D.; Geiger, D.; and Chickering, D. M. (1995). "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data". *Machine Learning*, 20(3), pp. 197–243. Available as Technical Report MSR-TR-94-09.

[Henrion, 1988] Henrion, M. (1988). "Propagating uncertainty in Bayesian networks by probabilistic logic sampling". In Lemmer, J. F. and Kanal, L. N., editors, *Uncertainty in Artificial Intelligence*, pages 149–163. Elsevier, Amsterdam, London, New York.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

[Hordijk, 1996] Hordijk, W. (1996). "A measure of landscapes". *Evol. Comput.*, 4, pp. 335–360.

[Imade et al., 2004] Imade, H.; Morishita, R.; Ono, I.; Ono, N.; and Okamoto, M. (2004). "A grid-oriented genetic algorithm framework for bioinformatics". *New Gen. Comput.*, 22, pp. 177–186.

[Javabayes,] Javabayes. "Javabayes". `http:www.cs.cmu.edu/~javabayes/Home/`.

[Jensen, 1996] Jensen, F. (1996). *An introduction to Bayesian networks*. Number v.1. Springer.

[Jones and Forrest, 1995] Jones, T. and Forrest, S. (1995). "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms". In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann.

[Jordan, 1999] Jordan, M. I. (1999). *Learning in Graphical Models*. Chapman and Hall.

[Kabli et al., 2007] Kabli, R.; Herrmann, F.; and McCall, J. (2007). "A chain-model genetic algorithm for Bayesian network structure learning". In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1264–1271, New York, NY, USA. ACM.

[Kabli et al., 2008] Kabli, R.; John, M.; Frank, H.; and Eng, O. (2008). "Evolved Bayesian Networks as a Versatile Alternative to Partin Tables for Prostate Cancer Management". In *GECCO*, pages 1547–1554.

[Kaelbling et al., 1996] Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. (1996). "Reinforcement learning: a survey". *Journal of Artificial Intelligence Research*, 4, pp. 237–285.

[Kahn, 1962] Kahn, A. B. (1962). "Topological sorting of large networks". *Commun. ACM*, 5, pp. 558–562.

[Kendall, 2001] Kendall, P. C. G. (2001). "A hyperheuristic approach to scheduling a sales summit". In *LNCS2079*, pages 176–190.

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

[Korb and Nicholson, 2010] Korb, K. and Nicholson, A. (2010). *Bayesian Artificial Intelligence*. Chapman and Hall, 2nd edition.

[Kozlov and Singh, 1996] Kozlov, A. V. and Singh, J. P. (1996). "Parallel Implementations of Probabilistic Inference". *IEEE Computer*, 29, pp. 33–40.

[Larrañaga et al., 1996] Larrañaga, P.; Kuijpers, C. M. H.; Murga, R. H.; and Yurramendi, Y. (1996). "Learning Bayesian Network Structures by Searching For the Best Ordering With Genetic Algorithms". *IEEE Transactions on Systems, Man and Cybernetics*, 26, pp. 487–493.

[Larrañaga et al., 1994] Larrañaga, P.; Poza, M.; Yurramendi, Y.; Murga, R. H.; and Kuijpers, C. M. H. (1994). "Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18, pp. 912–926.

[Lauritzen and Spiegelhalter, 1988] Lauritzen, S. L. and Spiegelhalter, D. J. (1988). "Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion)". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 50(2), pp. 157–224.

[Lucas et al., 2004] Lucas, P. J. F.; van der Gaag, L. C.; and Abu-Hanna, A. (2004). "Bayesian networks in biomedicine and health-care". *Artificial Intelligence in Medicine*, 30(3), pp. 201–214.

[Manderick et al., 1991] Manderick, B.; de Weger, M.; and Spiessens, P. (1991). "The Genetic Algorithm and the Structure of the Fitness Landscape". In *Proc. of the Fourth International Conference on Genetic Algorithms*, pages 1143–150, San Diego, CA.

[Margaritis, 2003] Margaritis, D. (2003). *Learning Bayesian Network Model Structure from Data*. PhD thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA. Available as Technical Report CMU-CS-03-153.

[Marín-Blázquez and Schulenburg, 2007] Marín-Blázquez, J. G. and Schulenburg, S. (2007). "A hyper-heuristic framework with XCS: learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients". In *Proceedings of the 2003-2005 international conference on Learning classifier systems*, IWLCS'03-05, pages 193–218, Berlin, Heidelberg. Springer-Verlag.

[Martin et al., 1997] Martin, A.; Doddington, G.; Kamm, T.; Ordowski, M.; and Przybocki, M. (1997). "The DET curve in assessment of detection task performance". pages 1895–1898.

[Mattfeld and Bierwirth, 1999] Mattfeld, D. C. and Bierwirth, C. (1999). "A Search Space Analysis of the Job Shop Scheduling Problem". *Annals of Operations Research*, 86, pp. 441–453.

