2-2010

Learning the Structure of Bayesian Networks with Constraint Satisfaction

Andrew Scott Fast
University of Massachusetts Amherst, andrew.s.fast@gmail.com

LEARNING THE STRUCTURE OF BAYESIAN NETWORKS WITH CONSTRAINT SATISFACTION

A Dissertation Presented

by

ANDREW S. FAST

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2010

Department of Computer Science

LEARNING THE STRUCTURE OF BAYESIAN NETWORKS WITH CONSTRAINT SATISFACTION

A Dissertation Presented

by

ANDREW S. FAST

Approved as to style and content by:

David Jensen, Chair

Oliver Brock, Member

Victor Lesser, Member

Andrea Foulkes, Member

Andrew G. Barto, Department Chair
Department of Computer Science

Soli Deo Gloria

ACKNOWLEDGMENTS

This thesis would not have been possible without the kindness and support of David Jensen and Victor Lesser. Together they are responsible for helping me build the foundation on which this thesis is built. Victor gave me my first chance to try research and I am indebted to him for that opportunity. David, my advisor, has been invaluable to me over the years and his encouragement kept me going through the entire process. For this thesis and my research in general, I have adopted David's empirical sensibilities as my own and I hope that this rich heritage is evident throughout my work. David is truly both a gentleman and a scholar and it is a privilege to be his student.

In addition to David and Victor, I am grateful for many others who contributed time and energy towards my journey. I am especially grateful to my other committee members: Oliver Brock and Andrea Foulkes, who were both extremely helpful in providing focus and identifying the weak spots in my arguments and my results. Michael Hay was instrumental in shaping the early stages of this work and provided a knowledgeable sounding board the rest of the way as well. Jennifer Neville provided a consistent reminder that I alone was responsible for seeing this thesis through, while being willing to help in any way she could. I also thank the past and present members of the Knowledge Discovery Laboratory: Lisa Friedland, Marc Maier, Amy McGovern, Huseyin Oktay, Matthew Rattigan and Brian Taylor for being there along the way and helping sharpen the product that you see here. Cindy Loiselle provided more than her fair share of grammar corrections and this document would be wholly incomprehensible without her careful comments. Agustin Schapira and Matthew

Cornell provided expert technical advice at all levels and at nearly every hour of the night and day; I never found a problem they were not able to help me solve.

On a more personal note, Trevor Strohman and Jerod Weinman were only a phone call away when I was ready for a bitter end. Their friendship and encouragement meant a lot over the many rough patches. I am also grateful for Evie and Gram whose prayers and friendship made the last months bearable and warmly welcomed us back to Amherst for my defense. My parents, Loren and Lorraine, helped me foster a curious and scientific mind, laying the ground work for this thesis even from an early age. Finally, this thesis is a product of the continual encouragement and love of my wife Anna and son Peter who helped me to weather the ups and downs of research, and who, through their patience and understanding, helped me see this thesis through to completion.

ABSTRACT

LEARNING THE STRUCTURE OF BAYESIAN NETWORKS WITH CONSTRAINT SATISFACTION

FEBRUARY 2010

ANDREW S. FAST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor David Jensen

A Bayesian network is graphical representation of the probabilistic relationships among set of variables and can be used to encode expert knowledge about uncertain domains. The structure of this model represents the set of conditional independencies among the variables in the data. Bayesian networks are widely applicable, having been used to model domains ranging from monitoring patients in an emergency room to predicting the severity of hailstorms. In this thesis, I focus on the problem of learning the structure of Bayesian networks from data. Under certain assumptions, the learned structure of a Bayesian network can represent causal relationships in the data.

Constraint-based algorithms for structure learning are designed to accurately identify the structure of the distribution underlying the data and, therefore, the causal relationships. These algorithms use a series of conditional hypothesis tests to learn independence constraints on the structure of the model. When sample size is limited, these hypothesis tests are prone to errors. I present a comprehensive empirical

evaluation of constraint-based algorithms and show that existing constraint-based algorithms are prone to many false negative errors in the constraints due to running hypothesis tests with low statistical power. Furthermore, this analysis shows that many statistical solutions fail to reduce the overall errors of constraint-based algorithms.

I show that new algorithms inspired by constraint satisfaction are able to produce significant improvements in structural accuracy. These constraint satisfaction algorithms exploit the interaction among the constraints to reduce error. First, I introduce an algorithm based on constraint optimization that is sound in the sample limit, like existing algorithms, but is guaranteed to produce a DAG. This new algorithm learns models with structural accuracy equivalent or better to existing algorithms. Second, I introduce an algorithm based constraint relaxation. Constraint relaxation combines different statistical techniques to identify constraints that are likely to be incorrect, and remove those constraints from consideration. I show that an algorithm combining constraint relaxation with constraint optimization produces Bayesian networks with significantly better structural accuracy when compared to existing structure learning algorithms, demonstrating the effectiveness of constraint satisfaction approaches for learning accurate structure of Bayesian networks.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

A Bayesian network is a directed, acyclic, graphical representation of the probabilistic relationships among a group of variables. The network is defined by a set of conditional distributions that can be used to represent the joint probability distribution over the variables. The graphical structure of a Bayesian network model describes, in an understandable, visual manner, which variables have direct influence on other variables under consideration. Consequently, Bayesian networks have long been used to encode expert knowledge about uncertain domains [41].

To augment available expert knowledge, many algorithms have been developed to learn the structure of Bayesian networks from data [16, 18, 25, 44, 85]. When certain assumptions hold, the structure of Bayesian networks has a causal interpretation [75, 76, 87]. If the structure of a causal model is accurate, it can be used to predict the effects of manipulations, called interventions, on the system being studied [75]. Consequently, structure learning of Bayesian networks can be used to identify actionable knowledge, which is also the goal of the field of knowledge discovery from databases (KDD) [32, 34].

The focus of this work is developing structure learning algorithms for Bayesian networks that produce actionable models where the structure of the model accurately captures the causal relationships in the data. A subset of structure learning algorithms, called *constraint-based* algorithms, are designed for this purpose. [75, 76, 87]. These algorithms operate in two independent phases [18, 87]. The first phase, called *constraint identification*, uses a series of conditional hypothesis tests to identify a set

of independence constraints on the structure of the final model. The second phase, called *edge orientation*, merges the learned independence constraints into a fully directed Bayesian network model. While constraint-based approaches are sound in the sample limit [76, 87], at more reasonable sample sizes typically encountered in practice, the hypothesis tests used to identify the constraints are not perfectly accurate, leading to errors in the learned model structure and incorrect causal conclusions.

The primary innovation of this thesis is the development of new algorithms, inspired by constraint satisfaction, that are able to learn more accurate structure of Bayesian network than existing constraint-based algorithms. Like traditional constraint satisfaction algorithms, but in contrast to constraint-based algorithms, constraint satisfaction algorithms for learning the structure of Bayesian networks consider all independence constraints jointly. In the remainder of the thesis, I provide additional motivation for this new approach to structure learning and show how constraint satisfaction techniques can produce learned models with significantly fewer structural errors than existing approaches. Chapters 2 and 3 provide background on Bayesian networks and motivate the constraint-based structure learning paradigm. Chapter 4 presents an empirical evaluation of existing constraint-based algorithms that shows false negative errors as the most prevalent error in constraint identification. Furthermore, that evaluation demonstrates that the majority of the false negative errors are a result of running low-power hypothesis tests and that the existing methods used for controlling false negative errors are not able to adequately account for all factors contributing to low statistical power. Chapter 5 considers statistical approaches for improving the statistical power of constraint identification, including a new approach based on propensity score matching. The empirical results presented there show that there is a fundamental trade-off between false negative and false positive errors and that it is difficult to achieve gains in power without a significant increase in false

positive errors. Therefore, alternative approaches are needed for improving the rate of errors.

Constraint satisfaction provides an alternative approach for reducing errors in structure learning. Constraint satisfaction algorithms use the same independence constraints as existing algorithms, but exploit the interaction among the constraints by considering the constraints jointly instead of independently and by using search in place of deterministic rules. Chapter 6 describes the first algorithm to use constraint satisfaction ideas for structure learning. This algorithm, using a constraint satisfaction strategy called constraint optimization, considers all constraints jointly and is guaranteed to produce a Bayesian network, unlike existing edge orientation algorithms. The models learned using constraint optimization meet or exceed the accuracy of the models learned using the previous approach. Constraint optimization alone is not sufficient to reduce errors as the constraints being satisfied could be incorrect. However, since constraint optimization guarantees that the learned structure is a directed acyclic graph, it enables other constraint satisfaction approaches for structure learning.

Constraint relaxation is a constraint satisfaction approach that allows learned independence constraints to be "relaxed" or corrected in the face of additional evidence. Combined with constraint optimization, constraint relaxation is a powerful way to improve the accuracy of learned structure. Chapter 7 describes the first algorithm for constraint relaxation. This algorithm combines multiple structure learning strategies and unifies constraint identification and edge orientation into a single algorithm. The result is an algorithm that produces models with significantly higher structural accuracy than competing approaches. Implementation of these new constraint satisfaction algorithms are available in the PowerBayes open-source software package, which is described in more detail in Appendix B.

As experimental results throughout the thesis demonstrate, incorporating constraint satisfaction approaches into algorithms for learning Bayesian networks produces significant reductions in error over previous algorithms. The algorithms introduced in this thesis embody two primary strategies for reducing errors in structure learning of Bayesian networks. First, constraint satisfaction permits the reduction in errors by incorporating more available information than existing algorithms during structure learning. The constraint optimization algorithm is the first edge orientation algorithm to consider all independence constraints jointly. Constraint relaxation is the first approach to do model selection based on the constraints and simultaneously unify constraint identification and edge orientation into a single search process. Second, these approaches reduce errors by relying on the combination of different, but complementary, techniques in both phases of structure learning. Additionally, the constraint satisfaction approaches introduced in this thesis provide an extensible platform that can incorporate additional advances as they are discovered.

CHAPTER 2

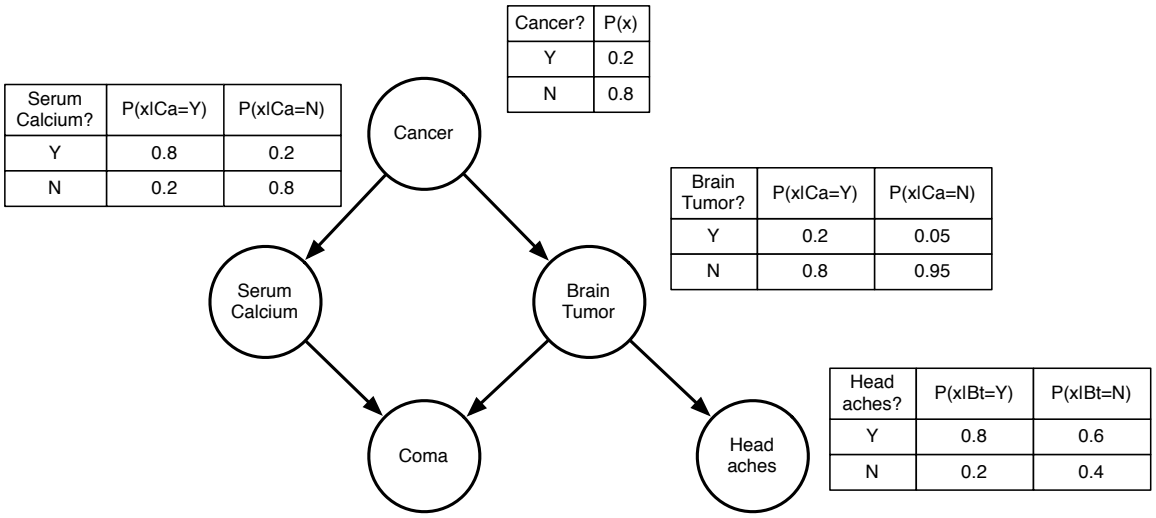
BACKGROUND

2.1 Bayesian Networks

Bayesian networks are a concise, graphical representation of a joint probability distribution P over a set of variables V. A Bayesian network $M = \{G, \Theta\}$ consists of an acyclic, directed graph $G = \{V, E\}$ containing vertices and edges and a set of conditional probability distributions, Θ. The edges of G indicate a dependency between the variables. Throughout this thesis, I will refer to the collection of edges in G as the *structure* of the Bayesian network. The parents of a variable $v \in V$, denoted $\mathbf{pa}(v)$, are the set of variables that are the source of directed edges pointing to the variable v. The children of a variable are all the nodes that are pointed to by edges leaving a variable v, denoted $\mathbf{c}(v)$. The neighbors or adjacent variables of a variable v are $Adj(v) = \mathbf{pa}(v) \cup \mathbf{c}(v)$. For each variable, θ_v is a conditional probability distribution defined as the probability $P(v|\mathbf{pa}(v))$. A Bayesian network is *compatible* with a distribution P if P can be factored according to the parent relations defined by the structure of G [75]. An example of a Bayesian network is shown in Figure 2.1

The structure of a Bayesian network also represents the conditional independence relations among the variables. The correspondence between conditional independence relations and certain graphical structures is summarized by the *d-separation criterion*.

Definition 1. *(from Pearl [75]) A path p is said to be d-separated or (blocked) by a set of nodes Z if and only if*



Cancer?	P(x)
Y	0.2
N	0.8

Serum Calcium?	P(x|Ca=Y)	P(x|Ca=N)
Y	0.8	0.2
N	0.2	0.8

Brain Tumor?	P(x|Ca=Y)	P(x|Ca=N)
Y	0.2	0.05
N	0.8	0.95

Head aches?	P(x|Bt=Y)	P(x|Bt=N)
Y	0.8	0.6
N	0.2	0.4

Coma?	P(x | Sc=Y, Bt = Y)	P(x | Sc=Y, Bt = N)	P(x | Sc=N, Bt = Y)	P(x | Sc=N, Bt = N)
Y	0.8	0.8	0.8	0.8
N	0.2	0.2	0.2	0.2

Figure 2.1: An example Bayesian network modeling the relationship between cancer and common symptoms. Probability distributions are shown in the tables. The example is due to Pearl [74].

1. *p contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in Z, or*

2. *p contains an inverted fork (or collider) $i \to m \leftarrow j$ such that the middle node m is not in Z and such that no descendent of m is in Z.*

A set Z is said to d-separate X from Y if and only if Z blocks every path from a node in X to a node in Y. The definition of d-separation matches an intuitive causal understanding of the edges in the graph.

Pearl [74] gives an alternative, but equivalent, definition of a Bayesian network in terms of d-separation and a dependency model M. A *dependency model* is a set of assertions of the form $(X \perp\!\!\!\perp Y|Z)$ indicating that "X is independent of Y given Z." A DAG D is a Bayesian network of M if and only if every independence assertion in

M corresponds to a valid d-separation relationship in D; that is, the global structure of D is consistent with every local assertion in M.

2.2 Overview of Structure Learning

Given a set of variables V and a dataset D containing independent and identically distributed (i.i.d.) instances sampled from an unknown distribution P the goal of structure learning is to identify a graph G that is compatible with P. To form a complete model M, an additional step of parameter estimation is needed to determine Θ from D. Techniques for parameter estimation are generally well-known and will not be addressed here (see e.g., Heckerman [41] for more details).

The two definitions of Bayesian networks described in Section 2.1 have led to the development of two broad classes of structure learning algorithms. The first approach, called *search-and-score*, searches over possible Bayesian network structures to find the best factorization of the joint distribution implied by the training data [16, 25, 44]. The model selection criterion is usually a penalized likelihood score such as BIC [81], AIC [5], or BDeu [15, 44]. The second approach, *constraint-based* algorithms, learns the structure of a Bayesian network by first running local hypothesis tests to identify a dependency model M containing independence assertions that hold in the training data [18, 75, 87, 96]. The learned independence assertions in M are viewed as *constraints* on the final model structure, and constraint-based algorithms select a model structure that is consistent with those constraints. A third class of algorithms, called hybrid algorithms, combines techniques from both constraint-based and search-and-score techniques [2, 96].

The different categories of algorithms were each designed with a particular purpose in mind. Search-and-score techniques are generally very flexible and find high-likelihood structures but do not enforce conditional independence relationships and often do not accurately reproduce the generating structure [1, 92]. By constraining

the space with conditional independence relations, constraint-based techniques are more efficient and more accurately recover the structure of the generating distribution but often do not achieve comparable likelihoods with search-and-score techniques [96]. Unlike traditional search-and-score algorithms, constraint-based algorithms exist that have been proven to be sound and complete in the sample limit. Hybrid approaches are designed to take advantage of the power of a constraint-based algorithm but while maintaining the flexibility of a search-and-score algorithm [96].

No matter which approach is taken, finding an optimal structure for a given set of training data is a computationally intractable problem. Structure learning algorithms determine for every possible edge in the network whether to include the edge in the final network and which direction to orient the edge. The number of possible graph structures grows exponentially with $|V|$ as every possible subset of edges could represent the final model. Due to this exponential growth in graph structures, even a restricted form of structure learning where variables are constrained to have only k parents has been proven to be **NP**-Complete [22]. Unless $\mathbf{P} = \mathbf{NP}$, there is no known efficient, polynomial-time algorithm for exhaustively searching the space of possible graph structures to determine a graphical structure that best describes the available data, D. Since an exhaustive search algorithm is not possible, existing structure learning algorithms either solve a restricted problem (i.e., find the best structure given a partial ordering of the variables) or find an approximation to the best compatible graph.

In general, the set of conditional independence relations of P do not entail a unique graph structure; there may be many graphs that are compatible with P. Therefore, accurately identifying conditional independence relationships present in the data may not be sufficient to produce a fully oriented model [19, 21, 87]. Rather than learning a fully directed model, many algorithms instead produce a partially directed acyclic graph (PDAG) model, also called a *pattern* or *essential graph* [6, 87, 98]. Edges that

are directed in the PDAG are edges that are directed in the set of all graphs $\mathbf{G}$ that are compatible with P. Consequently, edges directed in the PDAG are called *compelled* edges as only that edge orientation is compatible with the data [20]. Undirected edges in the PDAG represent edges that could be oriented in either direction and still be compatible with the training data. The set of directed graphs that match a fully directed instantiation of a PDAG are considered to be in the same *equivalence class*.

2.3 Evaluating Structural Accuracy

For evaluation purposes, structural accuracy of learned networks can be measured with a variety of different metrics that compare the structure of both the learned and true models. This requires the structure of the true model to be known *a priori*. This is typically achieved by generating data from a known model and then learning from that data. The first metric is the accuracy (percent of edges correct) of edges in the model [8, 14, 86]. A closely related metric is the precision and recall of causal structures [60]. I use precision and recall of compelled edges, where compelled precision is defined as the number of correct compelled edges divided by the total number of compelled edges in the learned model. A compelled edge is an edge that has the same orientation in every member of the equivalence class of the learned model [20]. For simplicity, I report a single number, the compelled F-measure, which is the harmonic mean of compelled precision and compelled recall [26] and is a general purpose metric of the correctness and actionability of the structure.

An alternative metric is structural Hamming distance (SHD), based on the raw counts of errors in the learned model [96]. The SHD of a model is a type of graph edit distance and is equal to the number of edge deviations between the model and the true model. It is often expedient to consider decompositions of the SHD, particularly into skeleton errors (false positive and false negative errors) and orientation errors

(errors of edge direction). I also use the number of true positive edges (number of correct edges) for evaluation.

CHAPTER 3

CONSTRAINT-BASED ALGORITHMS FOR LEARNING BAYESIAN NETWORKS

In this thesis, I focus on constraint-based approaches for learning the structure of Bayesian networks from data. Existing constraint-based algorithms perform structure learning in two independent phases. First, using a collection of conditional hypothesis tests, constraint-based algorithms learn independence constraints for variables that can be shown to be conditionally independent in the training data. This is the *constraint identification* phase, sometimes called *skeleton identification* after a popular representation of the constraints as an undirected *skeleton*, indicating the location but not orientation of edges appearing in the final model. Second, constraint-based algorithms select a fully oriented model using the learned constraints. This phase is often called *edge orientation*, as it can be viewed as finding an orientation for the undirected edges appearing in the skeleton. In the remainder of the chapter, I provide motivation for choosing a constraint-based algorithm over a search-and-score algorithm and provide greater detail of the inner workings of constraint-based, hybrid and refinement algorithms for learning the structure of Bayesian networks.

3.1 Motivation

One of the motivations for this thesis is to make structure learning of Bayesian networks a better tool for knowledge discovery tasks. Since the goal of knowledge discovery is actionable knowledge, learning accurate structure of the underlying distribution is desired. Learning a model that only provides accurate probability estimates,

but not accurate structure, is not sufficient for taking action because adjusting the variables to change the probabilities might not correspond to adjustments changing the underlying process. Constraint-based algorithms are ideally suited for knowledge discovery tasks for two reasons: (1) they are optimized for producing more accurate structure, and (2) they can produce Bayesian network models with causal interpretation in certain situations.

Constraint-based algorithms are better suited for learning accurate structure of Bayesian networks than search-and-score algorithms, which typically use a penalized likelihood score such as BDeu to choose model structures. Since training data are limited, the structure of the true model often contains more parameters than are supported by the data when optimizing for penalized likelihood. Optimizing for constraint satisfaction makes it possible to find structures with a large number of parameters as the size of the model is not part of the model selection criterion. In addition, algorithms based on constraints have been proven sound in the sample limit [87].

In addition to being able to learn more accurate structures compared to search-and-score algorithms, constraint-based algorithms can learn Bayesian networks with a causal interpretation when certain assumptions hold about the data [75, 87]. These assumptions are that all variables are measured (no latent variables), the distribution underlying the training data can be represented by a DAG, and the statistical decisions made from the data are correct (e.g., made in the sample limit). I will not focus on these assumptions in this thesis as they cannot be validated experimentally [80]. Rather than seek out expert validation for each dataset under consideration, I will focus on the structure learning component of the causal inference problem. The ability of a structure learning algorithm to recover the generating distribution can be evaluated experimentally if the data are generated from a known model. By improv-

ing the recovery of the generating structure, it is possible to improve the strength of causal inferences if the assumptions above are valid for a given data set.

3.2 Constraint Identification

Constraint-based algorithms are based on Pearl's definition of a Bayesian network in terms of a dependency model M. Viewed this way, the independence assertions contained in M are constraints that are learned from data. These assertions are of the form $(X \perp\!\!\!\perp Y|Z)$ indicating that "X is independent of Y given Z." Each independence constraint is typically decomposed into two parts. The first indicates a binary decision indicating whether an edge should exist between X and Y. There is a single constraint for each distinct pair of variables. The collection of all the binary decisions can be represented as an undirected skeleton. The second part of the independence constraints is the separating sets, or sepsets, Z. For each pair of variables X and Y that are determined to be independent, the separating sets indicate which set of variables Z that are necessary and sufficient to d-separate X from Y.

Constraint identification is the process of learning the skeleton and separating sets from the training data. Due to limited size of the training data, this process is inherently error-prone. The goal of constraint identification is to efficiently identify the independence assertions while minimizing the number of constraints that are inaccurate. Constraint identification algorithms appearing in the literature can be differentiated by three different design decisions: (1) the type of independence test used, (2) the ordering heuristic and other algorithmic decisions, and (3) the technique used to determine the reliability of the the test. The following sections identify the possible choices for each decision and the implications of those decisions on the error rate of the overall algorithm.

3.2.1 Hypothesis Tests of Independence

Constraint-based skeleton algorithms utilize tests of conditional independence to determine which structure to include in the skeleton. In practice, many types of hypothesis tests are used to determine independence, including classical, Bayesian, and information theoretic tests. Each type of test follows the traditional hypothesis testing framework consisting of null and alternative hypothesis and a significance threshold to determine whether to accept or reject the alternative hypothesis. Since the overall goal of this work is learning accurate constraints, one critical characteristic of hypothesis tests for structure learning is the ability to characterize and bound the rate of errors incurred by running the hypothesis test.

Classical hypothesis tests for categorical data typically utilize either the χ^2 or G^2 statistic when determining whether to accept the alternative hypothesis [87, 96]. Both the χ^2 and G^2 statistics are computed from a contingency table containing counts of variable values occurring in the data. The null hypothesis assumes the data are independent and distributed as χ^2 with degrees of freedom that depend on the size of the table. The alternative hypothesis is accepted and dependence is proven when the the probability of the observed statistic under the null hypothesis is below a specified significance threshold, typically $p = 0.05$. The degrees of freedom are an indicator of the number of parameters that can be varied in the model. Fewer parameters indicates more data can be used to estimate each parameter, leading to more accurate estimates. The χ^2 distribution can also be used to specify a precise alternative hypothesis for computing statistical power.

An alternative to the classical tests is a hypothesis test based on a Bayesian score such as BDeu [1, 27]. This score is computed for the current network with and without the current edge. If the score is greater for the network with the edge, then the two variables are dependent and the edge is added to the network. One advantage of determining independence with a Bayesian approach is the ability to smooth with a

prior [27]. Incorporating a prior can mitigate problems with false negative errors due to small sample sizes. Calculation of the false negative rate (i.e., statistical power) during learning with a Bayesian score is difficult; however, as the statistic depends on the number of parameters of the model which vary with the structure being learned.

A third approach for testing conditional independence utilizes a test of mutual information [18]. If one variable provides some information about another variable, then the two variables are be dependent. As with all tests of conditional independence, determining mutual information is often noisy at small sample sizes. To address this problem, a threshold is used to determine if the observed effect is significant. Unfortunately, the threshold of significance varies with both sample size and the size of the test [18, 96]. Cheng et al. [18] devised a heuristic approach for choosing a threshold for a given sample. Because a heuristic technique is used and the threshold can vary with sample size, it is difficult to compute a consistent rejection region necessary for performing analysis of statistical power. Though the threshold is designed as a control for statistical power, the threshold is not sufficient for computing power exactly. In addition, Hutter [47] shows that point estimates of mutual information are often inaccurate and that consideration of the second-order distribution is necessary to improve structural accuracy.

In this thesis, I will only consider classical hypothesis tests using the χ^2 score. These tests have a well-defined framework for evaluating the rates of type I and type II error rates. As I will demonstrate in Chapter 4, precise analysis of type I and type II error rates is useful for understanding structure learning algorithms for Bayesian networks.

3.2.2 Ordering Heuristics

Given a particular independence test, each constraint identification algorithm applies the tests in a particular order. Many ordering heuristics appear in the literature;

instead of presenting each one in detail, I highlight the major categories of ordering heuristics with citations to relevant work.

The predominant type of skeleton heuristics are local algorithms that consider each test independently of other decisions. I use the first and most widely used ordering algorithm: Fast Adjacency Search (FAS). The FAS algorithm for constraint identification is drawn from steps A and B of the PC [1] algorithm, as described in Spirtes et al. [87]. Pseudocode for FAS is shown in Algorithm 1. An implementation of the FAS algorithm can be found in the TETRAD IV[2] package. FAS operates in a breadth-first manner considering all pairwise tests followed by all tests conditioned on a single variable and so on until no more tests can be run.

Other ordering heuristics have been presented in the literature as improvements on the FAS algorithms. Max-Min Parents Childen (MMPC) [95, 96] is the constraint identification algorithm used with the Max-Min Hill Climbing (MMHC) algorithm. FAS and MMPC utilize different heuristics to produce a skeleton with the fewest statistical tests. Abellan et al. [1] propose an additional optimization to FAS that breaks triangles in the skeleton by removing the weakest link among three edges in the triangle before moving on to the next stage. MMPC operates in a more depth-first manner, considering all tests for a single target variable before considering additional variables. In a different approach, Xie and Geng [99] describe a recursive splitting algorithm to limit the number of other variables considered when searching for a separating variable. These algorithms share the same asymptotic properties of FAS, but these algorithms either encode additional assumptions from FAS or are not compatible with existing corrections for errors.

---

[1] PC is named for its creators Peter (Spirtes) and Clark (Glymour)

[2] `http://www.phil.cmu.edu/projects/tetrad/`

Neighborhood models are another type of ordering heuristic that considers intermediate path information when determining whether to add an edge between the two variables under consideration. If a possible conditioning variable does not lie on a path between the two variables being tested then it should not be used to prove conditional independence. This arises out of the definition of d-separation in graphical models (see Section 2.1)[74]. Three-Phase Dependency Analysis (TPDA) is the most prominent neighborhood structure learning algorithm [18]. Steck and Tresp [89] also incorporate neighborhood information into the PC algorithm using an additional constraint, called the Necessary Path Constraint, on the conditioning set of a test.

The FAS algorithm is the oldest and most widely studied constraint identification algorithm [1, 57, 87, 89, 97]. Like MMPC and TPDA, the FAS skeleton algorithm is asymptotically correct. However, unlike the other algorithms, the FAS algorithm is an extensible platform that can incorporate the majority of the other improvements and corrections proposed in the literature. Since one contribution of this thesis is evaluating the efficacy of these innovations, the majority of the reported experiments are run using the FAS algorithm to provide a stable comparison.

3.2.3 Determining the Reliability of a Hypothesis Test

It has long been noted in the constraint-based structure learning literature that the reliability of the hypothesis test used in structure learning is inversely related to the size of the conditioning set; as conditioning sets grow, reliability decreases [27, 87, 96, 97, 100]. There are two distinct but related characteristics of a test that determine its reliability. First, a reliable test meets the distributional assumptions of the statistic used; for example, classical tests assume that the data are asymptotically distributed as χ^2 . Second, a reliable test has sufficient statistical power. Statistical power is the probability of successfully identifying a significant effect if one exists in the data. As the size of the contingency table grows, for a fixed amount of data

both the distributional correspondence and the statistical power decrease, leading to a corresponding decrease in reliability.

Many structure learning approaches utilize a "rule of thumb" to determine whether the distributional assumptions hold and a hypothesis test will be reliable [87, 96]. This rule of thumb states that a test is reliable if there are five or more instances per parameter of the test (degree of freedom) [33]. Other algorithms are designed to limit the size of the possible conditioning sets to improve reliability [36, 97]. This is also one of the arguments for considering neighborhood information when making a statistical decision as neighborhood algorithms use smaller conditioning sets, which improves power [18, 89](see Section 3.2.2). Another approach for improving reliability is to average over multiple models with different structures either with a Bayesian approach [13] or by exploring the inconsistencies encoded in the skeleton [89]. Our focus for this work is knowledge discovery; therefore, I focus on algorithms that produce a single model for easier human interpretation.

If a test is determined to be unreliable, then many algorithms make a default decision to include the edge in the model. When a test is unreliable, an insignificant result does not differentiate between an error and a lack of correlation. The *default decision* is a decision to automatically reject the null hypothesis and assume dependence, providing protection against false negative errors that might occur as a result of unreliable tests.

Accurately determining the reliability of the hypothesis tests is critical for minimizing errors in the learned constraints. In Chapter 4, I compare the strategies mentioned here with a new approach based on statistical power analysis to determine which approach is most accurate in practice.

3.3 Alternative Skeleton Algorithms

In addition to constraint-based skeleton algorithms, which use hypothesis tests of conditional independence, there are a variety of alternative approaches for constraining search. Frequent item sets have been used in place of conditional independence tests as a basis for skeleton identification [39]. The Sparse Candidate (SC) algorithm uses an iterative search process to determine the best structure subject to the constraint that no variable can have more than k parents [36]. The parameter k is determined by the user prior to running the algorithm and can exclude high-scoring structures if set inappropriately. Tsamardinos et al. [96] show that MMHC is a sound and complete version of Sparse Candidate and only requires a single iteration.

In general, any approach for learning the structure of undirected networks could be used as a skeleton formation algorithm e.g., Bromberg and Margaritis [13]. Also, since dependency networks have undirected conditional independence semantics [42], it would be possible to extract an undirected skeleton from a learned dependency network. (Note this approach differs from Hulten et al. [46], which uses a dependency network as a mechanism for caching statistics for Bayesian network learning.)

Many of the early algorithms for learning the structure of Bayesian networks, such as K2, constrained search by requiring the user to specify an ordering over the variables [25]. This is a type of hybrid algorithm where the skeleton is specified by the user. Recent work by Teyssier and Koller describe an algorithm that searches over possible orderings [92]. While this approach is not an algorithm based on independence constraints, it uses ordering constraints to make search more efficient.

These algorithms do not use conditional hypothesis tests to determine the constraints. The errors of conditional hypothesis tests are well-defined and have been studied extensively. Since the goal of this thesis is reducing errors in constraint identification, my focus on constraint identification algorithms that use conditional hypothesis tests, though improving these other approaches may also be possible.

3.4 Constraint-Based Edge Orientation

The PC algorithm is a constraint-based algorithm that utilizes three deterministic edge orientation rules for determining edge direction [18, 76, 87]. These rules are based on the structure between variables and conditioning sets determined during the skeleton phase and are appropriate when all possible causes are represented in the data.

The edge orientation rules are as follows (from Spirtes, Glymour, and Scheines [87, Section 5.4.2]):

1. For each triple of vertices X, Y, Z such that the pair X, Y and the pair Y, Z are each adjacent in the skeleton but X, Z are not adjacent, orient $X - Y - Z$ as $X \rightarrow Y \leftarrow Z$ if and only if Y is not in the conditioning set separating X and Z

2. If $A \rightarrow B$, B and C are adjacent, A and C are not adjacent, and there is no arrowhead at B, then orient $B - C$ as $B \rightarrow C$.

3. If there is a directed path from A to B, and an edge between A and B, then orient $A - B$ as $A \rightarrow B$.

An alternative formulation of these rules is provided by Meek [65]. The rules are applied in lexicographic order and are proven to be correct in the sample limit when the skeleton and conditioning sets are correct. In most instances, however, the skeleton algorithm will not produce the correct skeleton and conditioning sets. Therefore, empirical analysis is necessary to understand when and how errors during the skeleton phase impact edge orientation.

3.5 Hybrid Learning Algorithms

Hybrid algorithms combine aspects of both constraint-based and search-and-score algorithms. The standard hybrid approach first runs some form of constraint identification followed by a heuristic search limited to the edges appearing in the skeleton

[96, 97]. Any heuristic search technique for Bayesian network structure learning that can be constrained to the skeleton can be used. These approaches do not use the constraints for model selection.

Heuristic search techniques are the most popular form of unconstrained structure learning algorithms for Bayesian networks. These techniques frame structure learning as a global optimization problem searching for the model structure which optimizes some global score. Popular scores include BIC [81], AIC [5], and BDeu [15, 44]. At each step in the algorithm, all possible edge additions, deletions and reversals are considered. The highest scoring operator is applied to the network and the algorithm continues until no operator can improve the score. Often techniques such as random restarts or tabu lists are maintained to avoid local extrema. These search techniques can be tailored for either generative or discriminative purposes [40].

Rather than searching over all possible model structures, some approaches only search the space of equivalence classes of models [3, 20]. These algorithms consider a smaller search space but do not restrict the possible models within that search space. Other approaches for improving the search phase include advance search operators to escape local extrema [67] and more advanced stochastic search techniques [45].

By combining the strengths of both constraint-based and search-and-score algorithm, hybrid algorithms are able to provide many of the advantages of both approaches. Consequently, a hybrid strategy is instrumental for the success of the constraint satisfaction approaches introduced later in the thesis.

3.6 Refinement Algorithms

In addition to constraint-based and hybrid approaches, there are refinement approaches for learning the structure of Bayesian networks. A refinement algorithm first runs a constraint-based algorithm (constraint identification and edge orientation) to completion and then uses a complementary approach to improve the final structure.

Abellan et al. [1] describe a hybrid refinement algorithm. After the PC algorithm has completed, Abellan et al. use heuristic optimization techniques to improve the likelihood of the final models at the expense of the causal interpretation of the models [1]. Bromberg and Margaritis [14] use the logic of argumentation combined with the axioms of conditional probability to refine the results of the PC algorithm based on the logical consistency of the constraints. Refinement algorithms also provide a strong influence on the constraint satisfaction algorithms introduced later in the thesis.

3.7 Summary

Constraint-based algorithms are designed to learn accurate Bayesian network structure from data. Existing constraint-based algorithms, such as PC and TPDA, have been proven to be sound and complete in the sample limit. With more reasonable sample sizes, however, these algorithms are prone to errors. Many different approaches have been tried to reduce errors. In the following chapters, I explore many of these approaches and propose new constraint satisfaction approaches to reduce errors. For these experiments, I primarily use the FAS constraint identification algorithm and the deterministic edge orientation rules as a comparison. Although there many possible choices for each component of constraint-based algorithms, the goal is provide a fair comparison of any new approaches.

Algorithm 1 Fast Adjacency Search (FAS)

procedure FAS(Data **D**, Variables **V**)
  $P = \emptyset$ ▷ P contains dependent pairs of variables
  C = Empty graph over **V**
  for $v_1, v_2 \in$ **V do**
    $P = P \cup \{\langle v_1, v_2 \rangle\}$ ▷ Assume every pair is dependent, add to P
    Add edge $\langle v_1, v_2 \rangle$ to C ▷ Form complete undirected graph over V
  end for
  $n = 0$ ▷ n indicates the size of the candidate conditioning sets.
  while $|P| > 0$ **do**
    for $\langle v_1, v_2 \rangle \in P$ **do**
      RunTest = $false$
      adj = Adjacencies(C,v_1) $\setminus$ v_2 ▷ Get the neighbors of v_1 excluding v_2
      if $|adj| < n$ **then** ▷ No more tests to be run.
        $P = P \backslash \{\langle v_1, v_2 \rangle\}$ ▷ Remove pair $\langle v_1, v_2 \rangle$ from P
      end if
      for $S \subseteq adj$ s.t. $|S| = n$ **do**
        if Test is not reliable **then**
          Continue
        end if
        RunTest = $true$
        $pVal$ = Run hypothesis test (v_1, v_2, S, D) ▷ Run Hypothesis Test.
        if $pVal > 0.05$ **then** ▷ 0.05 is standard threshold
          Add constraint $(v_1 \perp\!\!\!\perp v_2 | S)$
          Remove edge between v_1 and v_2 in C
          $P = P \backslash \{\langle v_1, v_2 \rangle\}$ ▷ Remove pair $\langle v_1, v_2 \rangle$ from P
        end if
      end for
      if RunTest = false **then** ▷ Insufficient power to run any test.
        $P = P \backslash \{\langle v_1, v_2 \rangle\}$
      end if
    end for
    $n = n + 1$
  end while
end procedure

CHAPTER 4

ERRORS OF CONSTRAINT IDENTIFICATION

The first step in reducing the errors of constraint-based structure learning is understanding the sources of errors. In this chapter, I identify the types and analyze the sources of errors made by the FAS algorithm during constraint identification. The results of this analysis indicate that false negative errors due to low power statistical tests are the largest source of errors during constraint identification.

4.1 Introduction

There are two kinds of errors made during constraint identification: skeleton errors and separating set errors. Recall that the constraints considered here are independence constraints of the form $(X \perp\!\!\!\perp Y|Z)$ indicating that "X is independent of Y given Z," and there is a single constraint for each unique pair of variables occurring in the network. *Skeleton errors* are errors in the binary edge decision for a pair of variables, that is, whether to add an edge in the model between those two variables. An error of including an edge in the learned model that does not occur in the true model is a *false positive* error. An error of excluding an edge from the learned model that does occur in the true model is a *false negative* error. *Separating set errors*, also called *sepset errors*, occur when the constraint correctly indicates independence for a pair of variables, but identifies a separating set Z that is not consistent with the true model.

Errors can arise from either the determination of test reliability or the statistical test used by the algorithm for constraint identification. The ordering heuristic, while

a significant portion of a constraint identification algorithm, cannot directly produce errors. Existing ordering heuristics, including FAS, MMPC, and TPDA, are proven to be correct when the hypothesis tests are perfectly correct (i.e., in the sample limit) [18, 87, 96]. Therefore, the errors occur as a result of a faulty statistical decision, either independently or in conjunction with a faulty decision to run an unreliable test, although there is the potential for complex interactions between the algorithmic and statistical components of constraint identification.

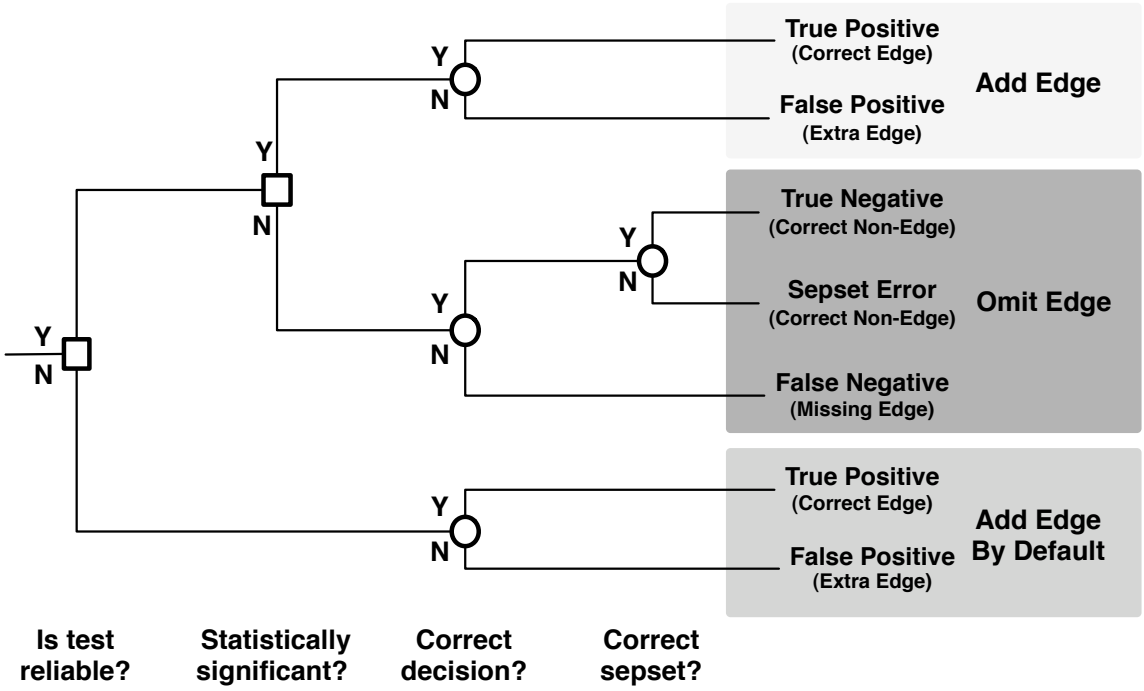


Figure 4.1: A Decision Tree used to determine the composition of the constraints.

False positive and false negative errors are closely related to traditional type I and type II errors from the statistical literature. Recall that a type I error is the error of rejecting the null hypothesis when it is true, and a type II error is the error of accepting the null hypothesis when it is false. For clarity, I will always refer to false positive and false negative errors in the learned model, while type I and type II errors refer to the results of individual hypothesis tests. Therefore, false positive and false

negative errors can occur as a result of multiple interacting individual tests, whereas type I and type II errors can only result from a single test. A false positive error in the learned model can occur from either a type I error in an individual test, or as a result of a default decision to add an edge when an individual hypothesis test is determined to be unreliable. A false negative error always occurs as a result of a type II error somewhere in the series of tests leading to the decision to not add the edge to the model. A graphical representation of this decision process is shown in Figure 4.1.

4.2 Analysis of Errors

The frequency of each kind of structural error can be determined by comparing the structure of the learned model with the structure of the model that generated the training data. The easiest way to accomplish this type of evaluation is to generate the training data from a known Bayesian network. A Bayesian network can be created manually by experts or sampled randomly from the space of possible models. Generating data from known models makes possible the evaluation of a learning algorithm across a range of sample sizes and data characteristics. It may also be possible to create a biased sampling procedure to synthetically generate models with desired structural characteristics [48].

The remainder of this section describes an empirical analysis of the Fast Adjacency Search (FAS) algorithm. The goal of this analysis is to determine whether false positive or false negative errors more frequently occur in runs of the FAS algorithm under realistic conditions. For this exploratory phase of the thesis, I considered datasets generated from eight Bayesian network models gathered from real decision support problems. The list of networks considered along with information about the number of edges appearing in the model and the average cardinality of the variables

can be found in Table 4.1. Additional details about each of these models can be found in Appendix A.

For each model, I generated five datasets at each of the following sample sizes: $n = \{500, 1000, 2000, 5000, 10000\}$. For each dataset of each model, I ran an annotated version of the FAS algorithm that has access to the true model and, for each edge decision, reports whether that decision results in a statistical false positive, false positive as a result of the default decision (default false positive), or a false negative error. The results of that experiment are shown in Figure 4.2. On every network with the exception of HAILFINDER, the number of false negative errors dominates the statistical false positive errors. On networks with low-cardinality variables such as ALARM, false negative errors also dominate the default false positives as very few tests have sufficiently large degrees of freedom to trigger the default decision. On BARLEY, DIABETES0, and HAILFINDER, the default false positives are the largest source of error at smaller sample sizes but drop below the false negatives at high sample sizes.

The aim of making a default decision is to reduce the overall number of errors, however making the decision to add an edge when the test is unreliable actually leads to more errors than it corrects. The total number of edges appearing in each network is shown in Table 4.1. At the smaller sample sizes, the number of default false positive errors is considerably larger than the total number of edges in the generating model Therefore, a simple way to reduce errors would be to simply run every test and never make a default decision to add an edge. Since the default decision was intended to decrease the chance of making a false negative error, never making a default decision should increase the number of false negative errors.

Additional evidence for the importance of correcting false negative errors is provided by analyzing the separating sets of the learned and true networks. A separating

Figure 4.2: Number of false negative (FN) and false positive (FP) errors. False positive errors are decomposed into errors due to making a default decision and due to a statistical error. In all figures, fewer errors is better.

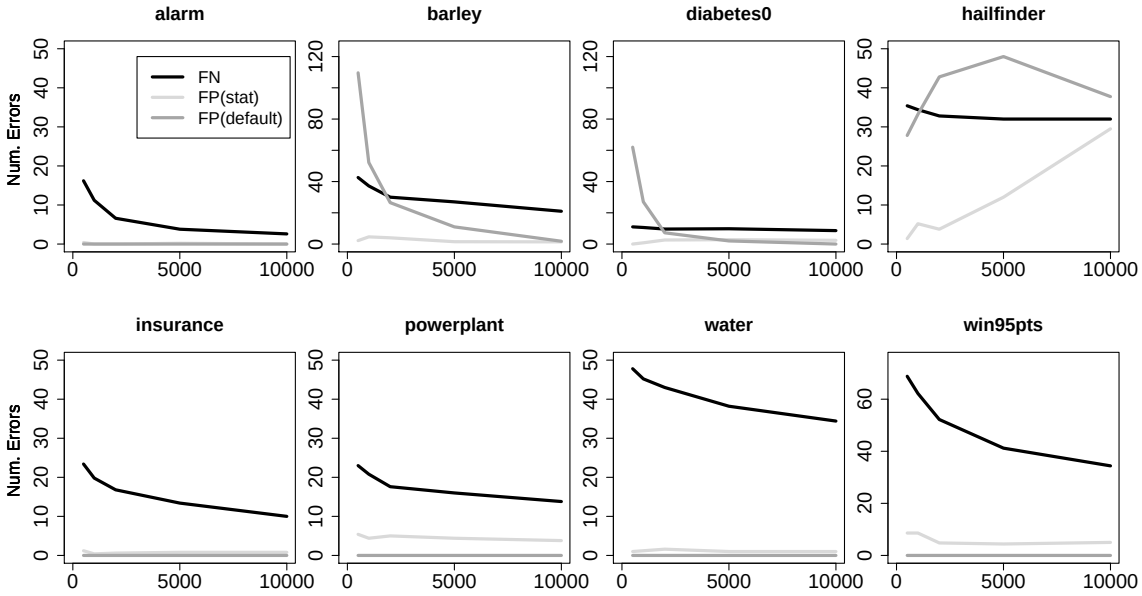


Table 4.1: Number of edges and average cardinality of Bayesian networks considered during error analysis.

Data Name	Num. Edges	Avg. Cardinality
Alarm	46	2.84
Barley	84	8.77
Diabetes0	23	11.21
Hailfinder	66	3.98
Insurance	52	3.30
Powerplant	42	3.0
Mildew	46	17.6
Water	66	3.63
Win95pts	112	2.0

set error occurs when the algorithm concludes independence with a separating set that is not compatible with the true network. For a given pair of variables, there may be multiple separating sets of the minimal size that are all sufficient to d-separate the variables. A learned separating set is compatible with the true network if it is also the minimal size and exactly matches ones of the true separating sets. Since the binary edge decision of independence is correct, separating set errors are the result of what Mosteller [68] has called type III errors—making the right decision for the wrong reason.

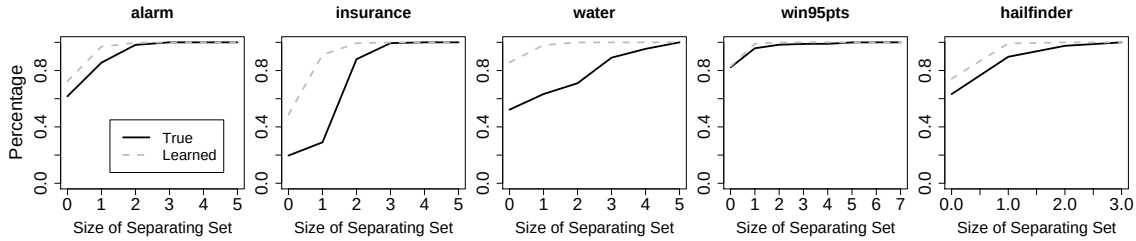


Figure 4.3: CDFs of true versus learned sepsets.