[Merz and Freisleben, 2000] Merz, P. and Freisleben, B. (2000). "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem". *IEEE Transactions on Evolutionary Computation*, 4, pp. 337–352.

[Metz, 1978] Metz, C. E. (1978). "Basic principles of ROC analysis.". *Seminars in nuclear medicine*, 8(4), pp. 283–298.

[Minaei-Bidgoli and Punch, 2003] Minaei-Bidgoli, B. and Punch, W. F. (2003). "Using genetic algorithms for data mining optimization in an educational web-based system". In *Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII*, GECCO'03, pages 2252–2263, Berlin, Heidelberg. Springer-Verlag.

[Mitchell et al., 1992] Mitchell, M.; Forrest, S.; and Holland, J. H. (1992). *The royal road for genetic algorithms: Fitness landscapes and GA performance*, volume 1001. MIT Press.

[Murphy, 1998] Murphy, K. (1998). "A Brief Introduction to Graphical Models and Bayesian Networks".

[Murphy, 2001] Murphy, K. (2001). "An introduction to graphical models". `http://www.ai.mit.edu/~murphyk/Papers/intro_gm.pdf`.

[Muruzabal and Cotta, 2004] Muruzabal, J. and Cotta, C. (2004). "C.: A primer on the evolution of equivalence classes of Bayesian-Network structures". In *Parallel Problem Solving from Nature VIII. Volume 3242 of Lecture Notes in Computer Science*, pages 612–621. Springer-Verlag.

[Naik and Murthy, 2006] Naik, S. K. and Murthy, C. A. (2006). "Standardization of edge magnitude in color images". *IEEE Transactions on Image Processing*, 15(9), pp. 2588–2595.

[Neapolitan, 1990] Neapolitan, R. E. (1990). *Probabilistic reasoning in expert systems: theory and algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

[Neil et al., 2008] Neil, M.; Marquez, D.; and Fenton, N. (2008). "Using Bayesian Networks to Model the Operational Risk to Information Technology Infrastructure in Financial Institutions". *Journal of Financial Transformation*, 22, pp. 131–138.

[Nikovski, 2000] Nikovski, D. (2000). "Constructing Bayesian Networks for Medical Diagnosis from Incomplete and Partially Correct Statistics.". *IEEE Trans. Knowl. Data Eng.*, 12(4), pp. 509–516.

[Obuchowski, 2003] Obuchowski, N. A. (2003). "Receiver operating characteristic curves and their use in radiology.". *Radiology*, 229(1), pp. 3–8.

[ODS-Petrodata,] ODS-Petrodata. "ODS Petrodata Ltd.". http://rigpoint.ods-petrodata.com/.

[Özcan et al., 2008] Özcan, E.; Bilgin, B.; and Korkmaz, E. E. (2008). "A comprehensive analysis of hyper-heuristics". *Intell. Data Anal.*, 12, pp. 3–23.

[Parpinelli et al., 2002] Parpinelli, R. S.; Lopes, H. S.; and Freitas, A. A. (2002). "Data Mining with an Ant Colony Optimization Algorithm". *IEEE Transactions on Evolutionary Computation*, 6, pp. 321–332.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Pearl, 2000] Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge U.P.

[Pinto et al., 2008a] Pinto, P. C.; Nägele, A.; Dejori, M.; Runkler, T. A.; and da Costa Sousa, J. M. (2008a). "Learning of Bayesian networks by a local discovery ant colony algorithm". In *IEEE Congress on Evolutionary Computation*, pages 2741–2748.

[Pinto et al., 2008b] Pinto, P. C.; Nägele, A.; Dejori, M.; Runkler, T. A.; and da Costa Sousa, J. M. (2008b). "Learning of Bayesian networks by a local discovery ant colony algorithm.". In *IEEE Congress on Evolutionary Computation'08*, pages 2741–2748.

[Pinto et al., 2009] Pinto, P. C.; Nägele, A.; Dejori, M.; Runkler, T. A.; and da Costa Sousa, J. M. (2009). "Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning". *IEEE Trans. Evolutionary Computation*, 13(4), pp. 767–779.

[Reeves, 2000] Reeves, C. R. (2000). *Fitness Landscapes and Evolutionary Algorithms*, volume 1829. Springer Berlin Heidelberg.

[Robinson, 1973] Robinson, R. W. (1973). "Counting labeled acyclic digraphs". In *New Directions in the Theory of Graphs*, pages 239–273. Academic Press.

[Romero et al., 2004] Romero, T.; Larrañaga, P.; and Sierra, B. (2004). "Learning Bayesian Networks In The Space Of Orderings With Estimation Of Distribution Algorithms.". *IJPRAI*, pages 607–625.