There are two possible scenarios leading to separating set errors. The first occurs when a false negative error causes the algorithm to conclude independence with a separating set that is smaller than the true set. The second occurs when an earlier false positive error causes the algorithm to incorrectly conclude dependence with the true separating set and then conclude independence with a larger separating set. Figure 4.3 shows the empirical CDFs of the size of the learned and true separating sets. The sizes of the learned separating sets are consistently smaller than the true separating sets, indicating that the false negatives are the larger cause of separating set errors.

4.3 Focus on False Negative Errors

Although both false positive and false negative errors are committed by the FAS algorithm, the emphasis of the remainder of this chapter is on understanding the sources of false negative errors. While false positive and false negative errors both negatively impact the causal claims of a learned model, false negatives are particularly harmful. When a false positive error occurs, parameter estimation can mitigate the effect of that error by giving that edge a low weight. Since false negative errors lead to edges being omitted from the model, there is no opportunity for correcting these errors.

False negative errors are the only source of skeleton errors that do not currently have a satisfactory solution. My results show that statistical false positives are less frequent than other types of errors and recent work has provided a tight theoretical bound on the number of statistical false positive errors [94]. Default false positives are a larger source of errors, but they also a have an easy solution: do not add any edges by default. This would eliminate all default false positive errors at the risk of increasing the false negative errors. False negative errors are the largest source of error on many networks. As I show in the remainder of the chapter, existing corrections for false negative errors cannot address each of the sources of false negative errors. Consequently, reducing the number of false negative errors is the largest remaining obstacle for learning accurate structure of Bayesian networks.

4.4 Sources of False Negative Errors

Prior work in structure learning has identified three sources of false negative errors: (1) unsuitable hypothesis tests, (2) unexplained d-separation, and (3) low-power hypothesis tests. In the remainder of this section, I describe each possible source of error and introduce existing corrections for each source.

4.4.1 Unsuitable hypothesis tests

In categorical data, *unsuitable hypothesis tests* occur when the expected frequencies in some of the cells of the contingency table are small, either due to small sample sizes or large contingency tables [87, 96]. This was determined using Monte Carlo studies comparing the p-value of the G^2 statistic with a χ^2 test against the exact p-value produced using computationally intensive statistics to generate the sampling distribution [53]. These studies showed that the G^2 statistic is not a suitable approximation of the χ^2 distribution and does not produce accurate p-values if there are fewer than five training sample per degree of freedom of the test.

Traditional skeleton algorithms, such as the FAS algorithm, use a "rule of thumb" to determine whether a hypothesis test is suitable and therefore unlikely to result in a false negative error [87, 96]. Consequently, the rule of thumb prevents the running of a hypothesis test if the test is unreliable, that is, if there is insufficient training data, that is, if there are fewer than five training sample per degree of freedom of the test.

4.4.2 Unexplained d-separation

Unexplained d-separation produces a false negative error when a variable (or set of variables) can be used to prove that two variables are independent, but that variable does not appear on a path between the variables being tested [18, 89]. This follows from the d-separation rules that define conditional independence relationships based on the structure of the graphical model [74]. Variables that do not appear on an undirected path between the two variables being tested cannot d-separate those variables.

Steck and Tresp [89] proposed the necessary path condition as a correction for unexplained d-separation. The necessary path condition requires that for a variable z to d-separate x and y, it must fall on an undirected path between x and y. Enforc-

ing this condition leads to fewer variables being considered as possible conditioning variables, which could result in fewer edges being removed from the skeleton. Abellan et al. [1] use a stricter form of the necessary path condition that only conditions on variables in the minimum cut set appearing between the two variables.

The necessary path condition is incorporated in at least two variants of the FAS algorithm (but not the original) to prevent false negative errors due to unexplained d-separation [1, 89]. To enforce the path condition, the skeleton identification algorithm must maintain, at each step, a superset of all the edges that could be included in the skeleton. This requirement prohibits the necessary path condition from being applied in the MMPC algorithm, which operates in a depth-first fashion [96]. Steck and Tresp [89] use the necessary path condition to identify inconsistent regions produced by the PC algorithm, but do not return a single model.

4.4.3 Low statistical power

The *statistical power* of a hypothesis test is the probability of detecting a significant effect given that it exists in the data [24]. A test with low statistical power will often fail to detect a true effect resulting in a false negative error. This has long been recognized as a problem in constraint identification [87]. Statistical power depends on the sample size N, the degrees of freedom of the test, the significance threshold α (or type I error rate), and the expected effect size w [24]. The *effect size* of a test defines a specific alternative hypothesis to compare against the null and indicates the minimum strength of correlation that is detectable by the hypothesis test. Running tests with small sample sizes or large degrees of freedom result in sampling distributions with high variance, leading to a corresponding decrease in statistical power.

The primary approach for preventing low-power statistical tests is to incorporate an upper limit on the degrees of freedom of the tests used in constraint identification. The first, and most widely used, approach of this type is the "rule of thumb" de-

scribed in Section 4.4.1. As described above, the rule of thumb used to determine the suitability of a hypothesis test for discrete data is based on a threshold of the degrees of freedom of the test [87]. However, the power threshold determined by the rule of thumb is not consistent as the sample size changes. The inconsistency results from a failure to account for each of four dimensions for determining statistical power; in particular, the rule of thumb cannot account for the possible effect sizes present in the data. Consequently, the rule of thumb can only identify a subset of all possible low power tests. Limiting the number of variables considered in a conditioning set is a coarser approach for limiting the degrees of freedom of a hypothesis test [97].

An alternative approach for correcting low-power hypothesis is to use a different type of hypothesis test, such as tests of mutual information or tests using a Bayesian score such as BDeu [1, 18]. These tests typically use a weak significance threshold, such as determining whether the score is greater than zero, to determine independence. These approaches can also be plagued by the statistical issues that lead to false negative errors. For example, Hutter [47] shows that point estimates of mutual information are often inaccurate and that consideration of the second-order distribution is necessary to improve structural accuracy.

In the following section, I introduce the POWER correction as the first correction to explicitly control for all aspects of statistical power, including reasoning about the possible effect sizes in data. While the concept of statistical power is familiar to many researchers, to our knowledge full statistical power analysis has not been incorporated into any structure learning algorithms. For example, power calculations were mentioned by Tsamardinos et al. [96] as a possibility for determining test reliability but were not incorporated into the final algorithm [96, Sec. 4].

4.5 The Power Correction

The *statistical power* of a hypothesis test is the probability of detecting a significant effect given that it exists in the data [24]. Statistical power depends on the sample size N, the degrees of freedom of the test, the significance threshold α (or type I error rate), and the expected effect size w. The *effect size* of a test defines a specific alternative hypothesis to compare against the null and indicates the minimum strength of correlation that is detectable by the hypothesis test.

The Power correction is an original approach for controlling statistical power that first appeared in my prior work, Fast et al. [31]. The Power correction is the first approach that is able to address all four factors contributing to low statistical power. Approaches for controlling power that only limit the degrees of freedom of the test, such as the rule of thumb, do not account for all of the factors that determine statistical power. Consequently, these approaches provide only a weak bound on false negatives due to low-power tests. In particular, existing approaches do not account for the possible effect sizes present in the data. If the effect size to detect is small, then the test could have low statistical power and produce false negative errors despite using other forms of correction. The minimum power levels permitted under the rule of thumb for small effect sizes are shown in Figure 4.4. Our experiments show that effect sizes actually occurring in the experimental data result in low-power statistical tests under the rule of thumb.

4.5.1 Accounting for Effect Size

The Power correction uses the framework of statistical power analysis to reason about the potential effect sizes in the data and produce a correction for low-power statistical tests that can address each of the factors contributing to low statistical power [24]. Similar to the rule of thumb, the Power correction defines a limit on the acceptable ratio between the degrees of freedom of the test and the sample size. Since

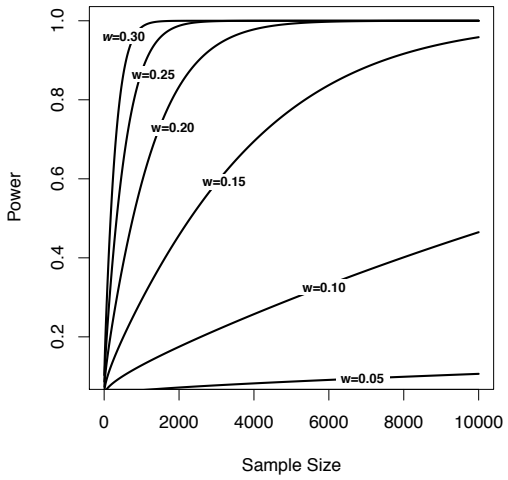


Figure 4.4: Minimum statistical power permitted under the rule of thumb.

the sample size N is fixed for a particular dataset, the POWER correction determines whether a test with the given degrees of freedom has sufficient statistical power. The desired level of statistical power can be determined by the analyst. Recall, the power of a statistical test is $1 - \beta$, where β is the type II error rate, which is the probability of rejecting the alternative when it is true. For the experiments described here, I used a desired power level of 0.95. This corresponds to β of 0.05, matching the standard level for α. When N, α, the degrees of freedom, and effect size w are specified, it is possible to compute the statistical power of the test [24]. In skeleton identification algorithms, the value of N and the degrees of freedom are determined by the data and the specific test, respectively. The values of α (typically 0.05) and w can be set by the user before running the algorithm.

The statistical power of a hypothesis test corresponds to the area under the alternative distribution that exceeds the critical value specified by α. For categorical data, the alternative distribution is specified by a noncentral χ^2 distribution with noncentrality parameter, $\lambda = w^2 N$ [24]. The evaluation of $1 - \beta$ requires an infinite summation of the noncentral χ^2 distribution [66]. To compute power, I use the effi-

cient two-stage approximation identified by Milligan [66]. Other implementations of power calculations are found in many common statistical packages (e.g., R [17]).

For categorical data, effect size is measured by w, a scale-free measure based on the χ^2 statistic, which is independent of the sample size provided to the test. The values of w can range from 0 to a maximum that is determined by the size of the table. In practice, it is rare to observe a value of w above 0.90 [24]. Since w is unfamiliar to most researchers, Cohen [24] suggests values of w for small ($w = 0.1$), medium ($w = 0.3$), and large ($w = 0.5$) effects. The value of w is determined by

$$w = \sqrt{\sum_{i=1}^{m} \frac{(P_{1i} - P_{0i})^2}{P_{0i}}} \tag{4.1}$$

where P_{0i} is the proportion in cell i under the null hypothesis, P_{1i} is the proportion in cell i under the alternative hypothesis and reflects the effect of that cell, and m is the number of cells in the contingency table [24]. Note the similarity to the χ^2 statistic used to assess association in categorical data.

In practice, rather than computing statistical power for every test, high-power tests can be identified by specifying an acceptable range for degrees of freedom of the test. Since power is inversely proportional to the degrees of freedom, the acceptable range for given level of power can is determined analytically by computing the statistical power for every possible degree of freedom, starting at one and increasing until the statistical power falls below the desired level. The pseudocode for this algorithm is shown in Algorithm 2. The computation is efficient and can be performed quickly before starting structure learning. I chose this approach because both FAS and MMPC already make reliability decisions based the degrees of the freedom of the test; I simply substitute a new threshold based on statistical power. In contrast to the rule of thumb, the power threshold maintains a constant level of power at all sample sizes. For categorical data, the degrees of freedom is $(r - 1)(c - 1)d$, where r and c are the number of distinct values taken on by the two variables whose dependence is

being tested, and d is the number of possible joint values of the set of conditioning variables.

Algorithm 2 Determining a DOF Threshold

procedure DOF(N, w, p, α)
Input: Sample size, effect size, significance level, and desired power.
Output: Degree of freedom (DOF) limit

$d = 0$ ▷ current DOF
$p' = 1.0$ ▷ power at current DOF
while $p' > p$ **do**
$d++$
$p' =$ POWER(N, w, α, d) ▷ Actual computation depends on the statistic.
end while
return d-1 ▷ Highest DOF with sufficient power.
end procedure

4.5.2 Determining the Effect Size Parameter

Determining an appropriate value of the effect size parameter w is critical to the success of the POWER correction. The w parameter specifies the smallest effect size appearing in the data that the hypothesis test can detect with the desired level of power. If this value is known *a priori* for a given dataset, then the value of w can be set by the analyst. Otherwise, cross-validation can be used to determine the best value of w. Although cross-validation produces good estimates of the minimum effect size w, it not does not permit fast learning of Bayesian network structure. To avoid the computational overhead for running cross-validation every time a new dataset is encountered, I determined a value for w using cross-validation for a set of benchmark datasets selected randomly from the Bayesian Network Repository.[1] I then used those values of w to determine a general method for estimating a suitable value of w for a given dataset.

[1] `http://compbio.cs.huji.ac.il/Repository/`

The choice of w is limited by two undesirable extremes. For any constraints to be determined by data, the value of w must permit the skeleton algorithm to run some tests. Therefore, the minimum value of w considered corresponds the largest effect size such that a test with only a single degree of freedom would not have sufficient power. The maximum value of w considered is the effect size which corresponds to the rule of thumb. These extremes are shown as dashed grey lines in Figure 4.5. For a dataset of a certain size, I can choose a value of w based on the distance between the minimum and maximum effect sizes using Equation 4.2, where $w_{min}(w_{max})$ is the minimum (maximum) effect size considered for that sample size and c is a constant computed from training data as described below.

$$w = c(w_{max} - w_{min}) + w_{min} \tag{4.2}$$

To determine the best distance between the extremes, I used ten-fold cross-validation to determine the optimal value of w for the FAS skeleton algorithm. Following the standard cross-validation procedure, I divided each training set into ten folds. I ran both phases of a hybrid algorithm (skeleton identification and heuristic search) to learn a Bayesian network from training data composed of nine of the folds, and computed the likelihood of data contained in the last fold given the learned model. The value of w used to compute the scaled distance traveled from the minimum to the maximum is an average of the values of w which produced the highest likelihood model on each fold. I repeated this procedure for a range of w at each sample size. The value of c is the average scaled distance across sample sizes. The datasets I considered were: DIABETES0[2], HAILFINDER, BARLEY, and INSURANCE. These networks are not considered later for evaluation. The learned effect size at each sample size and value of c I used to compute w are shown in Table 4.2. A graph

[2]We constructed DIABETES0 by removing all but the first time slice of the DIABETES network.

of the change in the learned w over sample sizes is shown in Figure 4.5. Despite the variety in the data characteristics, the value of w determined using cross-validation is tightly bunched across the different networks.

Table 4.2: The effect size parameters chosen via cross-validation.

Sample Size	w	Min	Max	Distance
500	0.2105	0.1612	0.3372	0.2801
1000	0.1514	0.1140	0.2761	0.2307
2000	0.1172	0.0806	0.2276	0.2544
5000	0.0778	0.0510	0.1775	0.2119
				c=0.2524

4.6 Evaluating Corrections for False Negative Errors

To determine the source of false negative errors in constraint identification, I applied the "rule of thumb" correction, the POWER correction and the necessary path correction to the FAS constraint identification algorithm and measured the corresponding decrease in false negative errors of each correction. Based on the magnitude of improvement provided by each correction, I can infer the proportion of errors attributed to each source. I found that the POWER correction was the only correction to produce a significant decrease in the number of false negative errors made by the FAS algorithm. To measure these error rates, I added each correction to the FAS skeleton algorithm and applied the new algorithm to a set of datasets sampled from models downloaded from the Bayesian Network Repository and not previously considered for cross-validation. The following networks were used: ALARM, MILDEW, PATHFINDER, WATER, WIN95PTS. I created training samples of 500, 1000, 2000, and 5000 instances and a single test sample of 10,000 instances. Please note that these results were generated using an early version of the POWERBAYES software. This

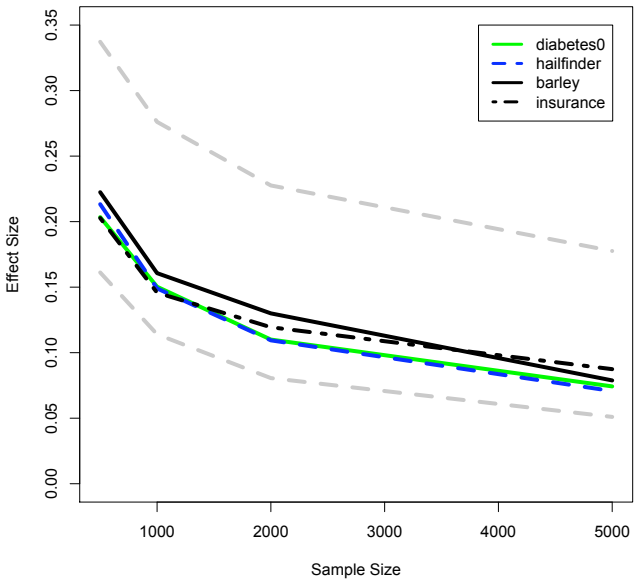


Figure 4.5: Results of cross-validation to select the best effect size parameter. The dashed grey lines indicate the outer limits considered during cross-validation. The minimum indicates the largest effect size where no tests are run and the maximum indicates the tests that are run under the rule of thumb threshold.

version of the code handles tests with zero degrees differently from the current code. Tests with zero degrees of freedom triggered a default decision to add an edge from the model. In later versions of the code, the decision was changed to omit the edge from the model to mirror the decision made in the TETRAD IV software package.

We considered three different corrections for false negative errors to determine the importance of each correction. The corrections I considered are listed in Table 4.3. The ideal baseline is using no correction at all for false negative errors meaning every possible test would be run. This is neither feasible due to runtime considerations nor sensible as conditioning on many variables would likely result in a conclusion of independence for all pairs of variables. As a baseline, I consider a weak correction which limits the degrees of freedom to be less than or equal to the sample size.

Table 4.3: Corrections for false negative errors.

Weak Correction	Permit tests with at least 1 instance per degree of freedom.
Rule of Thumb	Permit tests with at least 5 data samples per degree of freedom.
Power	Run tests with sufficient power (with estimated effect size parameter).
Necessary Path	Only condition on variables on a path.

Since I am evaluating the number of structural errors compared to the true network, I can only evaluate the corrections on networks with known structure. I found that the Power correction using the suggested effect size parameters was the only correction that resulted in a significant decreases in false negative errors (see Figure 4.6 and Table 4.4). On Alarm and Pathfinder, using the Power correction only resulted in an decrease at the lower sample sizes. The win95pts network is the exception and exhibits similar performance for all corrections, resulting in overlapping lines in Figure 4.6. This is a result of the unique structure of the network and the fact that win95pts consists of only binary variables. These two factors combine to run only tests with power above the threshold for all algorithms.

Using the Power correction also resulted in a significant increase in the number of false positive errors across all datasets; however, the total number was still fewer than the false positive errors obtained by using an unconstrained skeleton. Although the Power correction could be used in practice, the increase in false positive errors is likely prohibitive for most purposes. The primary purpose of the Power correction is to show that running only tests with high-statistical power results in a significant

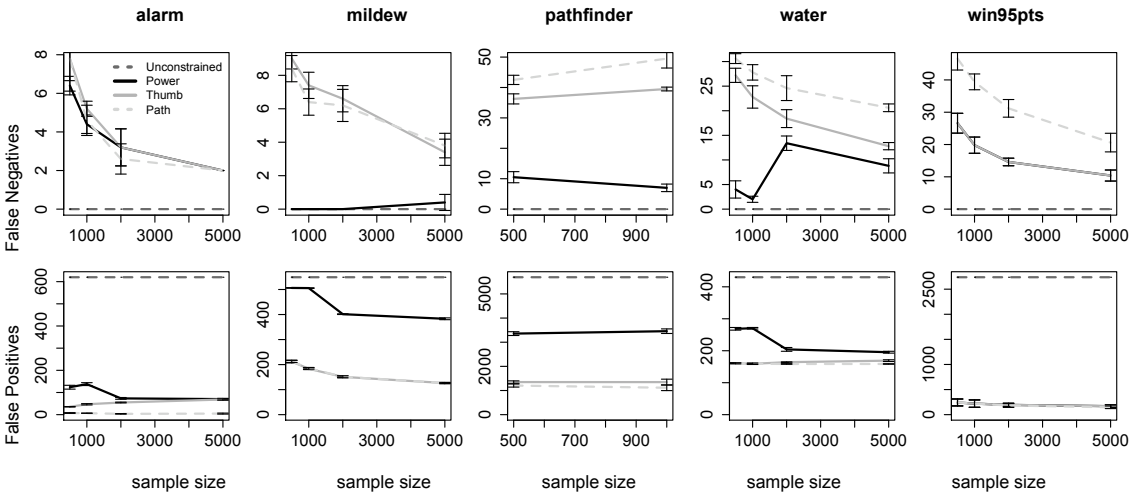


Figure 4.6: Comparison of the number of false negative and false positive errors for all the corrections. Error bars indicate 95% confidence intervals about the mean of 5 runs.

decrease in false negative errors. I explore alternative methods for improving power in Chapter 5.

4.7 Summary

In this chapter, I considered four types of errors made by constraint identification algorithms. These errors are false negative errors, statistical false positive errors, default false positive errors, and separating set errors. Using the custom implementation in the PowerBayes package to make it possible to count each type of error, I ran the FAS constraint identification algorithm on a range of datasets generated from known models. The results of these experiments show that false negative errors are the largest component of error on nearly all of the datasets considered. Further analysis shows that running tests with low statistical power is a primary cause of false negative errors. Therefore, further improvements in the accuracy of constraint-based algorithms will require new statistical tests and algorithmic approaches to improve

power and address false negative errors. Suggesting and evaluating candidate tests and approaches is the goal of the remainder of this thesis.

Table 4.4: Number of false negative and false positive (fn/fp) errors after applying corrections to the FAS algorithm. Bold text indicates a significant reduction in false negatives compared to the rule of thumb averaged across 5 training samples. Differences were significant at the 0.05 level using a one-sided t-test.

Dataset	Correction	500	1000	2000	5000
Alarm	Weak	7.8/35.6	5.2/46.8	3.2/54.8	2/68.4
	Rule of Thumb	7.8/35.6	5.2/46.8	3.2/54.8	2/68.4
	POWER	**6.4**/123.8	**4.4**/137/8	3.2/73.4	2/70.2
	RoT + Path	7.8/7.4	4.6/6.6	2.6/3.6	2/4.8
	POWER + Path	**5.4**/118.2	**3.6**/130	2.6/57.4	2/41.2
	GS	0/620	0/620	0/620	0/620
Mildew	Weak	13/124.6	10.4/96.8	8/78.4	4.4/64.6
	Rule of Thumb	9/212	7.4/183.2	6.6/151	3.4/126.2
	POWER	**0**/506.2	**0**/505.6	**0**/401.4	**0.4**/383
	RoT + Path	8.4/212.8	6.4/185.2	6.2/151.4	3.8/126.4
	POWER + Path	**0**/506.2	**0**/505.6	**0**/401.4	**0.4**/383
	GS	0/549	0/549	0/549	0/549
Pathfinder	Weak	73.6/1209.2	76.67/1052	-	-
	Rule of Thumb	36.25/1349	39.5/1348	-	-
	POWER	**10.5**/3356.5	**7**/3456.5	-	-
	RoT + Path	42.5/1204.75	49.5/1114	-	-
	POWER + Path	**10.5**/3359	**7**/3439.5	-	-
	GS	0/5691	0/5691	0/5691	0/5691
Water	Weak	27.2/161	22.8/159	18.4/164.2	12.8/169
	Rule of Thumb	27.2/161	22.8/159	18.4/164.2	12.8/169
	POWER	**4**/268.6	**2**/270.4	**13.4**/204.2	**8.8**/195.4
	RoT + Path	30.6/160.8	27.8/159.2	24.6/158.6	20.6/158.6
	POWER + Path	**4**/268.6	**2**/270.4	**13.6**/204.2	**9**/198.2
	GS	0/430	0/430	0/430	0/430
Win95pts	Weak	26.6/242	19.8/220.2	14.6/194.8	10.4/169
	Rule of Thumb	26.6/242	19.8/220.2	14.6/194.8	10.4/169
	POWER	26.6/242	19.8/220.2	14.6/194.8	10.4/169
	RoT + Path	46.6/245.6	39.4/220	31.2/182.8	20.6/146.8
	POWER + Path	44.6/246.2	37/221.4	29.6/183	19.4/147
	GS	0/2738	0/2738	0/2738	0/2738

CHAPTER 5

STATISTICAL APPROACHES FOR IMPROVING POWER

The focus of this chapter is evaluating statistical approaches for improving the power of hypothesis tests used for constraint identification. Since power is a statistical phenomenon, it would seem that statistical approaches would be the best approach for improving power. However, as the experimental results presented in this chapter show, statistical approaches for improving power often produce a net increase in errors. These results demonstrate a fundamental trade-off between false negative and false positive errors during constraint identification. In this chapter, I assess the strength of multiple approaches for improving the power of constraint identification, including a new constraint identification algorithm based on propensity score matching.