[Ross and Marín-Blázquez, 2005] Ross, P. and Marín-Blázquez, J. G. (2005). "Constructive hyper-heuristics in class timetabling". In *Congress on Evolutionary Computation*, pages 1493–1500.

[Ross and Hart, 2002] Ross, Peter, S.-S. M.-B. J. G. and Hart, E. (July 2002). "Hyper-heuristics: learning to combine simple heuristics in bin-packing problems". In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 942–948, New York.

[Rothlauf, 2006] Rothlauf, F. (2006). *Representations for genetic and evolutionary algorithms*. Springer.

[Rudini M. Sampaio and Silva., 2008] Rudini M. Sampaio, Felipe L. Valentim, L. A. S. and Silva., R. M. A. (2008). "Inference Algorithms for Systems of Medical Diagnosis Aid Based on Bayesian Networks.". *INFOCOMP Journal of Computer Science*, 7(2), pp. 90–96.

[Schwarz, 1978] Schwarz, G. (1978). "Estimating the Dimension of a Model". *Annals of Statistics*, 6, pp. 461–464.

[Shachter and Peot, 1990] Shachter, R. D. and Peot, M. A. (1990). "Simulation Approaches to General Probabilistic Inference on Belief Networks". In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '89, pages 221–234, Amsterdam, The Netherlands. North-Holland Publishing Co.

[Shibata et al., 2011] Shibata, K.; Nakano, H.; and Miyauchi, A. (2011). "A learning method for dynamic Bayesian network structures using a multi-objective particle swarm optimizer". *Artif. Life Robot.*, 16(3), pp. 329–332.

[Shmygelska et al., 2002] Shmygelska, A.; Aguirre-Hernndez, R.; and Hoos, H. H. (2002). "An Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem". In *in Proc. 16th Canad. Conf. Artif. Intell*, pages 400–417. Springer-Verlag.

[Slaný and Sekanina, 2007] Slaný, K. and Sekanina, L. (2007). "Fitness Landscape Analysis and Image Filter Evolution Using Functional-Level CGP". In *European Conference on Genetic Programming*, pages 311–320.

[Spackman, 1989] Spackman, K. A. (1989). "Signal detection theory: Valuable tools for evaluating inductive learning". In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 160–163, San Mateo, CA. Morgan Kaufman.

[Stadler, 1995] Stadler, P. F. (1995). "Landscapes and Their Correlation Functions". Working Papers 95-07-067, Santa Fe Institute.

[Stigler, 1986] Stigler, S. M. (1986). *The history of statistics*. Harvard University press.

[Stützle and Hoos, 2000] Stützle, T. and Hoos, H. H. (2000). "MAX-MIN Ant System.". *Future Generation Comp. Syst.*, pages 889–914.

[Swets et al., 2000] Swets, J. A.; Dawes, R. M.; and Monahan, J. (2000). "Better decisions through science.". *Scientific American*, 283(4), pp. 82–87.

[Tarjan, 1976] Tarjan, R. E. (1976). "Edge-Disjoint Spanning Trees and Depth-First Search". *Acta Informatica*, 6, pp. 171–185.

[Terashima-Marín et al., 2005] Terashima-Marín, H.; Flores-Álvarez, E. J.; and Ross, P. (2005). "Hyper-heuristics and classifier systems for solving 2D-regular cutting stock problems". In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 637–643, New York, NY, USA. ACM.

[Thomas H. Cormen, 2001] Thomas H. Cormen, Clifford Stein, R. L. R. C. E. L. (2001). *Introduction to Algorithms*. MIT Press and McGraw-Hill.

[Tsamardinos et al., 2006a] Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. (2006a). "The max-min hill-climbing Bayesian network structure learning algorithm". *Machine Learning*, 65, pp. 31–78.

[Tsamardinos et al., 2006b] Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. (2006b). "The max-min hill-climbing Bayesian network structure learning algorithm". In *Machine Learning*, page 2006.

[Tsutsui, 2008] Tsutsui, S. (2008). "Parallel Ant Colony Optimization for the Quadratic Assignment Problems with Symmetric Multi Processing.". In *ANTS Conference'08*, pages 363–370.

[Verma and Pearl, 1990] Verma, T. and Pearl, J. (1990). "Equivalence and Synthesis of Causal Models". In *Proceedings of the Uncertainty in Artificial Intelligence 6 Annual Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 255–268, Amsterdam, NL. Elsevier Science.

[Vernet and Markenzon, 1997] Vernet, O. and Markenzon, L. (1997). "Hamiltonian problems for reducible flowgraphs". In *Proceedings of the 17th International Conference of the Chilean Computer Science Society*, SCCC '97, pages 264–272, Washington, DC, USA. IEEE Computer Society.

[Wang et al., 2004] Wang, T.; Touchman, J. W.; and Xue, G. (2004). "Applying two-level simulated annealing on Bayesian structure learning to infer genetic networks". In *In Proceedings of the IEEE Computational Systems Bioinformatics Conference*, pages 647–648.

[Watson, 2000] Watson, J. P. (2000). "Problem difficulty and fitness landscapes of structured and random job-shop problems: What do existing analysis techniques really tell us". Berlin, Germany. ECAI2000 workshop.

[Webb and Ting, 2004] Webb, G. I. and Ting, K. M. (2004). "On the application of ROC analysis to predict classification performance under varying class distributions". In *Machine Learning*, pages 25–32.

[Wegener et al., 1997] Wegener, J.; Sthamer, H.; Jones, B. F.; and Eyres, D. E. (1997). "Testing real-time systems using genetic algorithms". *Software Quality Control*, 6, pp. 127–135.

[Whitley, 1994] Whitley, D. (1994). "A Genetic Algorithm Tutorial". *Statistics and Computing*, 4, pp. 65–85.

[Wong et al., 1999] Wong, M. L.; Lam, W.; and Leung, K. S. (1999). "Using Evolutionary Programming and Minimum Description Length Principle for Data Mining of Bayesian Networks". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, pp. 174–178.

[Wright, 1932] Wright, S. (1932). "The Roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution". *Proceedings of the Sixth International Congress of Genetics*, 1, pp. 356–66.

[Wu et al., 2010a] Wu, Y.; McCall, J. A. W.; and Corne, D. (2010a). "Comparative Analysis of Search and Score Metaheuristics for Bayesian Network Structure Learning Using Node Juxtaposition Distributions". In *PPSN (1)*, pages 424–433.

[Wu et al., 2010b] Wu, Y.; McCall, J. A. W.; and Corne, D. W. (2010b). "Two novel Ant Colony Optimization approaches for Bayesian network structure learning.". In *IEEE Congress on Evolutionary Computation'2010*, pages 1–7.

[Yang and Chang, 2002] Yang, S. and Chang, K.-C. (2002). "Comparison of score metrics for Bayesian network learning". *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 32(3), pp. 419–428.

Appendix A

Figures- Comparison of search and score algorithms on benchmark structures

The figures in this chapter relate to the experiments discussed in Chapter 3. In Figure A.9 ~ Figure A.16, the labels '1'–'4' in each figure represents the algorithm ChainACO, K2ACO, ChainGA and K2GA respectively.

Multiple Comparisons

Score

Bonferroni

(I) Approach	(J) Approach	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	39.92207*	10.33100	.003	11.0782	68.7659
	3	-4.17383	10.33100	1.000	-33.0177	24.6700
	4	-7.26680	10.33100	1.000	-36.1107	21.5771
2	1	-39.92207*	10.33100	.003	-68.7659	-11.0782
	3	-44.09590*	10.33100	.001	-72.9398	-15.2521
	4	-47.18887*	10.33100	.000	-76.0327	-18.3450
3	1	4.17383	10.33100	1.000	-24.6700	33.0177
	2	44.09590*	10.33100	.001	15.2521	72.9398
	4	-3.09297	10.33100	1.000	-31.9368	25.7509
4	1	7.26680	10.33100	1.000	-21.5771	36.1107
	2	47.18887*	10.33100	.000	18.3450	76.0327
	3	3.09297	10.33100	1.000	-25.7509	31.9368

*. The mean difference is significant at the 0.05 level.

Figure A.1: ANOVA test using the Bonferroni correction on Asia structure

Multiple Comparisons

Score

Bonferroni

(I) Approach	(J) Approach	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	24.66699	29.13434	1.000	-57.2688	106.6028
	3	-140.21133*	29.13434	.000	-222.1471	-58.2756
	4	-139.17539*	33.64143	.001	-233.7867	-44.5641
2	1	-24.66699	29.13434	1.000	-106.6028	57.2688
	3	-164.87832*	29.13434	.000	-246.8141	-82.9426
	4	-163.84238*	33.64143	.000	-258.4537	-69.2311
3	1	140.21133*	29.13434	.000	58.2756	222.1471
	2	164.87832*	29.13434	.000	82.9426	246.8141
	4	1.03593	33.64143	1.000	-93.5753	95.6472
4	1	139.17539*	33.64143	.001	44.5641	233.7867
	2	163.84238*	33.64143	.000	69.2311	258.4537
	3	-1.03593	33.64143	1.000	-95.6472	93.5753

*. The mean difference is significant at the 0.05 level.