5.1 Introduction

I explore techniques for improving the statistical power of individual hypothesis tests in order to reduce the number of false negative errors made during skeleton identification. Since the power of an individual test depends on the significance level α, the rates of type I and type II errors are tied together. This is illustrated best by the error rates at the extremes. At one extreme, it is possible to eliminate all false negative errors by adding every edge and incurring the maximum number of false positives. At the other extreme, it is possible to avoid any false positive errors by not adding any edges and incurring the maximum number of false negative errors.

The challenge, then, is to improve the power of an individual hypothesis test without significantly increasing false positive errors.

In the remainder of the chapter, I evaluate three alternative statistical procedures for improving the power of individual hypothesis tests. These alternatives are (1) a χ^2 test with different parameters, (2) the Cochran-Mantel-Haenszel test, and (3) propensity score matching. With the exception of propensity score matching on one network, each of the strategies results in an overall increase in the number of errors. Both the χ^2 test and propensity score matching improve the power of constraint identification, resulting in a decrease in false negative errors; however, these gains in power are negated by the increase in false positive errors. These results show that the statistical approaches considered here, though using different strategies and varying in complexity, are not sufficient for reducing false negative errors due to low statistical power. Other strategies, such as the constraint satisfaction approaches described in the following chapters, are needed to improve the overall error rate.

5.2 Improving Power With the χ^2 Test

As described previously in Section 4.5, there are four components of a hypothesis test that determine statistical power. These components are sample size, effect size, degrees of freedom, and the significance level of the test [24]. The statistical power of a test can only be improved by adjusting at least one of the four factors. In the social sciences, the primary focus of statistical power analysis is determining how much additional data (increase in sample size) needs to be collected in order to guarantee adequate power [24]. In most applications of structure learning, however, gathering more data is difficult, expensive, or both, limiting the sample size to the available training data. The effect size is solely a characteristic of the data and cannot be adjusted before running the test, although reasoning about the expected effect size can help determine a suitable alternative hypothesis (see Section 4.5 for more details.)

There are two remaining options for improving the power of the χ^2 test: (1) limit the degrees of freedom of the tests that are run (adjust the power threshold) or (2) adjust the significance threshold. The degrees of freedom are an indicator of the number of free parameters in the model and, therefore, the amount of data needed to accurately fit the model. These two thresholds are shown graphically in Figure 5.1. Here I use empirical evaluations to demonstrate how adjusting the significance threshold both reduces false negative errors (improves power) and causes an increase in false positive errors. In addition, I consider two different strategies for adjusting the power threshold: one approach is varying the "rule of thumb" and the other uses the POWER correction to estimate a particular level of power.

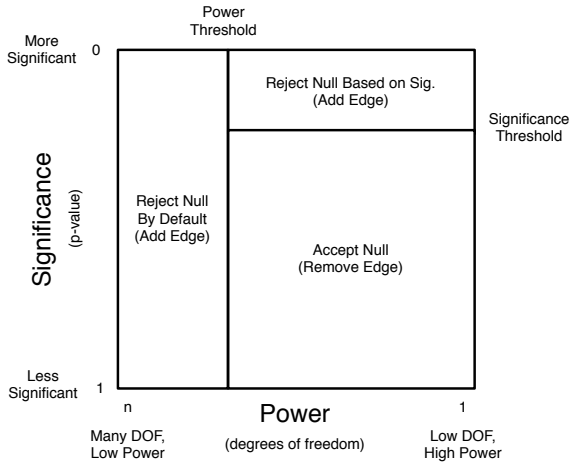


Figure 5.1: Pictorial representation of the two thresholds for determining power of a hypothesis test.

5.2.1 Varying the Power Threshold

The existing FAS algorithm for constraint identification uses a power threshold based on degrees of freedom ("rule of thumb") to determine whether to run a test. The motivation behind this threshold is to identify possible low-power tests and avoid running those tests. This is a default decision to conclude dependence instead of risking a false negative error due to low statistical power. The threshold is defined

by a ratio of the sample size (N) to the degrees of freedom of the test. The standard setting for the threshold is five ($\frac{N}{dof} \leq 5$). When the ratio falls below the threshold, the data is considered too sparse and the test is not run. Degrees of freedom are calculated using the method described by Spirtes et al. [87] and Fienberg [33].

To determine the effect of varying the degrees of freedom threshold, I ran the FAS algorithm with each threshold on data generated from the ALARM, INSURANCE, POWERPLANT, WATER and WIN95PTS networks. I considered the following values of the threshold: $\{5, 6, 7, 8, 9.10, 25, 50, 75, 100\}$. As shown in Figure 5.2, as the threshold increases, the rate of false positive errors dramatically increases as the number of tests that fall below the threshold also grows. Increasing the threshold in this way makes it more likely for a contingency table to be sparse enough to trigger a default decision. Changing this threshold also leads to a decrease in false negative errors, though this decrease is dwarfed by the increase in false positive errors. These experiments only consider networks that contain variables with low cardinality. However, the effects observed with low-cardinality variables are only magnified when high-cardinality variables are included and even more tests fall below the threshold for the default decision. For a given network, the distribution of degrees of freedom of the tests is fixed; thus, at larger sample sizes, fewer and fewer tests are affected by the threshold leading to the small changes in errors with larger sample sizes.

5.2.2 Varying the Significance Threshold

One approach for increasing the statistical power of the χ^2 test is lowering the significance threshold (increasing the p-value) that determines whether a test concludes dependence. This adjustment allows more tests to reject the null hypothesis and should result in a decrease in false negative errors, but also should result in an increase in false positive errors as more edges are added to the skeleton. To determine the effect of varying the significance hypothesis, I ran the FAS skeleton algorithm at

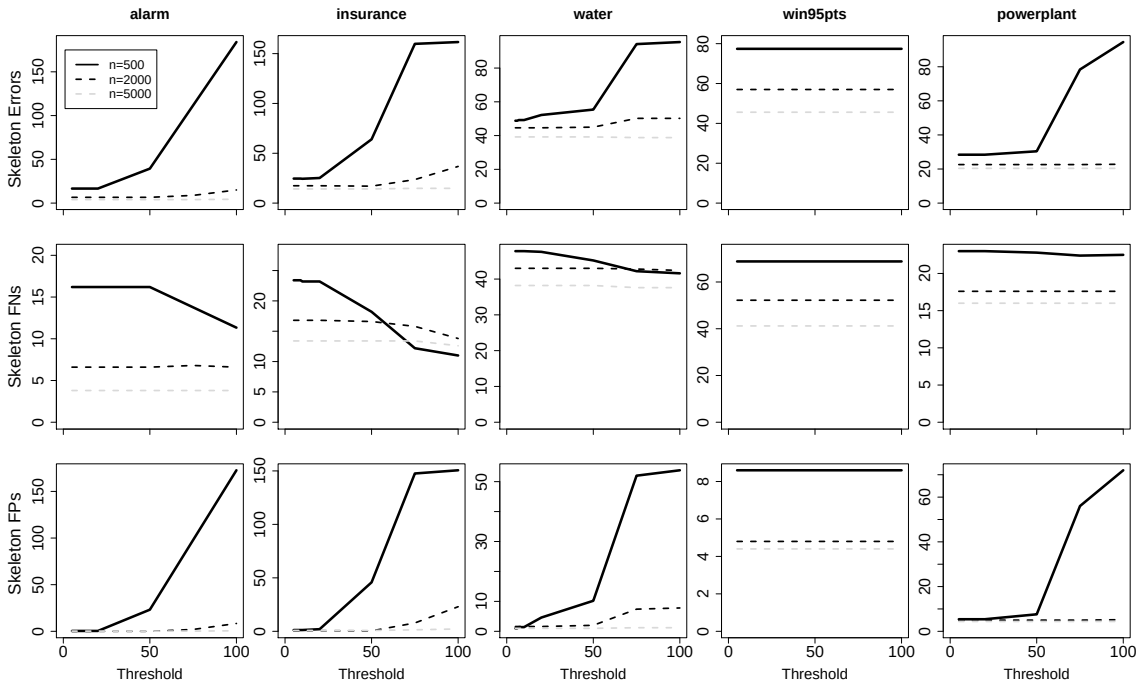


Figure 5.2: Skeleton errors as DOF threshold increases to improve power

each significance level on data generated from the ALARM, INSURANCE, POWERPLANT, WATER and WIN95PTS networks. I considered the following significance levels: $\alpha = \{0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35\}$. As shown in Figure 5.3, as α increases, the rate of false positive errors generally increases much faster than the rate at which false negatives decrease, leading to an overall increase in skeleton errors. While 0.05 is a reasonable general purpose threshold, on some networks other thresholds lead to fewer overall errors.

5.2.3 POWER Correction

The POWER correction described in Chapter 4 is another approach for limiting the degrees of freedom of the χ^2 hypothesis test. Using the POWER correction provides a uniform power threshold for all tests and all sample sizes. This is in contrast to the power threshold used above, which permits different power thresholds for different tests. The results of the POWER correction are shown in Section 4.6. Those show

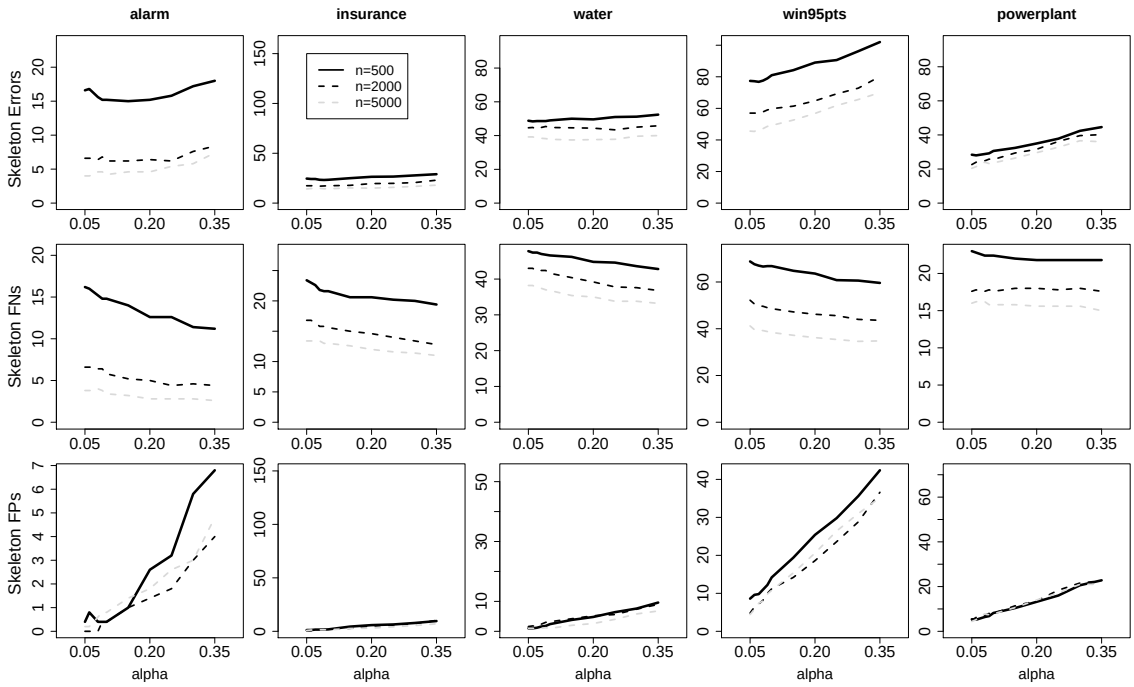


Figure 5.3: Skeleton errors as significance threshold decreases to improve power.

that while using the POWER correction does lead to a significant decrease in false negative errors, it also incurs many false positive errors.

5.3 Cochran-Mantel-Haenszel Test

The Cochran-Mantel-Haenszel (CMH) test is a statistical test of independence between two variables controlling for a set of possible confounding variables [23, 61, 62]. Unlike other tests of independence, the CMH test is based on a multiple hypergeometric probability model and does not require a strong assumption of no three-way (second-order) interactions [56]. These assumptions permit a model with fewer degrees of freedom and, consequently, increased power. For an $i \times j \times k$ table, the degrees of freedom for the CMH test is only $(i-1)(j-1)$ [4] whereas for the χ^2 test the degrees of freedom is typically computed as $(i-1)(j-1)k$ [87, 96]. However, the CMH test does tend to lose power in the presence of interactions [56].

Three-way interactions, however, are problematic in the general case of learning the structure of Bayesian networks. Since the true relationship among the variables is unknown prior to structure learning, three-way interactions may occur frequently, leading to decreased power in the CMH test. To determine whether the CMH test could be used as a replacement for the χ^2 test during constraint identification, I ran a set of empirical evaluations with the FAS algorithm where the χ^2 hypothesis tests were replaced with a generalized CMH test. Both tests are given the same contingency table computed from the data; the only difference was the statistic. I implemented this FAS-CMH algorithm in the PowerBayes software package using the CMH test available in the R statistics package and available using a Java to R interface. The results of these evaluations are shown in Figure 5.4. The results show that the FAS-CMH algorithm does consistently worse across the three test datasets. These results indicate that the possible gain in power from reducing the degrees of freedom is negated by possible losses in power due to three-way interactions.

5.4 Matching using Propensity Scores

Matching is a technique from the experimental design literature used to improve the reliability of causal inferences from data [82]. Rather than pooling the entire sample for statistical analysis, matching creates pairs of instances that are similar but have different values for the variables being tested for independence. This has the potential for increasing power by reducing the variance of the sampling distribution and improving the ability of the test to detect a significant effect. Propensity scores are a widely used mechanism for matching.

5.4.1 Overview of Propensity Scores

Rosenbaum and Rubin [79] introduce the propensity score as a technique for matching. Propensity scores are generally described from the perspective of an ad-

ministrator of a hypothetical clinical trial. The goal of the trial is to determine the efficacy of some new treatment (e.g., drug, surgery or other therapy) by applying that treatment to a sample of the population and comparing it to alternative treatments (often no treatment and/or the status quo) [52]. The gold standard is a randomized, controlled study; however, this ideal may not be possible due to ethical or practical reasons. Propensity score matching simulates a randomized, controlled study from observational data by using a statistical model of the propensity for treatment based on other observed variables, often called covariates. Units with similar propensity for treatment but different actual treatment values are matched together into pairs and used for analysis. The general propensity score approach has been defined in terms of binary treatment variables but extensions to other types of variables have been considered [59].

5.4.2 Propensity Score Matching for Constraint Identification

Although constraint identification does not match the standard uses of propensity score matching, it is possible to map constraint identification into a matching framework. The first task is determine the treatment variable T and the outcome variable O. Each hypothesis test determines the independence between two variables; thus, one variable is denoted treatment and the other outcome. The propensity score framework is defined with binary treatment variables. Consequently I will only consider Bayesian networks containing binary variables. Given a binary variable T, I define the least common value of T to indicate that treatment was applied ($T = 1$). The least common value must be used as the positive treatment value, otherwise matching would not be possible. The other variable value is designated to indicate that no treatment or the control condition was applied ($T = 0$).

Once treatment and outcome variables have be chosen, the next step is to learn a propensity score model to predict the probability of treatment given a set of covariates

X. A propensity score model is a conditional model of $P(T = 1|X)$ where, for Bayesian networks, X indicates all remaining variables once treatment and outcome are chosen. For this work, I used a random forest model to generate a propensity score for each instance [12], although any conditional probability model (e.g., linear regression or naive Bayes classifier) could be used instead. The random forest is learned using the entire training set and then applied to each individual instance to determine the probability that that row will receive treatment. This probability is computed by averaging the individual probability estimates of each of the trees involved in the random forest. For binary data, the distance between instances is the absolute value of the differences between the propensity scores of each instance.

Matching is performed using the distances between the treatment and non-treatment instances. The purpose of matching is to identify a set of non-treatment instances to be matched to the treatment instances that minimizes the distance between the two groups. I use a greedy matching algorithm that, for each treatment instance, finds the best remaining non-treatment instance. Greedy matching is not optimal [59], but is fast. The optimal bipartite matching problem is well-studied in the combinatorial optimization literature [73]. The first strongly polynomial algorithm is the Kuhn-Munkres algorithm, better known as the Hungarian algorithm [55, 69]. Optimal matching is ideal but quickly became intractable for larger sample sizes.

Given a set of matched pairs, statistical analysis is performed to determine the strength of correlation between the treatment and outcome variables. In the evaluation in Section 5.4.4, I considered the following three statistical approaches: paired statistics, stratification on propensity score, and pooling of the treatment and non-treatment instances. For binary data, McNemar's test is a suitable paired test [64]. The Stuart-Maxwell test is a generalization of McNemar's test for multi-way data [63, 90]. Stratification on propensity score is another common use of the propensity score methodology. Stratification bins instances with similar propensity scores to-

gether, then uses a conditional statistic to compute the correlation conditioned on the propensity score. Stratification also provides the opportunity to smooth over a poor match. Finally, the pooling approach uses the χ^2 statistic as in typical constraint identification but on a subset of the data defined by the matches. I compare the efficacy of each of the approaches below.

5.4.3 Trade-off between Sample Complexity and Power

Matching produces a reduction in sample size as instances not included in a matched pair are not included in the final analysis. This has two effects: reduction in variance (increasing power) and reduction in sample size (decreasing power). Matching will only be a benefit if the increase in power due to variance reduction exceeds the decrease in power due to sample size reduction. When treatment is rare, the resulting sample size is much smaller than the original training set. This is particularly a problem on datasets where many of the variables are highly skewed. To counter any possible reductions in power, I introduce an ensemble approach that combines both a χ^2 statistic and matching statistic to determine independence. If either test concludes dependence then the edge is maintained. Used this way, the matching approach can only improve the power of constraint identification.

5.4.4 Matching Evaluation

We evaluated the different approaches to matching on datasets generated from the WIN95PTS dataset and 25 binary datasets generated using the BNGenerator [48]. Evaluation was limited to these datasets since propensity score matching is designed for binary treatment variables. On WIN95PTS, I considered datasets with 500 and 2000 samples. On the synthetic binary datasets, I considered datasets with 500, 2000, and 5000 samples. There are three parts to the evaluation. First, I evaluated the different matching approaches on marginal (pairwise) accuracy alone. Marginal true positives indicate whether treatment and outcome are marginally independent

($T \perp\!\!\!\perp O|\emptyset$) in the true network. Marginal dependence indicates that a qualifying path, as defined by d-separation, between the two variables exists, and does not necessarily indicate that there is an edge between the two variables directly. The structural accuracy of the different matching approaches is shown in Figure 5.5. The stratification approach is the only approach to improve power over the standard FAS approach.

The next stage of evaluation incorporates the stratification matching approach into the full FAS algorithm. To run conditional independence tests that the full FAS algorithm requires, I add the stratification variable to the conditioning set when running a conditional independence test. Again, an ensemble approach is used; independence is concluded if both the χ^2 and matching statistic conclude independence. The results of these evaluations are shown in Figure 5.6. The full FAS algorithm using the matching statistic also leads to an increase in power, evidenced by the decrease in false negative errors. Like other statistical approaches described earlier in the chapter, matching also exhibits a trade-off between false negative and false positive errors. However, unlike varying the thresholds of the traditional χ^2 test, with matching it is possible to have the increase in power and reduce the false positive errors by increasing the significance threshold (lowering the p-value). The effects of varying the significance threshold used by the matching statistic is shown in Figure 5.7. On the synthetic binary networks, the matching statistic produces models with the same number of skeleton errors as the FAS algorithm but with fewer false negative errors and more false positive errors. On WIN95PTS, the increase in power provided by matching is offset by additional false positive errors, resulting in an overall increase in errors.

5.5 Related Work

5.5.1 Alternative Statistical Approaches

The most common alternative statistical approaches appearing in the literature are the Bayesian and mutual information tests of independence. As mentioned in Section 3.2.1, a Bayesian test is usually a Bayesian score such as BDeu [1, 27]. If adding the edge leads to an improvement in score, then dependence is assumed. This corresponds to adjusting the significance threshold to increase power as considered in Section 5.2.2. One advantage of the Bayesian approach is the ability to smooth with a prior to improve statistical power [27]. As with all Bayesian approaches, incorrectly choosing a prior can lead to biased results. In addition, averaging over multiple models can be computationally more intensive [58]. Mutual information has also been proposed as a method for improving statistical power [18]. Like the Bayesian approach, a test of mutual information is usually combined with a weak significance threshold that varies with both the sample size and the size of the test [18, 96]. The form of the mutual information statistic is closely related to G^2 statistic, and can be plagued by the same statistical issues as the G^2 statistic [47].

5.5.2 Reverse Multiple Comparisons

Recently, researchers have begun to view structure learning of Bayesian networks as a multiple comparisons problem (MCP) [57, 58, 94]. An MCP occurs when many hypothesis tests are considered simultaneously. Considering multiple tests increases the probability that a single test will exceed the significance threshold by chance alone, leading to an excess of type I errors. This type of multiple comparisons problem is prevalent in artificial intelligence algorithms [50]. Existing corrections for multiple comparisons focus on limiting the number of type I errors by adjusting the significance threshold to account for multiple tests [10, 38].

Constraint identification algorithms, such as FAS, also exhibit a reverse multiple comparison problem. A *reverse multiple comparisons problem* (RMCP) occurs when the goal of the multiple tests is to accept the null hypothesis (prove independence), rather than find significant correlations. In this situation, running multiple tests increases the probability of a type II error. In the FAS algorithm, a reverse multiple comparisons problem occurs both within a set of tests at a given separating set size and between sets of tests with different separating sets. Recall that the FAS algorithm runs all tests at a given separating set size (n) before considering tests at a larger set size. When $n \geq 1$, FAS tries all separating sets of size n in an attempt to prove independence. If a single test succeeds in proving independence, then the edge is removed from the model and the pair of variables is not considered again. The FAS algorithm also considers tests at different values of n in an attempt to prove independence. Both of these algorithmic choices contribute to an RMCP. Similar situations occur in other constraint identification algorithms such as MMPC or TPDA [18, 96].

Unlike traditional MCPs, where the rate of type I errors can be controlled by a single threshold α, it is difficult to analytically bound the rate of type II errors in RMCPs; the rate of type II errors β depends on multiple characteristics of the test. In addition to α, computing β (or statistical power, which is defined as $1 - \beta$) depends on the specification of an exact alternative hypothesis to pit against the null hypothesis. In situations where structure learning is required, the alternative hypothesis is rarely known with certainty, otherwise structure learning would not be necessary. Unfortunately, this means it is very difficult to improve the power of constraint identification by accounting for the RCMP.

5.6 Discussion

Statistical approaches for improving statistical power must account for the fundamental trade-off between false negative and false positive errors. In each of the approaches considered in this chapter, any decrease in false negative errors was accompanied by an increase in false positive errors, often resulting in a net increase of total number of errors. The one exception was propensity score matching on the synthetically generated binary networks. In this case, the gains in false negative errors offset the losses in false positive errors, resulting in no net change. One advantage the propensity score matching approach holds over χ^2 -based approaches is the ability to vary the significance thresholds to control the trade-off between false negative and false positive errors. This permits an increase in power due to matching and the ability to limit the number of false positive errors. Despite the promise of propensity score matching, the statistical approaches attempted here do not produce consistent net reductions in the the number of overall errors. Instead, other methods, such as constraint satisfaction, are needed to combine statistical and non-statistical improvements for reducing overall error. The first constraint satisfaction approaches for structure learning of Bayesian networks are described in the following chapters.

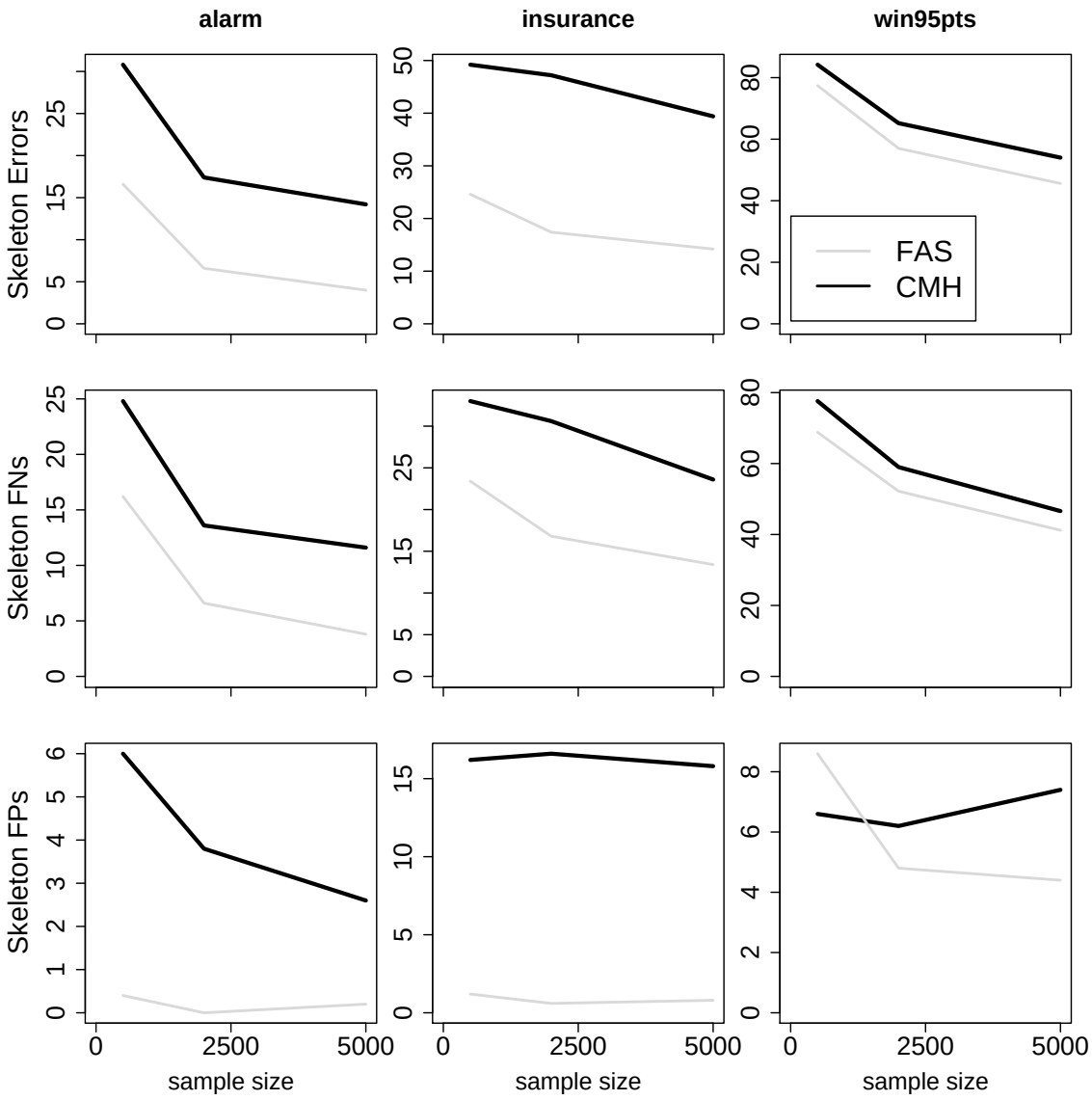


Figure 5.4: Skeleton errors made by the FAS algorithm after replacing the χ^2 test with the Cochran-Mantel-Haenszel test.

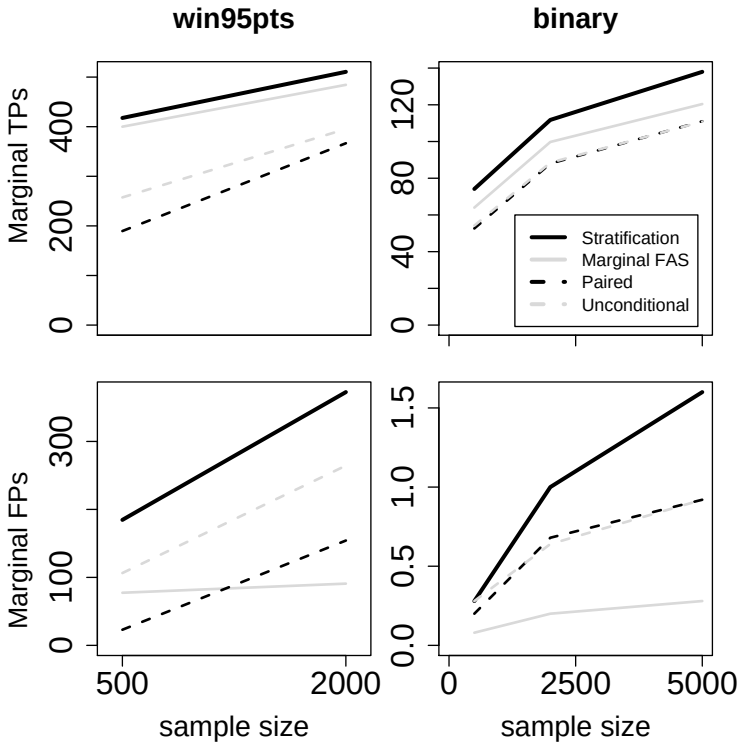


Figure 5.5: Structural accuracy of the different matching approach on marginal (pairwise) tests.

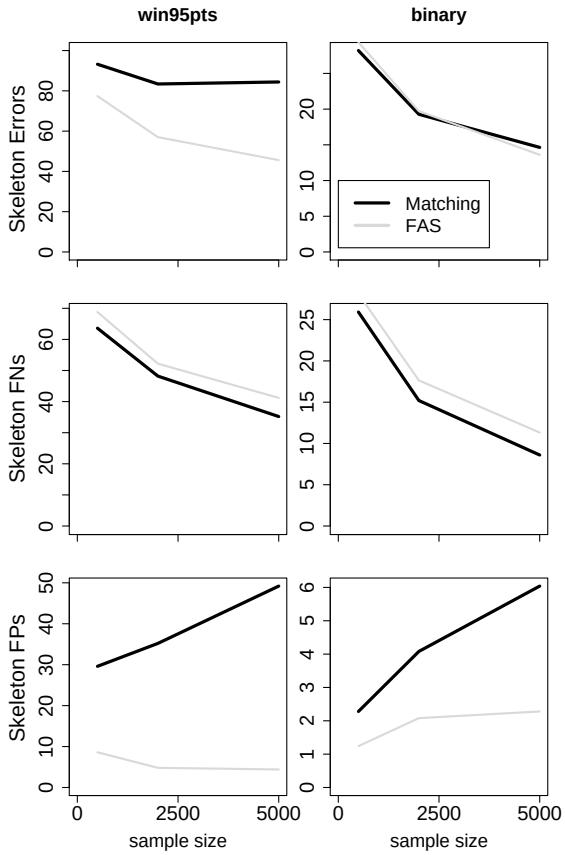


Figure 5.6: Skeleton errors using propensity score matching with a significance threshold of $p = 0.0001$.

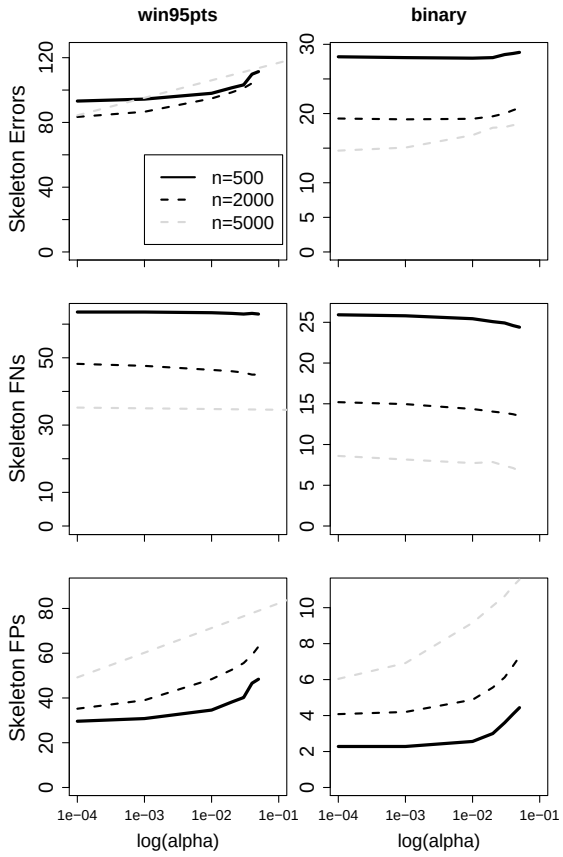


Figure 5.7: Skeleton errors using propensity score matching as the significance threshold varies.

CHAPTER 6

EDGE ORIENTATION AS CONSTRAINT OPTIMIZATION

Until now, the focus of the thesis has been reducing errors in the constraint identification phase of structure learning by improving individual hypothesis tests. However, the previous chapter demonstrated that statistical approaches alone are not sufficient to reduce errors in constraint identification algorithms. In the following two chapters, I describe new algorithms based on constraint satisfaction as alternative methods for improving structural accuracy.

First, I apply constraint satisfaction techniques to the edge orientation phase of constraint-based structure learning. In the edge orientation phase, a learning algorithm uses the learned constraints to form a fully directed Bayesian network. Due to errors in constraint identification, the constraints may not be consistent, making it impossible to find an orientation for the edges that satisfies all of the constraints. When faced with inconsistencies, existing edge orientation algorithms produce models that are not Bayesian networks. This is a result of a decision to consider constraints independently.

In this chapter, I introduce a new edge orientation algorithm based on constraint optimization. Constraint satisfaction approaches, like constraint optimization, use the fact that independence constraints are global path constraints that share individual edges in the graph. Constraint optimization exploits this interaction among the independence constraints to produce model structures that are guaranteed to be DAGs, but still have structural accuracy that meets or exceeds the structural accuracy of existing edge orientation algorithms. In addition, this new algorithm eschews

deterministic orientation rules in favor of powerful search techniques that find the structure which is most consistent with the constraints.

6.1 Introduction

Edge orientation is the process of taking the independence constraints identified during constraint identification and creating a fully oriented Bayesian network. Recall that the independence constraints can be decomposed into an undirected skeleton, indicating the binary edge decisions implied by the constraints, and the separating sets. The standard approach to edge orientation assumes that the skeleton and separating sets are fixed, and only the orientations of the edges can be changed in order to satisfy the constraints.

Unlike traditional constraint optimization problems, independence constraints are not atomic. There is no single operation that can be applied to the skeleton to satisfy a constraint. Instead, independence constraints are path constraints that depend on interactions among the orientations of possibly many edges. The interactions depend on the d-separation relationships between the variables involved with the constraint (see Section 2.1 for more details). In the worst case, changing the orientation of a single edge can affect the status of every constraint.

Due to errors in constraint identification, the learned constraints may not be consistent and cannot be satisfied simultaneously by a valid Bayesian network. When this situation occurs, edge orientation algorithms either produce a structure which is not a Bayesian network or produce valid Bayesian networks that do not satisfy all of the constraints. The only existing constraint-based orientation algorithm, which I call PC-Edge, uses the first strategy and can produce models with bidirected edges and cycles. I describe the PC-Edge algorithm in more detail in the following section. In the remainder of the chapter, I describe a novel, second algorithm for edge orientation, called Edge-Opt, based on a constraint optimization strategy that guarantees a

valid Bayesian structure and produces orientations that are no less accurate than the orientations produced by PC-Edge. Edge-Opt was first introduced by Fast and Jensen [30].

6.2 Edge Orientation in the PC Algorithm

The PC-Edge algorithm for edge orientation is derived from steps C and D of the PC algorithm [87]. The basis of the algorithm is the following theorem:

Theorem 1. *(Verma and Pearl [98]) Two DAGs are equivalent if and only if they have the same skeletons and same v-structures.*

A v-structure is a structure of edges of the following form: $X \rightarrow Z \leftarrow Y$. Assuming constraint identification is accurate, the learned constraints identify a skeleton that matches the skeleton of the generating distribution. All that remains for identifying an equivalent DAG to the generating distribution is accurately orienting the v-structures.

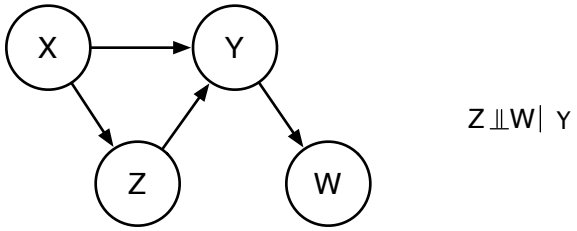
PC-Edge relies on the correspondence between conditional independence and graphical d-separation relations to orient the v-structures in a given skeleton. A *collider* is a v-structure $X \rightarrow Z \leftarrow Y$ such that $X \perp\!\!\!\perp Y$ and $Z \notin Sepset(X, Y)$. Colliders are the only d-separation structure that can be uniquely identified directly from conditional independence constraints. To orient colliders, PC-Edge first identifies all triples in the skeleton. A *triple* is a structure $X - Z - Y$ with no edge between X and Y. Each triple is then checked against the separating sets to determine if the conditional independence condition (i.e., $X \perp\!\!\!\perp Y$ and $Z \notin Sepset(X, Y)$) holds for that triple. If so, then the triple is oriented as a collider. Note that only the variables involved with the collider are considered; the remainder of the sepset is ignored. After all of the colliders have been oriented, an additional set of deterministic rules is applied to propagate the orientations to as many remaining undirected edges as possible [87]. These rules are often called Meek's Rules since they were formalized by Meek [65].

When the skeleton and separating sets are perfectly correct (i.e., learned in the sample limit), the PC-Edge algorithm is sound and complete [87]. However, when there are errors in constraint identification, PC-Edge often fails to return a valid Bayesian network. Since PC-Edge considers each constraint independently when orienting colliders, it is possible for bidirected edges or cycles to occur in the final network. Figure 6.1 shows a simple case where considering the constraints independently leads to a bidirected edge. In the example, the constraints both have empty separating sets, however, there is no DAG which is consistent with the skeleton and both constraints. Since the PC-Edge algorithm considers each constraint independently, it orients each collider according to the constraints and produces a model with a bidirected edge.

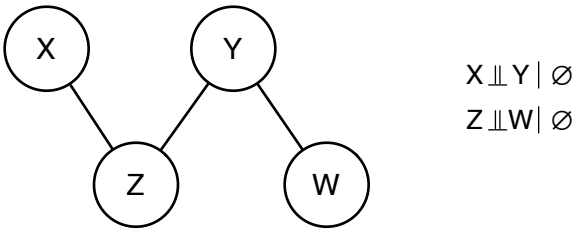
A bidirected edge can be interpreted either as an indicator of an unobserved variable influencing both endpoints of the edge or as an indicator of a possible error. A group of graphical models called ancestral graphs have been developed to deal with the former interpretation of bidirected edges [78]. Unfortunately, these models cannot be scored with existing penalized likelihood scores as there is currently no known parameterization of these models for discrete data [87]. This precludes the use of likelihood-based scores such as BDeu. Therefore, I do not consider that case in this thesis. As in the example described above, overlapping colliders occur when the separating set does not contain the proper d-separators. Since running low-power statistical tests during constraint identification often results in smaller separating sets, the results in this thesis provide evidence that bidirected edges are quite likely to occur as a result of an error and should be avoided, if possible.

6.3 Constraint Optimization Algorithm

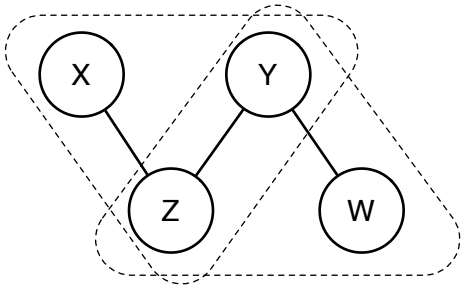
In this section, I describe the Edge-Opt algorithm for edge orientation. In contrast to previous approaches, the Edge-Opt algorithm always produces a valid



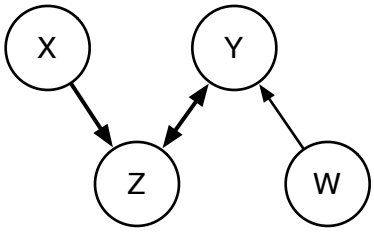
(a) Original network



(b) Skeleton and separating sets with a false negative error and separating set error from low-power tests.



(c) Overlapping triples as a result of errors.



(d) Orienting colliders independently.

Figure 6.1: Anatomy of a bidirected edge. Since the PC-Edge algorithm considers constraints independently, overlapping colliders are oriented with a bidirected edge as a result of errors in the constraints.

Bayesian network model. This is achieved by considering all constraints jointly and using a constraint optimization strategy to search over possible orientations to find the orientation which satisfies the maximum number of constraints. The Edge-Opt algorithm is the first edge orientation algorithm based on constraint optimization and, to my knowledge, only the second edge orientation algorithm to use the constraints for edge orientation.

The Edge-Opt algorithm is a greedy search algorithm that starts from a random orientation of the skeleton S. For each triple appearing in the skeleton, Edge-Opt applies the new ToggleCollider edge operator to produce successors to the current structure. Each successor is then scored based on the number of constraints that are satisfied by that structure. The successor which satisfies the most constraints is set as the current state and the algorithm repeats until no application of the ToggleCollider operator results in more constraints to be satisfied. This is a search procedure over edge orientations of a fixed skeleton, no edges can be added or removed from structure. Pseudocode describing Edge-Opt is found in Algorithm 3. Ideally, I would like to use a complete search algorithm to find the best orientation of edges appearing in S. However, this is not feasible with reasonably sized networks, as the number of possible orientations grows exponentially with the number of edges appearing in S.

In the following sections, I will describe the details and innovations that went into the Edge-Opt algorithm. These include how to determine whether a constraint is satisfied, details on the ToggleCollider search operator, and strategies considered for avoiding local optima. In addition, I provide an analysis of the computational complexity of the algorithm.

Algorithm 3 Edge Orientation via Constraint Optimization

```
procedure EDGE-OPT(Constraints C, k, numRestarts)
                                        ▷ Get skeleton and dependency model.
    (S, M) = C
    D* = nil                                          ▷ Global best structure.
    for i = 1 to numRestarts do
        D = GET-RANDOM-ORIENTATION(S)
        while True do
            N = GET-SUCCESSORS(D, M)
            if |N| == 0 then
                break
            end if
            D' = GET-BEST-SUCCESSOR(N, M)
            if NUMSAT(D', M)>NUMSAT(D*, M) then
                D* = D'
            end if
            D = D'
        end while
    end for
    return D*
end procedure
procedure GET-SUCCESSORS(D,M)
    T = GET-TRIPLES(D)
    N = {}
    for t ∈ T do
        n = TOGGLE-COLLIDER(t,D)
        if NUMSAT(n,M) > NUMSAT(D,M) then
            N = N ∪ n
        end if
    end for
    return N
end procedure
procedure GET-BEST-SUCCESSOR(N, M)
    N = SHUFFLE(N)
    size = MAX(1, k * |N|)
    N' = {N_1, ..., N_size}
    n* = nil
    for n ∈ N do
        if NUMSAT(n, M)>NUMSAT(n*, M) then
            n* = n
        end if
    end for
    return n*
end procedure
```

6.3.1 Definition of Constraints

The first step to creating an algorithm for edge orientation based on constraint optimization is defining the constraints to be satisfied. As described elsewhere in this thesis, the constraints being satisfied are independence assertions of the form $(X \perp\!\!\!\perp Y|Z)$ indicating that "X is independent of Y given Z." The edge is included in the skeleton if there is no independence constraint between the endpoints of the edge. If the constraints represent a dependency model, then satisfying all of the constraints will result in a Bayesian network compatible with that dependency model [74]. Due to errors in the constraints, it is unlikely that any Bayesian network will satisfy all of the constraints; therefore, edge orientation via edge optimization can be viewed as selecting the largest dependency model from the set of constraints that is consistent with a Bayesian network.

If we make the assumptions of Spirtes et al. [87]—no latent variables, the distribution represented by the training data is faithful to a DAG, and the statistical decisions made from the data are correct (e.g., made in the sample limit)—then it follows directly from the definitions of a dependency model and a Bayesian network that this constraint formulation is sound. A DAG which satisfies all of the constraints will be equivalent to the structure of the generating distribution. Therefore, like PC-Edge, Edge-Opt is sound in the sample limit since both algorithms use the independence constraint identified from skeleton identification.

6.3.2 Determining whether a Constraint is Satisfied

Due to the correspondence between conditional independence and d-separation, a natural definition for constraint satisfaction is that a constraint $(X \perp\!\!\!\perp Y|Z)$ is satisfied by a given structure if X and Y are d-separated given Z in that structure. This definition alone is not suitable for our purposes as d-separation is defined as the absence of a path with particular characteristics (c.f. Section 2.1). Therefore, it is

possible to create a structure to satisfy a d-separation statement in two ways: (1) create a path with the particular characteristics or (2) do not create any paths at all. Under the natural definition for constraints, the empty network (no edges) would satisfy every independence constraint.

To avoid this situation, I define an independence constraint ($X \perp\!\!\!\perp Y|Z$) to be satisfied by a DAG D if Z is a minimal d-separator of X and Y in D. A set Z is a minimal d-separator if Z is a d-separator and no proper subset of Z is also a d-separator [93]. Minimality guarantees that Z is both necessary and sufficient to d-separate X and Y. Defining constraints in terms of minimal d-separation requires that every variable in the separating set appear on a path between X and Y. The collider constraints used by PC-Edge do not consider the entire sepset, only checking whether a single variable appears in the set. To check whether a constraint holds in D, I use Algorithm 1 of Tian et al. [93]. The minimal d-separation algorithm relies on the algorithms of Geiger et al. [37] for checking d-separation in a subset of the original graph. The check for minimal d-separation runs in time proportional to $|Z|$ times the number of edges in the graph.

To choose among possible successors, I count the number of constraints satisfied by each successor and choose the successor that satisfies the maximum number of constraints. In the case of ties, I choose the successor with the higher BDeu score. This scoring scheme gives each constraint an implicit unit weight. In addition, I add a hard acyclicity constraint to guarantee that the best successor is always a DAG.

6.3.3 ToggleCollider Search Operator

Every DAG D that is a member of a particular equivalence class shares the same d-separation relationships with every member of the class [21]. Since d-separation relationships correspond to conditional independencies, it is impossible to improve the number of independence constraints satisfied without changing the d-separation

relationships. Therefore, the most efficient method for searching orientation is to search over equivalence classes of DAGs and not just search in DAG space since not every single edge reversal results in a change in equivalence class.

Edge orientation is unique because the skeleton is fixed and only the edge orientations can change. Existing methods for searching over equivalence classes assume a variable skeleton and are designed to maximize a likelihood score, not constraint satisfaction [20, 91]. To make efficient search over orientations possible, a search operator is needed to search over equivalence classes with a fixed skeleton.

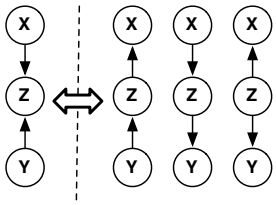


Figure 6.2: The Toggle-Collider operator

To address this need, I created the ToggleCollider search operator for edge orientation. From Theorem 1, two DAGs are equivalent, i.e., in the same equivalence class, if they have the same skeleton and the same v-structures (colliders). Since the skeleton is fixed in edge orientation, the only way to move between equivalence classes is to change the v-structures. Consequently, each successor state generated by the ToggleCollider operator differs from the current search state in the number of v-structures. This is achieved in the following manner. For every triple of variables in a DAG D, Toggle-Collider either makes the triple into a collider (right to left in Figure 6.2) or breaks the existing collider in each of three possible ways (left to right in Figure 6.2). These operations correspond to removing or adding a conditional independence relation to graph, respectively. Note that changing one collider may introduce one or more additional colliders. Since the successors states contain the same skeleton but a different set of colliders from D, the successor states are not

in the same equivalence class as D (though multiple successors may be in the same equivalence class) [98]. The size of the search space depends on the number of triples in S.

6.3.4 Avoiding Local Optima

As with all greedy search strategies, it is possible for the search process to get stuck in local optima. I incorporate two strategies for avoiding local maxima: a k-greedy approach [70] and random restarts. The k-greedy strategy consists of choosing the successor randomly from a fraction of the set of improved successors as defined by $k, k \in (0, 1]$. If $k = 1$, all of the successors are considered and pure greedy search is performed. If k is sufficiently small, then a successor is chosen at random. I found that the choice of k didn't affect the performance, and for all of the experiments described in this paper I used $k = 0.5$. The results of the parameter search are shown in Figures 6.3 and 6.4. I observed that only a small number of random restarts were sufficient to produce structural accuracies that were equivalent or better than PC-Edge. When I ran Edge-Opt alone, I used $numRestarts = 25$; for runs as part of constraint relaxation in Chapter 7, I used $numRestarts = 3$.