Figure A.2: ANOVA test using the Bonferroni correction on Credit structure

Multiple Comparisons

Score

Bonferroni

(I) Approach	(J) Approach	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	24.66699	27.58671	1.000	-52.3543	101.6883
	3	-140.21133*	27.58671	.000	-217.2326	-63.1900
	4	-135.28652*	27.58671	.000	-212.3078	-58.2652
2	1	-24.66699	27.58671	1.000	-101.6883	52.3543
	3	-164.87832*	27.58671	.000	-241.8996	-87.8570
	4	-159.95352*	27.58671	.000	-236.9748	-82.9322
3	1	140.21133*	27.58671	.000	63.1900	217.2326
	2	164.87832*	27.58671	.000	87.8570	241.8996
	4	4.92480	27.58671	1.000	-72.0965	81.9461
4	1	135.28652*	27.58671	.000	58.2652	212.3078
	2	159.95352*	27.58671	.000	82.9322	236.9748
	3	-4.92480	27.58671	1.000	-81.9461	72.0965

*. The mean difference is significant at the 0.05 level.

Figure A.3: ANOVA test using the Bonferroni correction on Tank structure

Multiple Comparisons

Score

Bonferroni

(I) Appraoch	(J) Appraoch	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	22.80059	17.59687	1.000	-26.3294	71.9306
	3	2.34258	17.59687	1.000	-46.7874	51.4725
	4	5.39517	17.59687	1.000	-43.7348	54.5251
2	1	-22.80059	17.59687	1.000	-71.9306	26.3294
	3	-20.45801	17.59687	1.000	-69.5880	28.6720
	4	-17.40542	17.59687	1.000	-66.5354	31.7245
3	1	-2.34258	17.59687	1.000	-51.4725	46.7874
	2	20.45801	17.59687	1.000	-28.6720	69.5880
	4	3.05259	17.59687	1.000	-46.0774	52.1826
4	1	-5.39517	17.59687	1.000	-54.5251	43.7348
	2	17.40542	17.59687	1.000	-31.7245	66.5354
	3	-3.05259	17.59687	1.000	-52.1826	46.0774

Figure A.4: ANOVA test using the Bonferroni correction on Car structure

Multiple Comparisons

Score

Bonferroni

(I) Approach	(J) Approach	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	92.20156	46.66245	.335	-38.0787	222.4818
	3	-219.95000*	46.66245	.000	-350.2302	-89.6698
	4	-196.11172*	46.66245	.001	-326.3920	-65.8315
2	1	-92.20156	46.66245	.335	-222.4818	38.0787
	3	-312.15156*	46.66245	.000	-442.4318	-181.8713
	4	-288.31328*	46.66245	.000	-418.5935	-158.0330
3	1	219.95000*	46.66245	.000	89.6698	350.2302
	2	312.15156*	46.66245	.000	181.8713	442.4318
	4	23.83828	46.66245	1.000	-106.4420	154.1185
4	1	196.11172*	46.66245	.001	65.8315	326.3920
	2	288.31328*	46.66245	.000	158.0330	418.5935
	3	-23.83828	46.66245	1.000	-154.1185	106.4420

*. The mean difference is significant at the 0.05 level.

Figure A.5: ANOVA test using the Bonferroni correction on B structure

Multiple Comparisons

Score

Bonferroni

(I) Approach	(J) Approach	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	31.05645*	11.05540	.048	.1901	61.9228
	3	-22.74609	11.05540	.282	-53.6125	8.1203
	4	-2.65547	11.05540	1.000	-33.5218	28.2109
2	1	-31.05645*	11.05540	.048	-61.9228	-.1901
	3	-53.80254*	11.05540	.000	-84.6689	-22.9362
	4	-33.71192*	11.05540	.026	-64.5783	-2.8456
3	1	22.74609	11.05540	.282	-8.1203	53.6125
	2	53.80254*	11.05540	.000	22.9362	84.6689
	4	20.09063	11.05540	.465	-10.7757	50.9570
4	1	2.65547	11.05540	1.000	-28.2109	33.5218
	2	33.71192*	11.05540	.026	2.8456	64.5783
	3	-20.09063	11.05540	.465	-50.9570	10.7757

*. The mean difference is significant at the 0.05 level.

Figure A.6: ANOVA test using the Bonferroni correction on Boerlaga structure

Multiple Comparisons

Score

Bonferroni

(I) Approach	(J) Approach	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	311.32617	119.29850	.079	-21.7519	644.4042
	3	-618.39101*	119.29850	.000	-951.4691	-285.3129
	4	-499.61484*	119.29850	.001	-832.6929	-166.5368
2	1	-311.32617	119.29850	.079	-644.4042	21.7519
	3	-929.71719*	119.29850	.000	-1262.7953	-596.6391
	4	-810.94101*	119.29850	.000	-1144.0191	-477.8629
3	1	618.39101*	119.29850	.000	285.3129	951.4691
	2	929.71719*	119.29850	.000	596.6391	1262.7953
	4	118.77617	119.29850	1.000	-214.3019	451.8542
4	1	499.61484*	119.29850	.001	166.5368	832.6929
	2	810.94101*	119.29850	.000	477.8629	1144.0191
	3	-118.77617	119.29850	1.000	-451.8542	214.3019

*. The mean difference is significant at the 0.05 level.

Figure A.7: ANOVA test using the Bonferroni correction on Insurance structure

Multiple Comparisons

Score

Bonferroni

(I) Appraoch	(J) Appraoch	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
1	2	279.43164	124.62003	.187	-68.5040	627.3673
	3	-285.63398	124.62003	.167	-633.5696	62.3016
	4	-217.62754	124.62003	.536	-565.5632	130.3081
2	1	-279.43164	124.62003	.187	-627.3673	68.5040
	3	-565.06562*	124.62003	.000	-913.0012	-217.1300
	4	-497.05918*	124.62003	.002	-844.9948	-149.1236
3	1	285.63398	124.62003	.167	-62.3016	633.5696
	2	565.06562*	124.62003	.000	217.1300	913.0012
	4	68.00644	124.62003	1.000	-279.9292	415.9421
4	1	217.62754	124.62003	.536	-130.3081	565.5632
	2	497.05918*	124.62003	.002	149.1236	844.9948
	3	-68.00644	124.62003	1.000	-415.9421	279.9292

*. The mean difference is significant at the 0.05 level.

Figure A.8: ANOVA test using the Bonferroni correction on Alarm structure

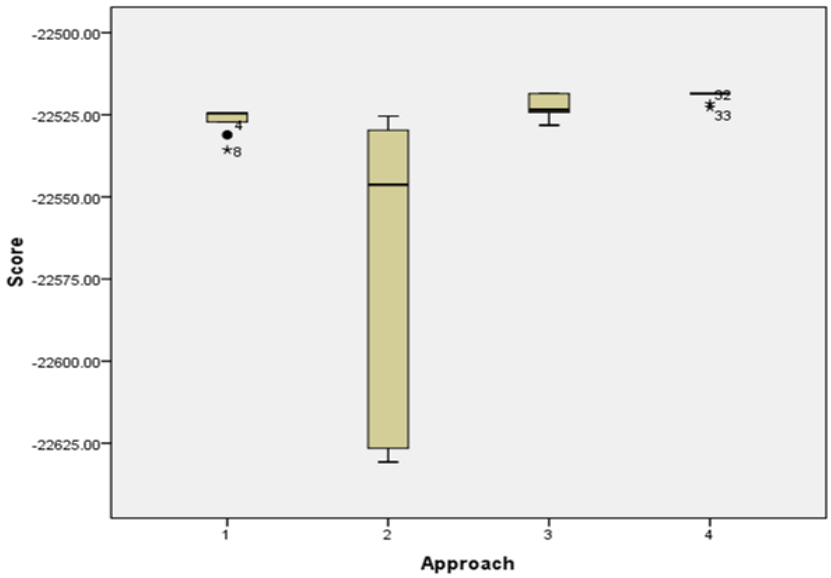


Figure A.9: Box-plots of the Best found Structure Score for Asia at each run for all algorithms

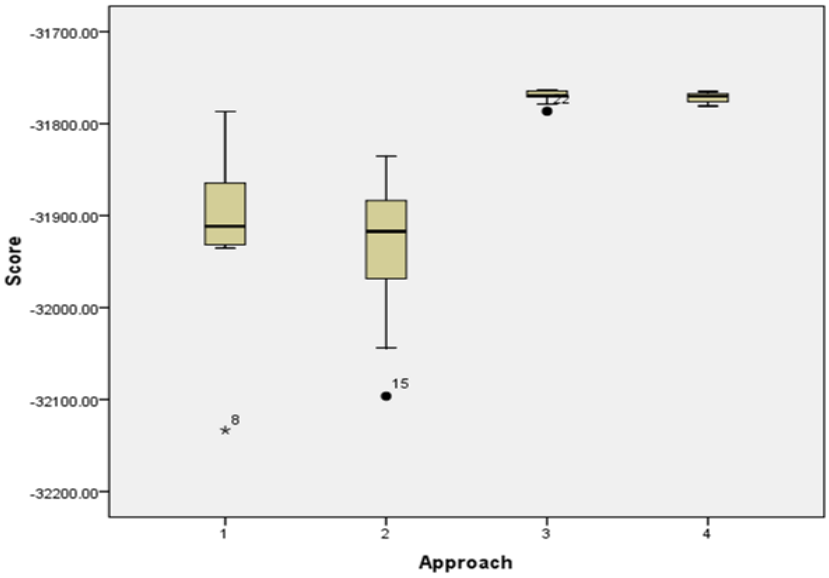


Figure A.10: Box-plots of the Best found Structure Score for Credit at each run for all algorithms