6.3.5 Computational Complexity

The computational complexity of the Edge-Opt algorithm depends on the characteristics of the learned constraints. First, the number of minimal d-separation checks required to evaluate a possible orientation depends on the number of independence constraints. Since there is at most one constraint for each pair of variables, the number of possible independence constraints is $\mathcal{O}(n^2)$ where n is the number of variables. Second, the number of states to evaluate depends on the branching factor of the search process. This process is $\mathcal{O}(T)$, where T is the number of triples in the

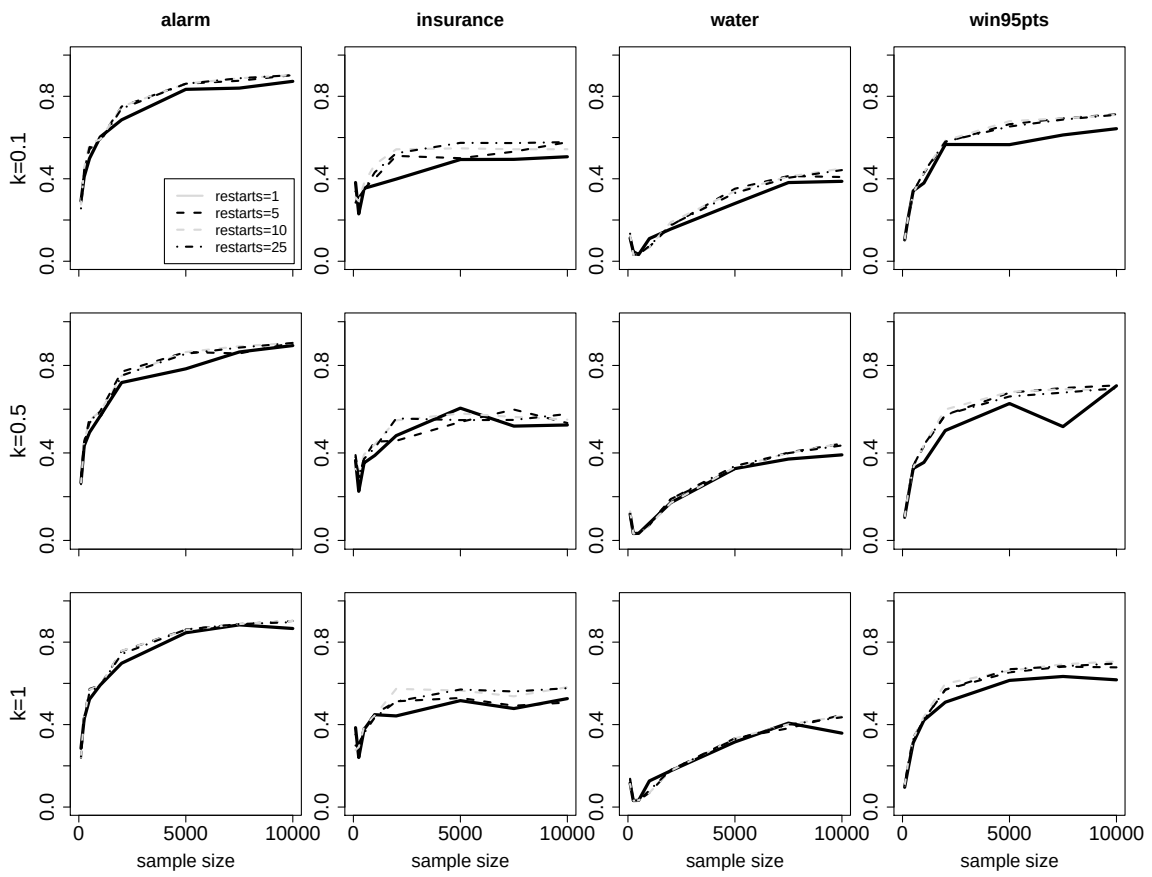


Figure 6.3: Comparison of Compelled F-Measure across number of random restarts for each value of k.

learned skeleton. Figure 6.5 shows the growth of the number of triples as the sample size grows.

Intuitively, the growth of these two characteristics are inversely related, though the actual relationship depends on the data. In general, as the number of independence constraints grows, both the number of edges in the skeleton and the number of triples decreases. However, both quantities also grow as the number of variables grows.

6.4 Evaluating Constraint Optimization

6.4.1 Structural Accuracy

To evaluate EDGE-OPT as a replacement for the PC-EDGE algorithm, I compared the structural accuracy of models produced by both our constraint optimization approach and the PC-EDGE algorithm. Evaluating structural accuracy requires data generated from a known structure. To satisfy this requirement, I trained on data with a range of sample sizes generated from the following networks drawn from real-world domains: ALARM, INSURANCE, POWERPLANT[1], WATER, and WIN95PTS[2]. I also trained on data generated from 25 random networks created using the BN-Generator[3] software [48]. I used the following parameter settings: *nNodes* chosen uniformly between 15 and 25, *minInDegree*=4, *maxOutDegree*=5, *maxVal*=5. The sample sizes I considered for ALARM, INSURANCE,WATER and WIN95PTS were $n = \{250, 500, 1000, 2000, 5000, 7500, 10000\}$ and for Powerplant and the synthetic networks I considered $n = \{500, 5000, 10000\}$.

We compare the structural accuracy of the PC-EDGE and EDGE-OPT algorithms using compelled F-measure and the structural Hamming distance (SHD) metric (Figure 6.6). Compelled F-measure is an indicator of the correctness of the causal claims made by the learned structure (higher is better). SHD is an overall measure of structural accuracy and indicates the number of deviations between the learned and true models (lower is better). EDGE-OPT performs significantly better on compelled F-measure ($p = 0.01$) than PC-EDGE on three of the five scenarios I considered (POWERPLANT, WIN95PTS, and SYNTHETIC), and is statistically indistinguishable in the other cases. The difference in performance on WATER could be considered

[1]Available from `http://bndg.cs.aau.dk/Bayesian_networks/powerplant.net`

[2]Alarm, Insurance, Water, and Win95pts are available from the Bayesian Network Repository: `http://compbio.cs.huji.ac.il/Repository/`

[3]`http://www.pmr.poli.usp.br/ltd/Software/BNGenerator/index.html`

weakly significant as $p = 0.099$. SHD is indistinguishable across all five networks considered. Performance is averaged over 5 training runs on each real network, and averaged across all 25 structures for the synthetic data. To test the significance of the differences between the learning curves, I used the randomized ANOVA approach developed by Piater et al. [77] with 1000 sample runs. These results indicate that constraint optimization is a suitable replacement for the PC-Edge algorithm within our constraint relaxation algorithm, which requires an edge orientation algorithm that is guaranteed to produce a DAG.

6.4.2 Likelihood

For datasets with unknown structure, it is impossible to evaluate the structural accuracy of a learning algorithm. Although Edge-Opt does not optimize for likelihood, I present likelihood of the learned models on common evaluation datasets for Bayesian network structure learning. I considered the following datasets in these experiments: Credit[4], COIL[5], and Letters[6]. On datasets with both training and test data, I pooled both collections into a single large dataset. I used five-fold cross-validation to create five training/test splits on this pooled dataset. Results of running these algorithms are shown in Table 6.1. As expected, the likelihood of the other models significantly outperforms the loglikelihood of the Edge-Opt algorithm. Significance calculations were performed using a paired t-test with a significance level of $p = 0.05$.

---

[4]Available from `http://www.stat.uni-muenchen.de/service/datenarchiv/kredit/kredit_e.html`

[5]COIL is available from the UCI KDD Archive `http://kdd.ics.uci.edu/`

[6]Letters is available from `http://www.autonlab.org/autonweb/15958.html?branch=1&language=2`

Table 6.1: Likelihood on datasets with unknown structure. **Bold** indicates a significant improvement over Edge-Opt.

	COIL	Credit	Letters
Edge-Opt	-26656.62	-14852.90	-181130.53
PC-Hybrid	**-26432.79**	**-13398.88**	**-159258.96**
Greedy Search	**-26347.85**	**-13465.42**	**-158395.92**

6.4.3 Runtime

With the current implementation, the Edge-Opt algorithm guarantees a DAG and improved structural accuracy in exchange for increased runtimes. Figure 6.8 shows the runtime results of the Edge-Opt algorithm on the runs described in Section 6.4.1. The Edge-Opt algorithm with 25 restarts is considerably slower than PC-Edge and slower than other popular approaches for learning the structure of Bayesian networks. However, based on the results of the parameter search presented in Section 6.3.4, similar performance could be achieved with many fewer restarts, reducing the overall runtime.

The majority of the additional time is spent on the minimal d-separation checks for each of the constraints for each of the successors. Although the structure of the graph remains the same between constraints, the algorithm for checking minimal d-separation does not incorporate information from previous runs. Therefore, the runtimes reported in Figure 6.8 are an upper-bound as it should be possible to cache the results of the minimal d-separation checks to save time when running multiple runs, as happens in the Edge-Opt algorithm. Developing a d-separation cache is beyond the scope of this thesis, but would be interesting to explore in the future.

6.5 Related Work

The Edge-Opt algorithm is inspired by but distinct from traditional constraint satisfaction problems (CSP) that are a staple of artificial intelligence [28, 35]. Both techniques rely on search to find satisfying assignments; however, the Edge-Opt algorithm searches in the space of Bayesian networks and traditional CSP search in the space of constraints. The operators used to search in the space of Bayesian networks are the same as the operators for search in directed graphs. Directed edges can be added, deleted, or removed from the graph at each step. In the context of constraint optimization, these operators are not atomic with respect to the independence constraints; a single edge operator can affect the status of many constraints, unlike operators in traditional CSPs. Because of this overlap, the constraint graph for Bayesian networks is fully connected. Therefore, optimizations for CSPs that rely on the structure of the constraint graph, such as forward checking, do not apply here.

Weighted constraint optimization is a related optimization problem considered in the CSP literature [28, 35, 72]. In place of the standard boolean constraints, weighted constraint optimization problems specify a set of weights along with the constraints. Rather than attempting to satisfy all of the constraints, the goal of weighted CSP is to optimize the sum of weights of the satisfied constraints. I considered both p-value and estimated power of the test as possible weightings for the Edge-Opt algorithm. Both of those approaches (and related approaches) are inappropriate for improving the structural accuracy of constraint optimization for Bayesian networks. Since low statistical power is the largest source of error in constraint identification, both the p-value and statistical power are non-linear in the constraints to be satisfied. High statistical power (p-value) always indicates a strong constraint; however, low statistical power can indicate either a true independence or an inability to detect a dependence in the data. Therefore, maximizing for statistical power results in a lower probability of inaccurate constraints being corrected by the search algorithm.

Refining the orientation produced by PC-Edge is an alternative strategy for addressing errors. Abellan et al. [1] use unconstrained greedy search to refine the model produced by the PC algorithm. This refinement does not consider either the skeleton or the sepsets in determining the final orientation. Badea [8] and Steck and Tresp [89] both describe *post hoc* approaches for revising the learned structure based on the inconsistency of the constraints. Both of these approaches operate by identifying inconsistencies and adding ambiguity to the model in the form of undirected edges to address those consistencies. Neither approach attempts to optimize the number of constraints satisfied by the structure.

Finally, there are other edge orientation algorithms that are applicable under different assumptions than those taken in this work. A related algorithm to PC-Edge, called FCI, is sound and complete with unobserved, or latent, variables and in the presence of both latent variables and selection bias [86, 87]. This is an interesting problem but beyond the scope of the current work.

6.6 Discussion

The Edge-Opt algorithm is an asymptotically sound approach for edge orientation in constraint-based algorithms. Unlike the PC-Edge algorithm, Edge-Opt always produces a DAG structure. This makes using traditional scoring metrics such as BDeu or loglikelihood a possibility for scoring learned structures. Despite the additional guarantee of producing a DAG, the Edge-Opt algorithm maintains the structural accuracy of the PC-Edge algorithm, and in certain instances is more accurate. The Edge-Opt algorithm is only the second algorithm to fully utilize independence constraints for edge orientation.

Due to errors in constraint identification, the constraints are typically not consistent and cannot be represented by a DAG. The PC-Edge algorithm and Edge-Opt algorithm embody two different design choices for dealing with this situation. PC-

EDGE effectively assumes that the constraints are correct and the assumption that the distribution can be represented by a DAG is incorrect. PC-EDGE, therefore, chooses to produce structures that are consistent with the constraints, but that are not a DAG. In contrast, EDGE-OPT effectively assumes that the constraints can be incorrect and that the assumption of a DAG model should be maintained. The results presented in this chapter show that both approaches produce models with similar accuracy; however, the approach taken by EDGE-OPT has some additional advantages over PC-EDGE, namely always produces a DAG structure, which I will capitalize on in the next chapter.

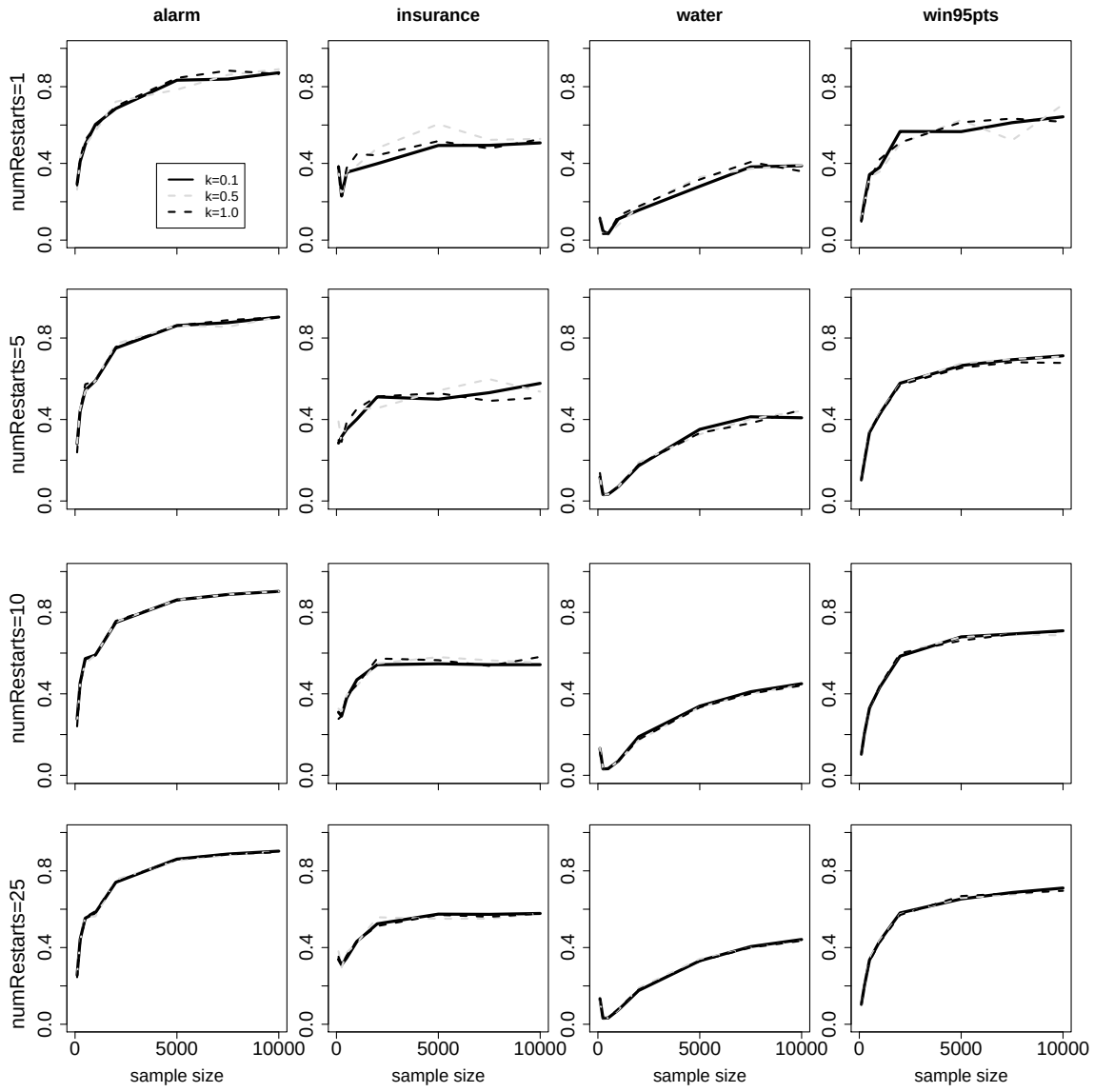


Figure 6.4: Comparison of Compelled F-Measure across values of k for each number of random restarts.

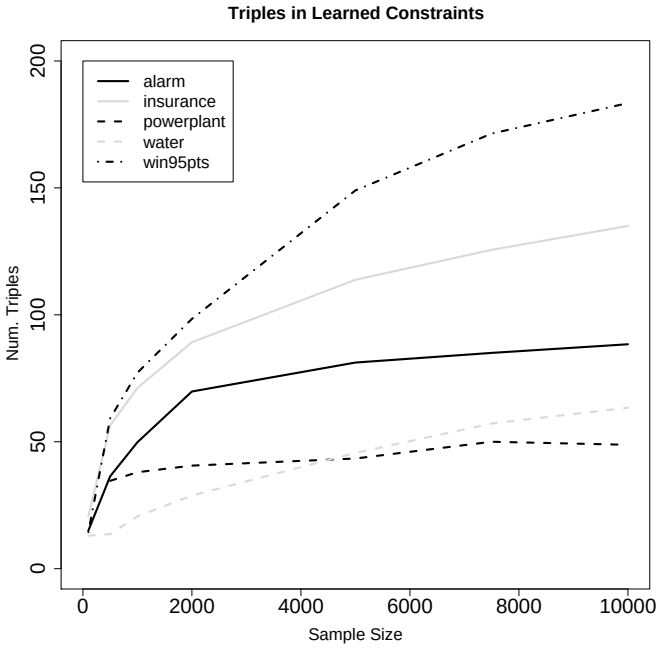


Figure 6.5: Number of triples in the learned constraints as sample size grows. The increase in the number of triples can be accounted for by the increase in statistical power as sample increases.

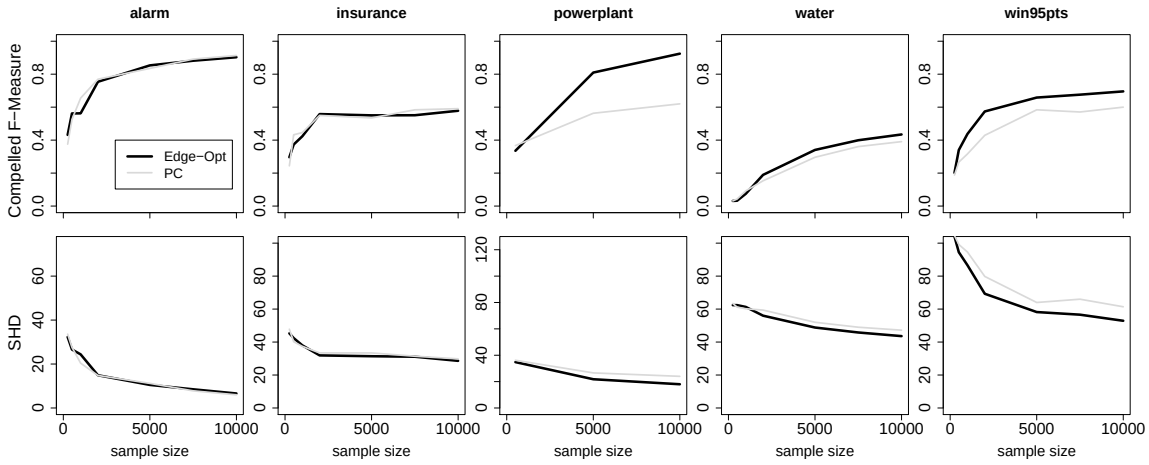


Figure 6.6: Compelled F-measure and structural Hamming distance (SHD) rates of PC-Edge and Edge-Opt. Differences are significant at $p = 0.01$ on Powerplant and Win95pts networks and statistically indistinguishable in the other cases.

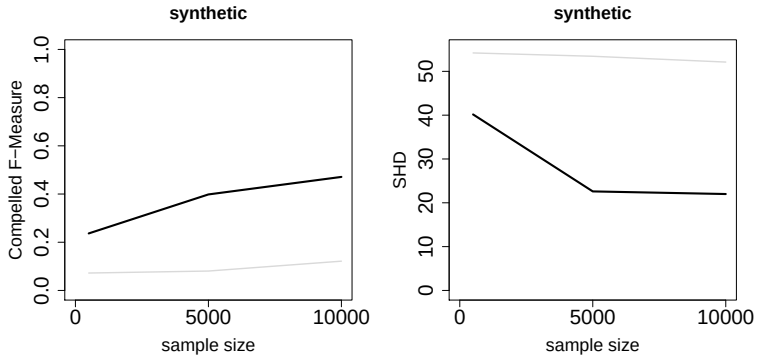


Figure 6.7: Compelled F-measure and structural Hamming distance (SHD) rates of PC-Edge and Edge-Opt on datasets generated using the BNGenerator package. Differences are significant at $p = 0.01$ on both metrics.

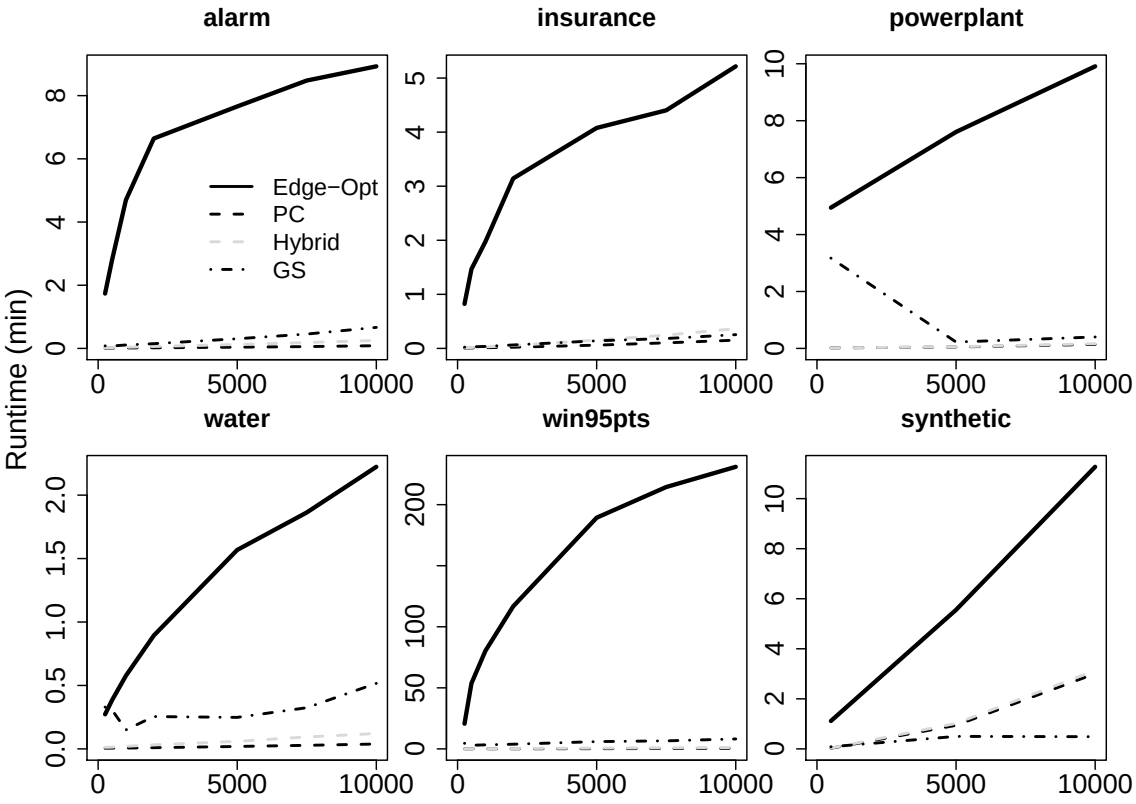


Figure 6.8: Runtime of Edge-Opt algorithm with 25 restarts compared with other algorithms.

CHAPTER 7

CONSTRAINT RELAXATION

Constraint relaxation is a new paradigm for learning the structure of Bayesian networks. Unlike existing constraint-based algorithms, constraint relaxation unifies constraint identification and edge orientation into a single search procedure. Combining the two phases of structure learning provides an opportunity to reduce errors in the constraints and improve overall structural accuracy. The RELAX algorithm described below is the first algorithm to use constraint relaxation for learning the structure of Bayesian networks.

7.1 Introduction

Existing constraint-based algorithms run constraint identification independently from edge orientation, only passing the constraints to edge orientation and allowing no information to be passed back to constraint identification. Unifying both phases into a single search procedure provides an opportunity to improve the correctness of the learned constraints and the structural accuracy of the overall model. The independence constraints used in constraint-based algorithms are global constraints, providing information about paths between variables rather than individual edges. Although the constraints are global, they are learned using only local tests. By unifying the two phases, it makes it possible for the global structure from orientation to influence the determination of the constraints. Since the original constraints are likely to contain errors, the constraint optimization approach described in Chapter 6

is not enough to improve structural accuracy as satisfying the (incorrect) constraints will still produce incorrect structure.

Constraint relaxation is the first approach for learning the structure of Bayesian networks that searches over constraints and orientations simultaneously. A unified search procedure provides the opportunity to improve the correctness of the learned constraints by "relaxing" or ignoring constraints that have a negative effect on structural accuracy. In Section 7.3, I describe RELAX, the first algorithm for constraint relaxation for learning the structure of Bayesian networks. RELAX first appeared in my prior work [30]. To make constraint relaxation possible, RELAX is a synthesis of three major strategies for learning the structure of Bayesian networks into a single algorithm. RELAX is a hybrid algorithm that combines constraint-based and search-and-score algorithms into a single algorithm and uses the learned independence constraints in conjunction with a likelihood score to orient edges in a Bayesian network [83, 96]. In addition, RELAX is a refinement algorithm. Refinement algorithms use complementary approaches for structure learning to refine the results of a constraint-based algorithm [1]. In the following sections, I provide additional background on both hybrid and refinement algorithms, then describe the RELAX algorithm in detail followed by experimental evaluation.