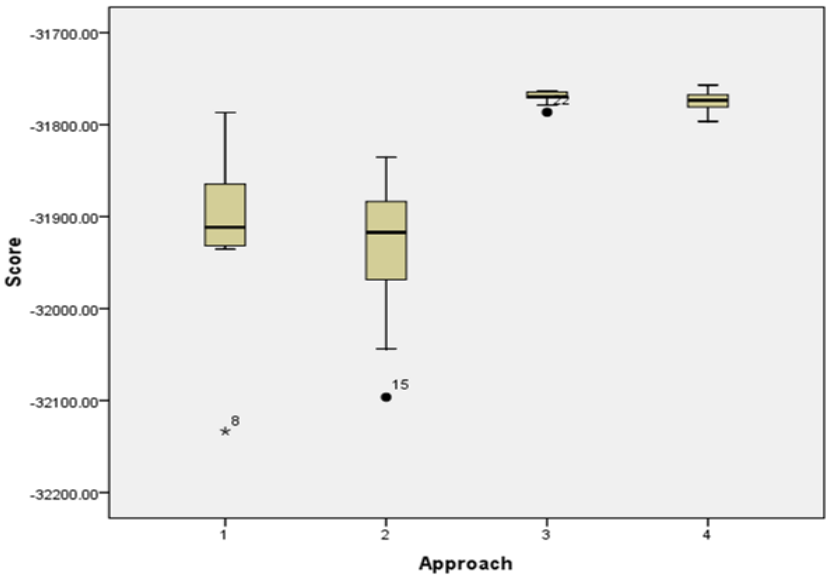


Figure A.11: Box-plots of the Best found Structure Score for Tank at each run for all algorithms

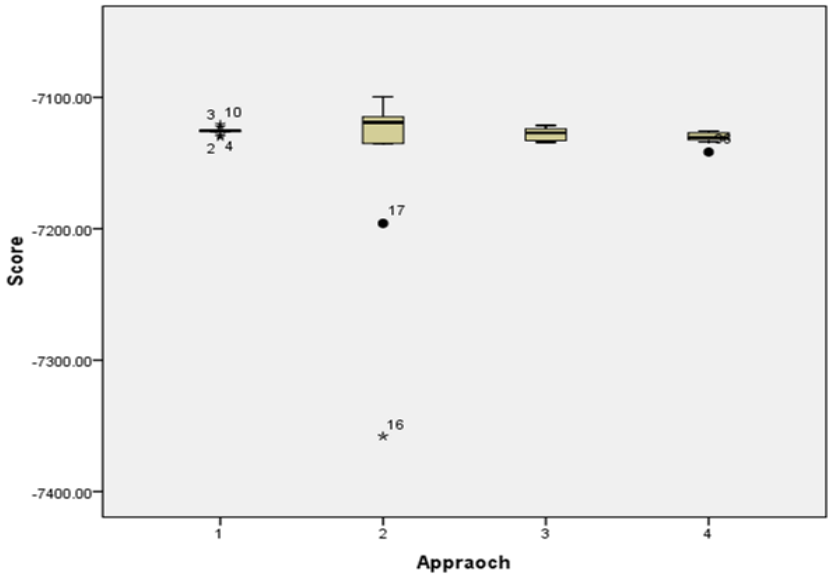


Figure A.12: Box-plots of the Best found Structure Score for Car at each run for all algorithms

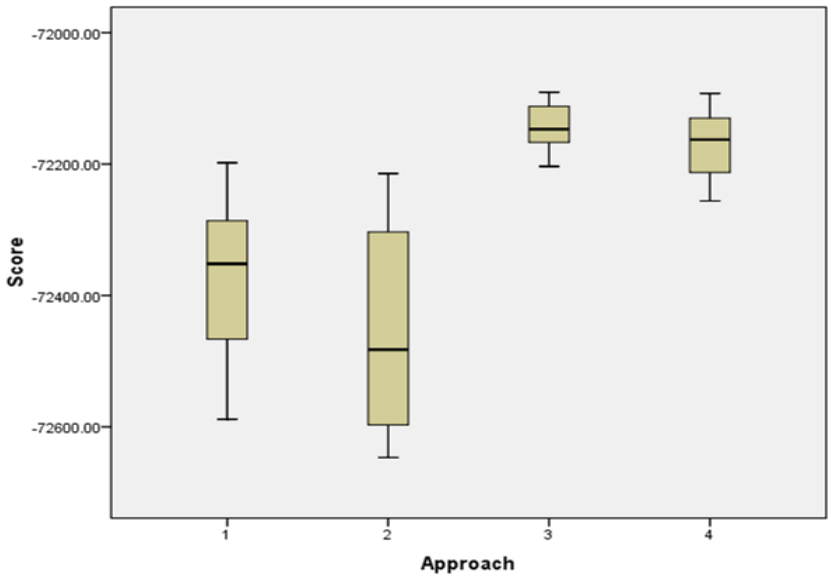


Figure A.13: Box-plots of the Best found Structure Score for B at each run for all algorithms