7.2 Background

7.2.1 Existing Hybrid Algorithms

Hybrid algorithms for learning the structure of Bayesian networks combine approaches from both the constraint-based and search-and-score paradigms. Constraint-based algorithms use only local hypothesis tests to learn the constraints, and as a result, can be quite fast. Search-and-score techniques use a penalized likelihood score and not hypothesis tests to consider all dependencies that increase the score, not just those indicating significant correlations. This flexibility permits the exploration

of structures which are "nearby" to those structure which satisfy many constraints. Since search-and-score algorithms are typically scoring a global structure, they can be less efficient than constraint-based algorithms. Hybrid algorithms use constraint-based algorithms to provide a boundary for the search procedure in an attempt to combine the efficiency of constraint-based approaches with the flexibility of a search-and-score algorithm.

The Max-Min Hill Climbing (MMHC) algorithm is a hybrid algorithm which uses an undirected skeleton as a boundary for the search procedure, followed by a greedy search procedure that is restricted to edges appearing in the skeleton [96]. Although the undirected skeleton is learned using a constraint identification algorithm, MMHC disregards the separating sets and only considers edges that appear in the skeleton. MMHC uses a greedy search procedure that starts from an empty network and then adds, deletes, and reverses edges until the scoring metric cannot be improved. Only edges appearing in the skeleton can be added to the final network; however, since the search starts from the empty network, not all edges appearing in the skeleton will be added to the final model. Consequently, MMHC allows corrections to false positive errors in the constraints by excluding an edge that appears in the skeleton from the final model, but cannot address false negative errors as no edge omitted from the skeleton can be included in the final model.

Instead of an undirected skeleton as a boundary, Singh and Valtorta [83] introduce a hybrid algorithm that uses a constraint-based approach to learn an ordering over the variables that can then be used as an input to a greedy search algorithm such as K2 [25], which requires variable ordering as an input. Given a variable ordering, a structure which maximizes the scoring metric can be found in polynomial time [25, 92].

The hybrid algorithms described above both use constraint-based algorithms first to identify a boundary on the greedy search. Acid and de Campos [2] describe an

alternative hybrid approach that searches over model structures and, for a given structure, checks whether the independence relations implied by the structure hold in the data. This is not a true constraint-based approach as the independence tests do not constrain the possible model structure. Instead, the independence tests are used to score competing structures.

7.2.2 Existing Refinement Algorithms

Refinement algorithms run a constraint-based structure learning algorithm, such as PC, to completion and then use an alternative approach as a post-processing step. One approach for refinement is to use the logical relationships among constraints and orientations to identify possible errors. Bromberg and Margaritis [14] use the logic of argumentation to identify the results of single independence tests that are not consistent. This approach can address both false negative and false positive errors, but does not use likelihood to determine the possible value of those improvements, instead relying solely on logical consistency and the p-value of the test. Steck and Tresp [89] and Badea [8] use the logical inconsistencies among oriented edges to identify and resolve possible errors. Both of these approaches are conservative and remove orientations when there are conflicts.

An alternative approach to refinement is to use greedy search that is seeded from the result of the constraint-based algorithm. Abellan et al. [1] describe a refinement algorithm that also uses greedy search in DAG space as its final step. Spirtes and Meek [84] use a similar strategy for refinement but search over equivalence classes instead of DAGs. Neither of these algorithms utilize the learned constraints during the refinement and are not optimized for structural accuracy.

7.3 Greedy Relaxation Algorithm

7.3.1 Overview

Here I present a simple greedy algorithm for constraint relaxation, called RELAX (Algorithm 4). I show in Section 7.4 that this approach for relaxation leads to significant improvements in the structural accuracy of learned models. The algorithm starts by running constraint identification to learn constraints followed by edge orientation to produce the starting structure. After the first model has been identified, RELAX uses a local greedy search over possible relaxations of the constraints. Each time through the while loop, the algorithm chooses the single constraint which, if relaxed, leads to the largest improvement in the score. I use the BDeu score to score models [44], but any similar penalized likelihood score could be used in its place. If the best relaxation produces a model with a higher score than the current best model, then the relaxation is applied to the constraints and the algorithm continues searching for additional relaxations. If no relaxation leads to an improvement, the algorithm halts with the current structure.

To make this search possible, the constraints described in the previous chapter (see Section 6.3.1) are augmented by an explicit representation of the set of dependence constraints implied by the lack of independence constraints between a pair of variables. With these additional constraints, the absence or presence of each possible edge in the network directly corresponds to a single constraint. Due the direct correspondence between constraints and edges, the RELAX algorithm is, at its core, a search over undirected skeletons. To relax a constraint, I simply toggle the corresponding edge in the current network and update the sepset accordingly. If a constraint is toggled from dependence to independence, the sepset is set to the empty set; if it is toggled from independence to dependence, the sepset can be ignored since the edge now exists. I then orient the edges based on the current formulation of constraints.

RELAX combines aspects of constraint-based, hybrid, and refinement algorithms for learning the structure of Bayesian networks. RELAX uses constraints to orient the edges like a constraint-based algorithm, but also incorporates a penalized likelihood score into model selection like a hybrid algorithm. RELAX is also a refinement algorithm as it runs a constraint-based algorithm (EDGE-OPT) to completion before attempting to refine the results via relaxation. By combining the strengths of multiple approaches, the RELAX algorithm is able to achieve higher structural accuracy than any single one of these approaches.

Algorithm 4 Constraint Relaxation

```
procedure RELAX(Data)
    C = LEARN-CONSTRAINTS(Data)
    B = ORIENT-EDGES(C)
    S = BDEU(B, Data)
    B* = B, S* = S, C* = C
    updated = True
    while updated=True do
        updated = False
        for c ∈ C do
            // Relax the constraint
            C' = C\c
            B' =ORIENT-EDGES(C')
            S' =BDEU(B', Data)
            if S' > S then
                B = B', S = S', C = C'
            end if
        end for
        if S > S* then
            B* = B, S* = S, C* = C
            updated = True
        end if
    end while
    return B*
end procedure
```

7.3.2 Learn-Constraints Module

To emphasize that constraint relaxation is a general approach, the Learn-Constraints and Orient-Edges functions that appear within Algorithm 4 are place holders for any suitable algorithm. For the Learn-Constraints module, any constraint-based algorithm which produces independence constraints can be plugged into that spot. To learn the constraints, Relax uses the FAS algorithm. Other possibilities include MMPC [95, 96] and TPDA [18].

7.3.3 Orient-Edges Module

The requirements for the Orient-Edges module are more restrictive than the requirements for the Learn-Constraints module. To make Relax a constraint-based algorithm, the Orient-Edges module must orient the edges using the constraints. Since the Relax algorithm also uses BDeu (or other penalized likelihood score) to choose between models, the results of edge orientation must produce a DAG or other structure that can be scored using these metrics. Therefore, Relax must be used with the constraint optimization approach described in Chapter 6 because that algorithm orients edges using the constraints and guarantees that the structure is a DAG. PC-Edge is not appropriate because it frequently produces networks with bidirected edges or cycles. On data with discrete variables, those networks cannot be parameterized, preventing scoring the models with likelihood-based scores such as BDeu [87]. To my knowledge, no additional constraint-based edge orientation algorithms exist.

7.3.4 Computational Complexity

At its core, the Relax algorithm is a greedy search over skeletons; there is a single constraint corresponding to each possible edge location. At each step in the search process, the algorithm considers relaxing every constraint, therefore the branching factor of the search tree is $\mathcal{O}(n^2)$ since there are $\mathcal{O}(n^2)$ edge locations in the graph.

Furthermore, for every relaxation considered, the algorithm also runs EDGE-OPT; therefore, the overall complexity of RELAX is n^2 times the complexity of EDGE-OPT. This is a polynomial algorithm.

7.4 Experimental Evaluation

7.4.1 Experimental Set-up

We evaluated the RELAX algorithm on training data generated from the 4 real-world Bayesian networks (ALARM, INSURANCE, POWERPLANT, AND WATER) and 25 synthetic networks generated using the BNGenerator (SYNTHETIC). Each of these datasets is described in more detail in Appendix A. I also evaluated widely used algorithms from four different classes of structure learning algorithms: (1) the constraint-based PC algorithm [87], (2) a hybrid algorithm similar to MMHC (HYBRID) [96], (3) unconstrained greedy hill-climbing (GS), and (4) Greedy Equivalence Search (GES), which searches in the space of equivalence classes. The RELAX, PC, and HYBRID algorithms all use constraints learned using the FAS algorithm. GES, while not a constraint-based algorithm, has been proven correct in the sample limit. RELAX uses the following setting for the EDGE-OPT algorithm: $k = 0.5, numRestarts = 3$. I implemented each these algorithms as part of the POWERBAYES Java package with the exception of GES, where I used the implementation in the TETRAD IV package[1]. Note that the GES algorithm did not terminate successfully on WATER. The POWERBAYES Java package for structure learning of Bayesian networks is described in Appendix B.

On ALARM, INSURANCE, and WATER, I generated a series of five training sets at each of the following sample sizes $n = \{250, 500, 1000, 2000, 5000, 7500, 10000\}$. On POWERPLANT and SYNTHETIC, I generated five training sets at $n = \{500, 5000, 10000\}$.

[1]Available from `http://www.phil.cmu.edu/projects/tetrad/tetrad4.html`

Every algorithm considered was given identical training sets and used the same held-out test set for likelihood and BDeu calculations. Since the goal of this thesis is understanding and improving structural accuracy, the primary evaluation metrics considered here are compelled F-measure and structural Hamming distance (SHD). Compelled F-measure is an indicator of the causal interpretability of the learned model. It is combination of precision and recall of the compelled edges. Compelled edges are oriented the same direction in every member of the equivalence class of the learned model. SHD is an overall measure of structural accuracy. It measures the errors of both compelled and uncompelled edges. I also considered a variety of other metrics including the number of true positive edges, compelled precision and compelled recall individually, the number of skeleton errors, the number of orientation errors, and loglikelihood and BDeu score computed on a held-out dataset.

7.4.2 Evaluation of Structural Accuracy

On data generated from the real-world networks, the RELAX algorithm produces models with significantly improved compelled F-measure while maintaining comparable SHD to the comparison algorithms. The learning curves for these results are shown in Figure 7.1. Learning curves for the additional metrics can be viewed in Section 7.7. The RELAX algorithm performs better than or comparable to the best comparison algorithm on both compelled F-measure and SHD across all real-world datasets. At small sample sizes, GS produces models with better compelled F-measure but also has significantly worse structural Hamming distance when compared to the RELAX algorithm. To test whether these improvements were significant, I again used the randomized ANOVA test of Piater et al. [77]. This test is designed to determine the significance of the differences between two learning curves such as the ones shown in Figure 7.1. The p-values of the RELAX algorithm compared to the other algorithms are shown in Table 7.1. These p-values indicate the strength of the main effect as mea-

Figure 7.1: Evaluation of structural accuracy using compelled F-measure and structural Hamming distance.

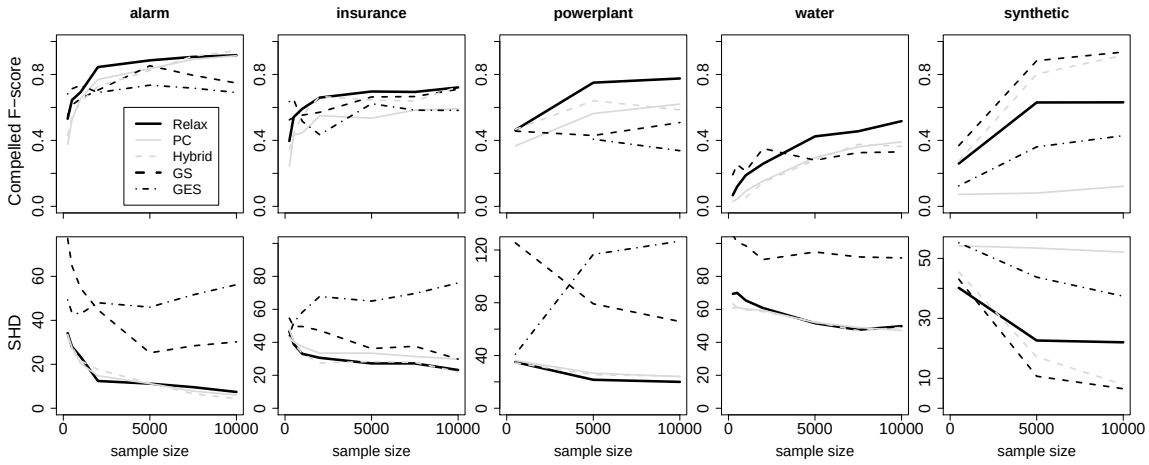


Table 7.1: P-values of the differences in compelled F-measure and structural Hamming distance (SHD) between RELAX and the baseline algorithms. *Italics* indicate that RELAX has worse performance.

	Network	PC	HYBRID	GS	GES	EDGE-OPT
F-measure	Alarm	0.006	0.006	0.027	0.01	0.035
	Insurance	0.01	0.135	0.617	0.052	0.007
	Powerplant	0.006	0.087	0.01	0.009	0.488
	Water	0.00	0.018	0.094	–	0.157
	Synthetic	0.00	*0.000*	*0.000*	0.00	0.00
SHD	Alarm	0.541	0.523	0.007	0.008	0.451
	Insurance	0.00	0.733	0.006	0.00	0.53
	Powerplant	0.006	0.021	0.00	0.00	0.472
	Water	0.00	0.00	0.00	–	0.00
	Synthetic	0.00	*0.122*	*0.02*	0.00	*0.009*

sured using 1000 randomization trials. The differences on the SYNTHETIC datasets are explained in more detail in Section 7.4.5.

A summary of results on the real data (shown in Table 7.2) shows that RELAX outperforms the comparison algorithms across all metrics with few exceptions. Comparing the results from RELAX and EDGE-OPT indicates that using constraint relaxation leads to consistent improvements over edge orientation alone. The one

Table 7.2: Proportion of runs on ALARM, INSURANCE, POWERPLANT, and WATER where RELAX exceeds the other algorithms on each metric.

Measure	PC	HYBRID	GS	GES	EDGE-OPT
TP	0.884	0.767	0.542	0.731	0.836
Prec.	0.813	0.508	0.842	0.881	0.662
Recall	0.800	0.675	0.367	0.627	0.822
F	0.849	0.683	0.617	0.701	0.773
SHD	0.440	0.375	0.958	0.985	0.364
Skel.	0.111	0.142	0.992	0.985	0.111
Orient	0.813	0.633	0.858	0.746	0.716
BDeu	–	0.867	0.017	0.896	1.000
LogLL	–	0.825	0.050	0.896	1.000

exception is skeleton errors. Since RELAX is permitted to search a larger space of models, it tends to incur a small number of additional skeleton errors. These errors, however, are more than compensated for by the corresponding decrease in orientation errors that relaxation permits.

7.4.3 Likelihood

In addition to producing models with the highest structural accuracy, the RELAX algorithm produces models with comparable likelihood to models produced using a standard hybrid algorithm on data with and without known structure. However, on both types of data, straight greedy search exceeds the performance of both the hybrid and relaxation approach. These results are shown in Figure 7.2 and Table 7.3. Note that I do not compare RELAX to the constraint-based PC algorithm as it does not consistently produce models that can be scored with these metrics. The hybrid approach of Tsamardinos et al. [96] uses a search-and-score approach for edge orientation in place of the usual constraint-based approach. This trade-off is made to improve likelihood at the expense of structural accuracy. However, as the results of the RELAX algorithm show, it is still possible to maintain (even improve) structural accuracy and achieve even higher likelihood than the standard hybrid approach. The

Table 7.3: Likelihood on datasets with unknown structure. **Bold** indicates a significant improvement over RELAX.

	COIL	Credit	Letters
RELAX	-26440.36	-13683.92	-166133.84
PC-HYBRID	-26432.79	-13398.88	**-159258.96**
GREEDY SEARCH	**-26347.85**	-13465.42	**-158395.92**

significant trade-off that the RELAX algorithm makes is the much longer runtime needed to achieve accurate structure. I consider runtime in the following section.

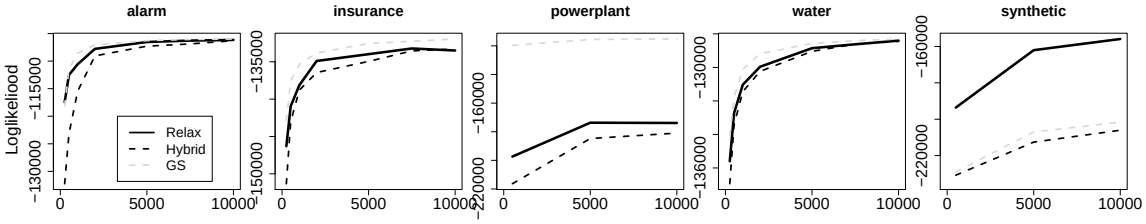


Figure 7.2: Loglikelihood on generated datasets.

7.4.4 Runtime

As with the EDGE-OPT algorithm described in Chapter 6, the runtimes of the RELAX algorithm (shown in Figure 7.3) are significantly longer than the runtimes of the comparison algorithms, though this is also an upper bound on the best possible runtime. For each possible relaxation, the EDGE-OPT algorithm is used for edge orientation on the updated skeleton. There are $\mathcal{O}(n^2)$ possible relaxations for each successor. As I noted before, EDGE-OPT requires many checks of minimal d-separators to determine how many constraints are satisfied and this is the largest component of runtime. It may be possible to cache the graphs and reduce the num-

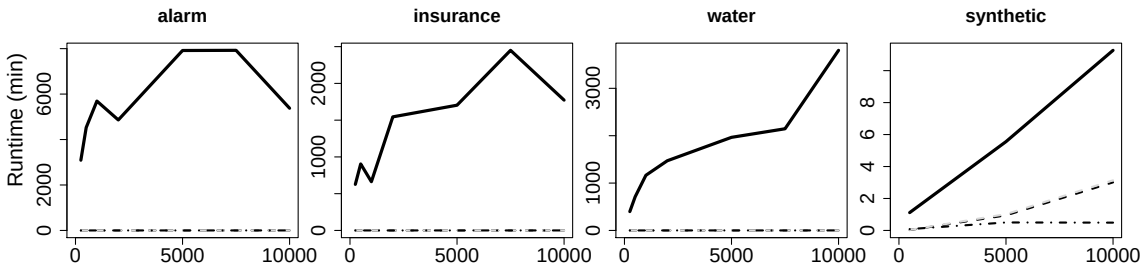


Figure 7.3: Runtime of the RELAX algorithm and the comparison algorithms.

ber of d-separation checks needed; however, that exploration is beyond the scope of this thesis.

7.4.5 Analysis of Synthetic Network Experiments

For both structural accuracy and likelihood, the results on the synthetically generated networks are markedly different from the results on data generated from real networks. The RELAX algorithm has better structural accuracy than PC and GES, but worse structural accuracy than greedy search and the hybrid algorithms. The algorithms that produce a final orientation by searching in DAG space have much higher structural accuracy than the other approaches. However, when comparing the of likelihood these approaches, the RELAX algorithm performs much better. An examination of the parameters of the true networks shows that distributions of the parameters vary significantly between the real and synthetic networks (Figure 7.4). The parameters on the real networks depend on the nature of the domain, whereas the BNGenerator software generates the parameters of the synthetic networks uniformly at random [48]. Since both distributions are reasonably plausible to encounter in practice, it is up to the practitioner to determine which learning algorithm is best suited to the task. However, it is informative to note that the ordering of the algorithms on likelihood when run on data without known structure is consistent with

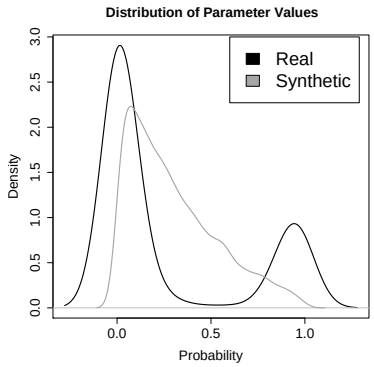


Figure 7.4: Distribution of the parameters of the true networks.

the results on data generated from the real networks. This provides evidence that the situations were RELAX performs better are more likely to be encountered in practice.

7.5 Related Work

One way of looking at the RELAX algorithm is as a simple hill-climbing search with an advanced search operator that applies a single addition or deletion of an edge in the skeleton, and that propagates that change to the orientations of all remaining edges in the skeleton. If viewed this way, the RELAX algorithm uses an approach similar to the approach of Steck [88]; however, it chooses an orientation at each step to maximize the number of constraints satisfied instead of a likelihood score. Because Steck does not consider any constraints when re-orienting edges, it is only a search-and-score algorithm. RELAX improves the structural accuracy of the approach by incorporating the constraints into edge orientation.

RELAX is a hybrid algorithm that uses a likelihood score to relax both independence and dependence constraints, allowing the algorithm to correct both false positive and false negative errors. This is in contrast to the MMHC algorithm, which can only exclude edges appearing in the skeleton and correct false postive errors, if adding that edge does not improve the score. However, MMHC can never correct a

false negative error, as edges not appearing in the skeleton can never be added to the final model. Both constraint relaxation and MMHC rely on a penalized likelihood score to determine which edges appear in the final model.

Existing constraint-based algorithms, such as PC [87], use deterministic rules for edge orientation. Constraint relaxation for learning Bayesian networks is inspired by but distinct from constraint relaxation in the partial constraint satisfaction literature, which emphasizes search in the space of constraints [35]. In this scenario, I am not only relaxing constraints to achieve a consistent solution but also removing constraints from the knowledge base that, upon relaxation, lead to increased likelihood of the global model. Additionally, if some of the constraints are incorrect, then constraint relaxation would be valuable even if there exists a global structure which is consistent with the current constraints.

Two recursive algorithms have recently been proposed to interleave conditional independence tests and edge orientation [99, 100]. These algorithms recursively partition the variables into subsets that satisfy the Markov property and independently determine separators and orientations for each of the subsets. While these algorithms combine independence tests with edge orientation, they differ from constraint relaxation in one key respect: there is no revision of initial statistical information based on a global structure or additional statistical information. Revising the independence constraints in the context of the fully oriented model is one of many opportunities taken advantage of by constraint relaxation.

7.6 Discussion

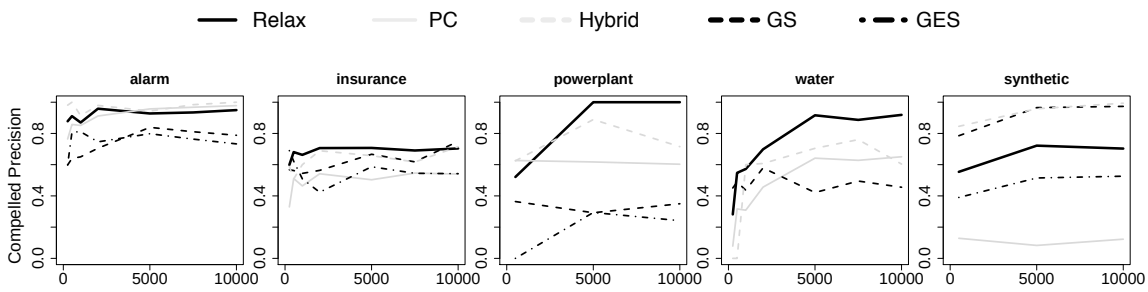
Constraint relaxation is a new approach for learning accurate structure of Bayesian networks. The Relax algorithm is an implementation of a constraint relaxation algorithm. The Relax algorithm combines the strengths of three different approaches for learning the structure of Bayesian networks: constraint-based, search-and-score,

and refinement algorithms. RELAX combines constraint optimization and penalized likelihood scoring for a hybrid approach to model selection. However, unlike other hybrid methods that use heuristic search to orient the edges appearing in the skeleton [2, 96], constraint relaxation searches over skeletons and uses the resulting constraints to orient the edges. This search over skeletons draws from the refinement paradigm as each relaxation selected improves the structure of the original model.

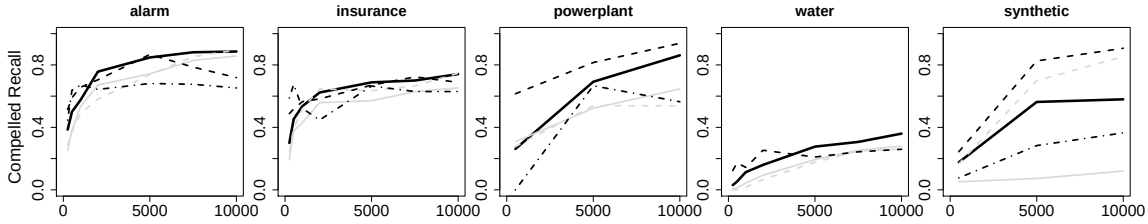
The RELAX algorithm also produces models with significantly higher structural accuracy across a wide range of metrics. In particular, using this algorithm results in improvements in compelled F-measure, which is a measure of model correctness and actionability of a Bayesian network, but without sacrificing performance on the other metrics such as structural Hamming distance, where constraint relaxation performs comparably to existing algorithms. In addition, constraint relaxation produces models with similar loglikelihood to hybrid search algorithms.