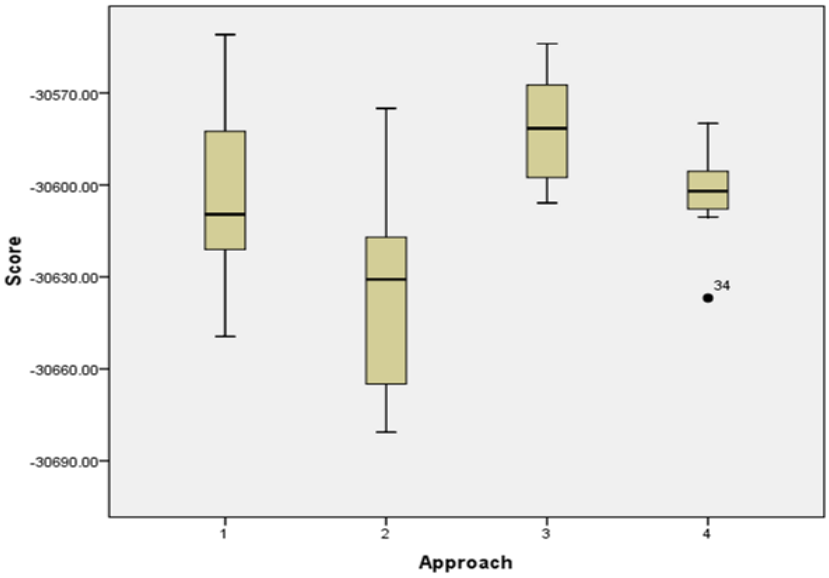


Figure A.14: Box-plots of the Best found Structure Score for Boerlaga at each run for all algorithms

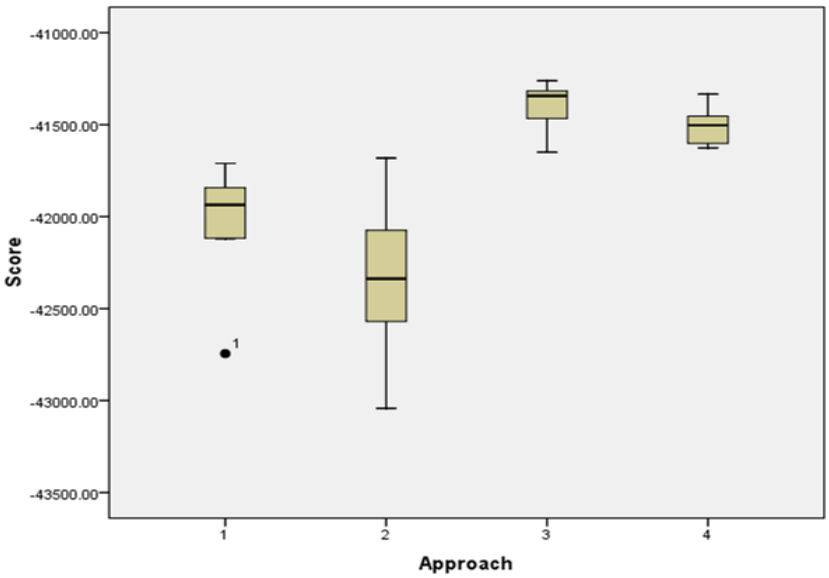


Figure A.15: Box-plots of the Best found Structure Score for Insurance at each run for all algorithms

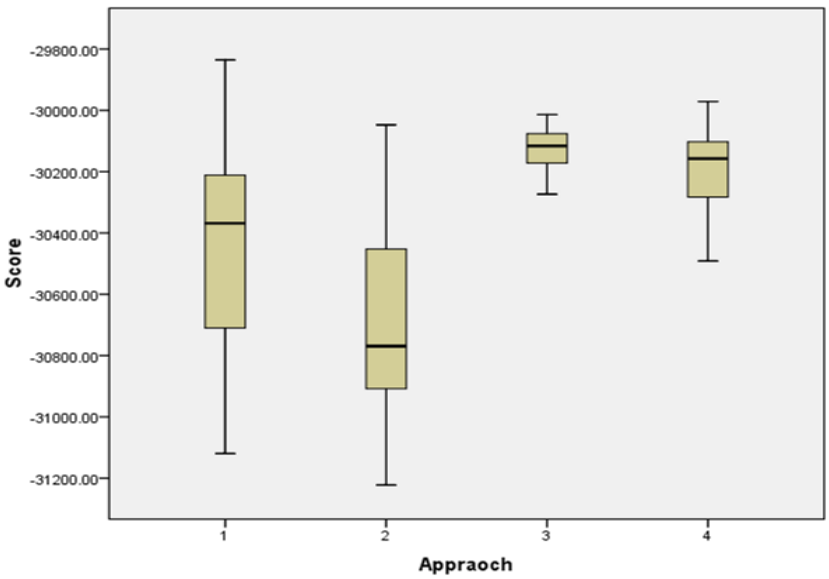


Figure A.16: Box-plots of the Best found Structure Score for Alarm at each run for all algorithms