7.7 Additional Experimental Results

This section contains a collection of additional experimental results demonstrating the effectiveness of the RELAX algorithm.



(a) Compelled Precision: Percentage of learned compelled edges also appearing in the true model (higher is better).



(b) Compelled Recall: Percentage of true compelled edges also appearing in the learned model (higher is better).

Figure 7.5: Compelled precision and recall.

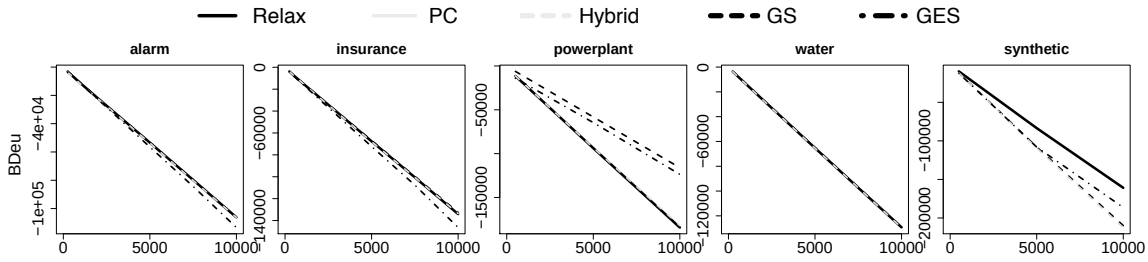
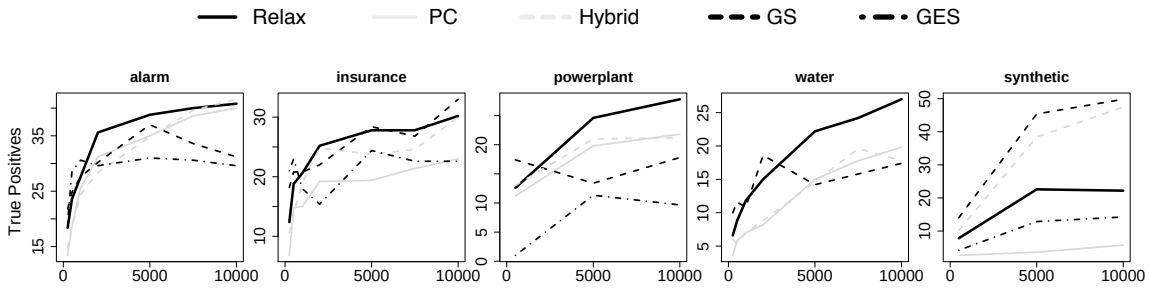
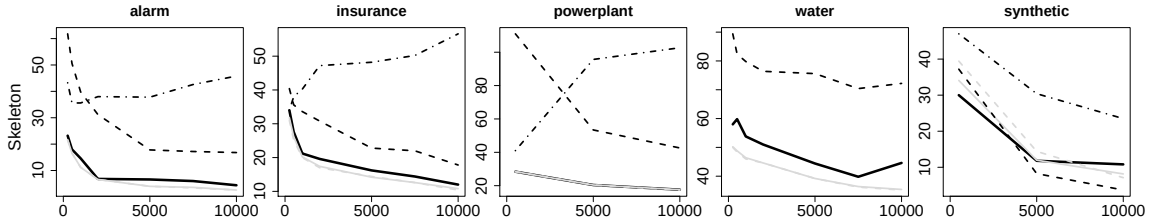


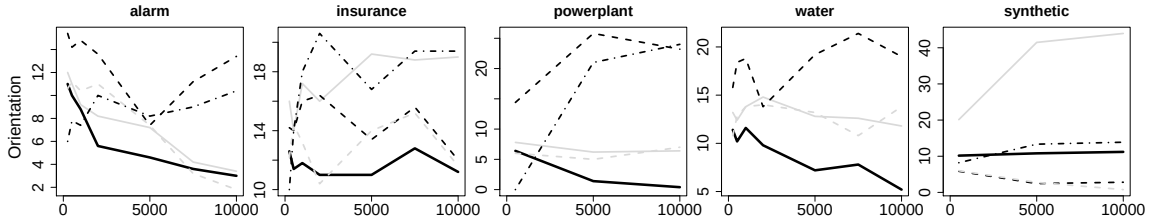
Figure 7.6: BDeu results of the learned model computed on a held-out test set of 10000 instances sampled from the true model (higher is better).



(a) True Positives: Number of edges oriented correctly (higher is better).



(b) Skeleton Errors: Incorrect addition or subtraction of edges in the learned model (lower is better).



(c) Orientation Errors: Edges where the orientation differs between the learned model and the true model (lower is better).

Figure 7.7: Additional edge metrics.

CHAPTER 8

SUMMARY AND CONCLUSIONS

8.1 Summary of Contributions

The learned structure of Bayesian networks can be used for guiding future action and understanding the causal mechanisms of a system if structure learning algorithms are able to learn accurate structure and certain assumptions are met. Constraint-based algorithms for learning the structure of Bayesian networks are designed to recover accurate structure and have strong asymptotic properties. However, given limited training data, these algorithms make errors and produce inaccurate structure.

The goal of this thesis is to provide understanding into the sources of errors and improve the structural accuracy of constraint-based algorithms for learning the structure of Bayesian networks. Structural accuracy measures the ability of an algorithm to recover the structure of the generating distribution, not just approximate the probability estimates of that model. Constraint-based algorithms work in two independent phases. First, constraint identification learns a set of independence constraints between variables. Second, edge orientation combines those constraints into a final model. This thesis makes the following contributions towards improving the structural accuracy of constraint-based algorithms:

- An empirical analysis of all types of errors in the Fast Adjacency Search (FAS) algorithm. This analysis showed that false negative errors are the largest source error and no existing solution can adequately solve the problem.

- The first correction, called Power, for low statistical power hypothesis tests that is able to address all four factors contributing to low statistical power.

Unlike other approaches for improving statistical power, the POWER correction incorporates the expected effect size into estimates of statistical power.

- An empirical analysis demonstrating a fundamental trade-off between false positive and false negative errors using different approaches for improving power that vary in both strategy and sophistication. These results indicate the need for algorithmic approaches, such as constraint satisfaction, for improving structural accuracy.

- A new statistical approach using propensity score matching to improve the power of hypothesis tests used in constraint identification. This approach combines orthogonal approaches—traditional conditional hypothesis tests and propensity score matching—into a single hypothesis test framework resulting in improved statistical power.

- Introduction and implementation of constraint satisfaction algorithms for learning the structure of Bayesian networks. Constraint satisfaction approaches are a non-statistical approach for mitigating the effects of errors in constraint-based algorithms. The two new constraint satisfaction algorithms are:

  - An edge orientation algorithm, called EDGE-OPT, that uses constraint optimization to consider constraints jointly and is guaranteed to produce a DAG structure. To my knowledge, EDGE-OPT is only the second sound constraint-based edge orientation algorithm appearing in the literature. The constraint optimization approach is extensible and can support different types of constraints, in addition to independence constraints.

  - A constraint relaxation algorithm, called RELAX, that combines constraint identification and edge orientation into a single search procedure to improve structural accuracy. The RELAX algorithm combines aspects of constraint-based, hybrid, and refinement algorithms to achieve learned

models with significantly improved structural accuracy compared to existing approaches.

There are two key principles underlying these contributions. First, combining statistical and algorithmic techniques with different strengths into a single algorithm is an effective strategy of improving the structural accuracy of constraint-based algorithms for learning the structure of Bayesian networks. Second, improvement of structural accuracy can be made by utilizing all available information. The first principle is demonstrated by the new propensity score matching approach and the RELAX algorithm as both combine statistical and algorithmic approaches, respectively, to produce more accurate structure. The second principle is demonstrated in the EDGE-OPT algorithm and the RELAX algorithm. EDGE-OPT considers all constraints jointly to improve the quality of learned structure, and the RELAX algorithm integrates constraint identification and edge orientation, allowing information from edge orientation to propagate back to constraint identification. Conceptually, these contributions also provide a platform for incorporating more and varied information into the learning process. Additional statistical tests could be incorporated along with the propensity score matching, and more advanced constraints could be considered alongside the independence constraints during edge orientation. These contributions are the next steps along what is hopefully a long and fruitful path towards improving the structural accuracy of algorithms for learning the structure of Bayesian networks.

8.2 Looking Beyond

In many ways, the constraint satisfaction approaches presented in this thesis only scratch the surface of the problem of automated causal inference. There are opportunities to (1) further utilize the interaction among constraints to improve the structural accuracy of the learned models, (2) explore the robustness of these algorithms un-

der different learning conditions, and (3) consider alternative metrics for evaluating causal models.

The constraint satisfaction algorithms presented in this thesis are only the first steps towards fully utilizing the interaction among the constraints to learn better causal models. In addition to enabling constraint-satisfaction algorithms, considering the constraints jointly also enables finer-grained reasoning about the results of individual hypothesis tests. This type of reasoning might provide an opportunity to more accurately identify incorrect conclusions resulting from individual low-power statistical tests and facilitate the creation of additional constraints on the structure.

This thesis examined the performance of structure learning algorithms under nearly ideal conditions; there are many practical learning tasks which do not have the same properties. One avenue for future exploration is exploring the robustness of different learning algorithms under less ideal conditions, such as when the data are not generated from a DAG, contain missing or unobservable variables, or are generated using a noisy sampling process. If deviations from the standard assumptions can be encoded via constraints, then these new constraints can be easily incorporated into learning.

Structural accuracy is the most widely used metric for evaluating causal models; however, structural accuracy does not differentiate between causal claims of different strengths. In addition to identifying which causal inferences are correct, it would be valuable for a learning algorithm to provide confidences about the accuracy or importance of each link. One possible method for providing confidences compares the results of interventions on the learned and true models. If an intervention using the learned model correctly predicts a large effect in the world, then that pathway should be given additional weight in the model. This would provide additional focus for choosing among edges in the model.

Each of these challenges can be addressed by constraint satisfaction, provided constraints that capture each of these phenomenon can be created. Constraint satisfaction for learning the structure of Bayesian networks is flexible and extensible, combining different constraints and different statistical approaches into a single paradigm for structure learning that provides a path to improved structural accuracy and interpretability of causal models.

APPENDIX A

DATA SOURCES

This appendix details the test Bayesian networks used throughout the thesis and provides pointers to sources for other Bayesian networks.

A.1 Bayesian Networks used in Analysis

The networks considered in different parts of this thesis are listed in Table A.1. The table includes the name, number of variables and edges, and other details about the structure. Recall that compelled edges are edges that have the same orientation in every member of an equivalence class of Bayesian networks [20]. Each of these networks has been obtained from a real modeling or decision problem and has been validated by experts. Citations for each network are included below. The WIN95PTS network was developed at Microsoft Research and contributed to the community by Jack Breese.

A.2 Bayesian Network Repositories

All of the networks shown in Table A.1 were obtained from an online Bayesian network repository. Four repositories are listed here:

1. Bayesian Network Repository (BNR)
    - `http://compbio.cs.huji.ac.il/Repository/networks.html`
2. Bayesian Network and Decision Graph (BNDG)

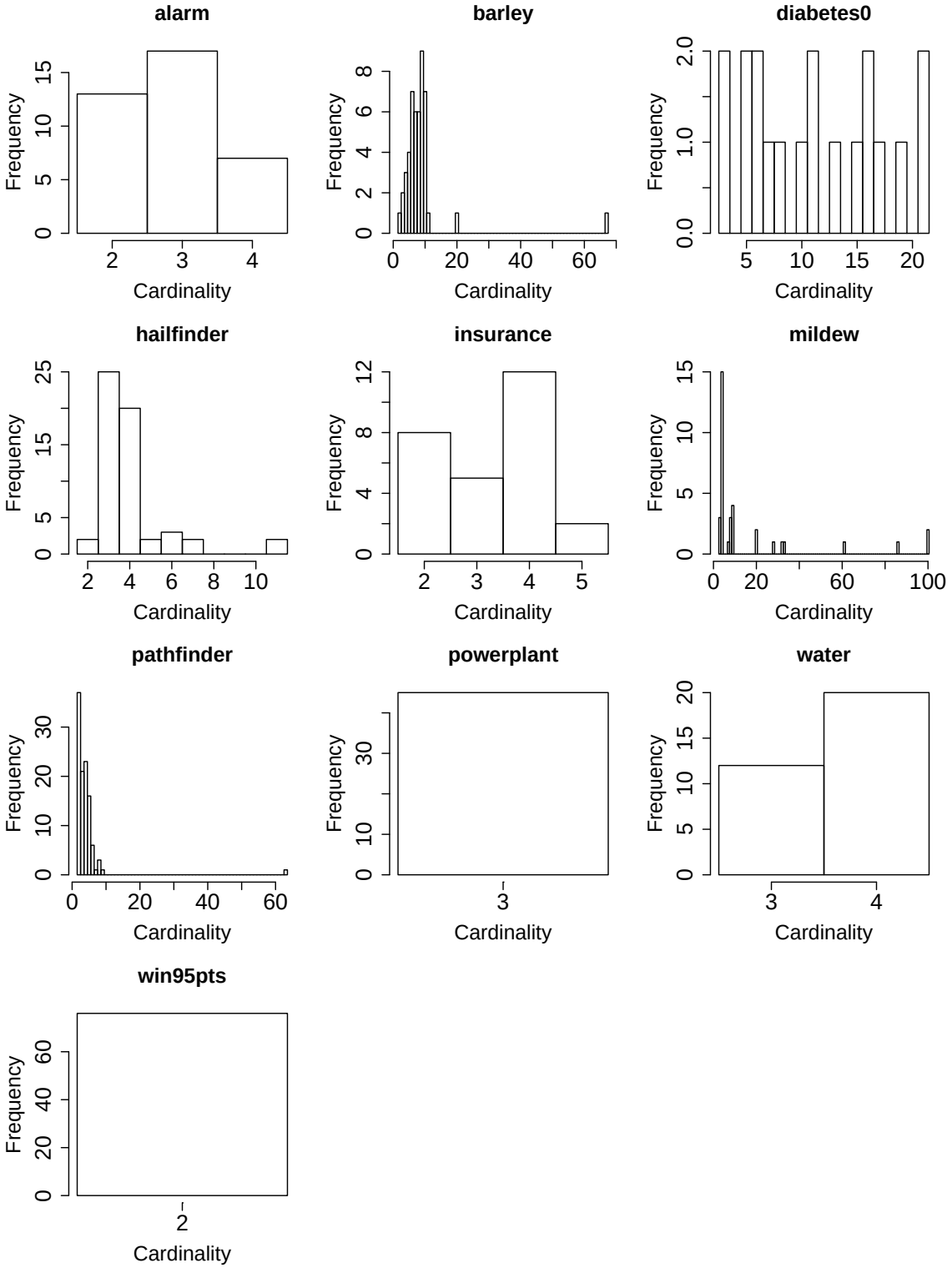


Figure A.1: Distributions of the cardinality of variables for the ten networks considered in this thesis.

Table A.1: Summary statistics of Bayesian networks used in this thesis.

Data Name	Num. Vars.	Num. Edges	Compelled Edges	Avg. Degree	Avg. Cardinality	Source	Reference
Alarm	37	46	42	1.24	2.84	BNR	[9]
Barley	48	84	75	1.75	8.77	BNR	[54]
Hailfinder	56	66	49	1.18	3.98	BNR	[29]
Insurance	27	52	34	1.93	3.30	BNR	[11]
Pathfinder	109	195	73	1.79	4.11	BNR	[43]
Mildew	35	46	46	1.31	17.6	BNR	[49]
Water	32	66	60	2.06	3.63	BNR	[51]
Win95pts	76	112	100	1.47	2.0	BNR	
Diabetes0	19	23	20	1.21	11.21	BNR	[7]
Powerplant	45	42	13	0.93	3.0	BNDG	[71]

- `http://bndg.cs.aau.dk/html/bayesian_networks.html`

3. Decision Systems Laboratory at the University of Pittsburgh

   - `http://genie.sis.pitt.edu/networks.html`

4. Netica

   - `http://www.norsys.com/netlibrary/index.htm`

5. Bayesian AI

   - `http://www.csse.monash.edu.au/bai/book/networks.html`

A.3 BNGenerator

Since the datasets from real domains have a limited range of characteristics, synthetic generation of Bayesian networks can be helpful in exploring the space of possible problems. The BNGenerator software package can be used to generate Bayesian network structures uniformly at random [48]. The package allows the specification of the number of variables, ranges on the edges, and distribution of degrees of the structures.

The parameter settings used to generate the networks are listed in the section with the experimental evaluation.

APPENDIX B

POWERBAYES **SOFTWARE**

The PowerBayes open-source software package was developed by the author as the experimental platform underlying the empirical evaluation of this thesis. The package was developed to be modular to ease the mixing and matching of different components for structure learning of Bayesian networks. Other packages typically implement monolithic algorithms which makes it difficult to pick and choose which components are used. The modular approach also lends itself to providing unit tests for the code, and many of the components have unit tests included to ensure the correctness and consistency of the results.

PowerBayes is written in Java and relies on many other open source libraries including WEKA, Colt, JUnit, Log4J and others. It is available for download at: `http://kdl.cs.umass.edu/powerbayes/`. The package contains implementations of many popular structure learning algorithms including those introduced in this thesis. An incomplete list includes:

- PC (broken down into FAS and PC-Edge)
- MMHC
- Greedy Search
- Edge-Opt
- Relax
- Power Correction

BIBLIOGRAPHY

[1] J. Abellan, M. Gomez-Olmedo, and S. Moral. Some variations on the PC algorithm. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models*, 2006.

[2] S. Acid and L.M. de Campos. A hybrid methodology for learning belief networks: BENEDICT. *International Journal of Approximate Reasoning*, 27(3): 235–262, 2001.

[3] S. Acid and L.M. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

[4] Alan Agresti. *Categorical Data Analysis*. John Wiley & Sons, 1990.

[5] H. Akaike. A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, 30(9):14, 1978.

[6] Steen A. Andersson, David Madigan, and Michael D. Perlman. On the Markov equivalence of chain graphs, undirected graphs and acyclic digraphs. Technical Report 281, Department of Statistics, University of Washington, 1994.

[7] Steen Andreassen, Roman Hovorka, Jonanthan Benn, Kristian G. Olesen, and Ewart R. Carson. A model-based approach to insulin adjustment. In *Proceedings of the Third Conference on Artificial Intelligence in Medicine*, 1991.

[8] L. Badea. Determining the direction of causal influence in large probabilistic networks: A constraint-based approach. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 263–267, 2004.

[9] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on AI and Medicine*. Springer-Verlag, Berlin, 1989.

[10] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[11] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29, 1997.

[12] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[13] Facundo Bromberg and Dimitris Margaritis. Efficient and robust indpendence-based Markov network structure discovery. In *Proc. of the 16th International Joint Conference on Artificial Intelligence*, 2007.

[14] Facundo Bromberg and Dimitris Margaritis. Improving the reliability of causal discovery from small data sets using argumentation. *Journal of Machine Learning Research*, 10:301–340, February 2009.

[15] Wray Buntine. Theory refinement on Bayesian networks. In *Proceedings on Uncertainty in Artificial Intelligence (UAI)*, pages 52–60, 1991.

[16] Wray Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, April 1996.

[17] Stéphane Champely. *The Pwr Package for R.* UCB Lyon 1, France, February 2007. URL `http://cran.r-project.org/doc/packages/pwr.pdf`.

[18] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137 (1-2):43–90, 2002.

[19] David Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, February 2002.

[20] David Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

[21] David Maxwell Chickering. A transformational characterization of equivalent Bayesian network structures. In *UAI*, pages 87–98, 1995. URL `citeseer.ist.psu.edu/chickering95transformational.html`.

[22] D.M. Chickering. Learning Bayesian networks is NP-complete. *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130, 1996.

[23] W.G. Cochran. Some methods for strengthening the common $\chi 2$ tests. *Biometrics*, pages 417–451, 1954.

[24] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences.* Lawrence Erlbaum Associates, Inc., 2nd edition, 1988.

[25] Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[26] W. B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice.* Addison-Wesley, 2009.

[27] D. Dash and M.J. Druzdzel. Robust independence testing for constraint-based learning of causal structure. In *Proc. of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 167–174, 2003.

[28] Rina Dechter. *Constraint Processing*. Morgan Kauffman, 2003.

[29] W. Edwards. Hailfinder: Tools for and experiences with Bayesian normative modeling. *American Psychologist*, 53:416–428, 1998.

[30] Andrew Fast and David Jensen. Constraint relaxation for learning the structure of Bayesian networks. Tech Report 09-18, University of Massachusetts Amherst, Computer Science Department, 2009.

[31] Andrew Fast, Michael Hay, and David Jensen. Statistical power analysis for improved learning Bayesian network structure. In *Submitted to Proceedings of the 25th International Conference on Machine Learning*, 2008.

[32] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge discovery and data mining*, pages 1–34. American Association for Artificial Intelligence Menlo Park, CA, USA, 1996.

[33] Stephan E. Fienberg. *The Analysis of Cross-Classified Categorical Data*. MIT Press, 2nd edition, 1980.

[34] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, 13(3):57–70, 1992.

[35] E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58(1-3):21–70, 1992.

[36] Nir Friedman, Iftach Nachman, and Dana Peer. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999. URL `citeseer.ist.psu.edu/article/friedman99learning.html`.

[37] Dan Geiger, Thomas Verma, and Judea Pearl. D-separation: From theorems to algorithms. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, 1990.

[38] C. Genovese and L. Wasserman. Operating characteristics and extensions of the false discovery rate procedure. *Journal Of The Royal Statistical Society Series B*, 64(3):499–517, 2002.

[39] A. Goldenberg and A. Moore. Tractable learning of large Bayes net structures from sparse data. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[40] Russell Greiner, Adam J. Grove, and Dale Schuurmans. Learning Bayesian networks that perform well. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 1997.

[41] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79–119, 1997.

[42] D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *The Journal of Machine Learning Research*, 1:49–75, 2001.

[43] D. E. Heckerman, E. J. Horvitz, and B. N. Nathwani. Toward normative expert systems. part I: The Pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.

[44] David Heckerman, Dan Geiger, and David Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[45] Avi Herscovici and Oliver Brock. Improving high-dimensional Bayesian network structure learning by exploiting search space information. Technical Report 06-49, Department of Computer Science, University of Massachusetts Amherst, 2006.

[46] Geoff Hulten, David Chickering, and David Heckerman. Learning Bayesian networks from dependency networks: A preliminary study. In *Proceedings of AI & Statistics*, 2003.

[47] M. Hutter. Distribution of mutual information. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[48] J.S. Ide and F.G. Cozman. Random generation of Bayesian networks. In *Proceedings of the 16th Brazilian Sympoisum on Artificial Intelligence*, pages 366–375. Springer, 2002.

[49] A. Jensen and F. Jensen. MIDAS—An influence diagram for management of mildew in winter wheat. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 349–356, 1996.

[50] David Jensen and Paul Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38:309–338, 2000.

[51] Finn V. Jensen, Uffe Kjærulff, Kristian G. Olesen, and Jan Pedersen. Et forprojekt til et ekspertsystem for drift af spildevandsrensning (an expert system for control of waste water treatment — a pilot project). Technical report, Judex Datasystemer A/S, Aalborg, Denmark, 1989. In Danish.

[52] M.M. Joffe and P.R. Rosenbaum. Invited commentary: Propensity scores, 1999.

[53] Kenneth J. Koehler. Goodness-of-fit tests for log-linear models in sparse contingency tables. *Journal of the American Statistical Association*, 81(394):483–493, June 1986.

[54] K. Kristensen and I.A. Rasmussen. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33:197–217, 2002.

[55] HW Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 52(1), 2005.

[56] J.R. Landis, E.R. Heyman, and G.G. Koch. Average partial association in three-way contingency tables: A review and discussion of alternative tests. *International Statistical Review/Revue Internationale de Statistique*, pages 237–254, 1978.

[57] Junning Li and Z. Jane Wang. Controlling the false discovery rate of the association/causality structure learned with the PC algorithm. *Journal of Machine Learning Research*, 10:475–514, February 2009.

[58] Jennifer Listgarten and David Heckerman. Determining the number of non-spurious arcs in a learned DAG model: Investigation of a Bayesian and a frequentist approach. In *Proceedings of 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.

[59] B. Lu, E. Zanutto, R. Hornik, and P.R. Rosenbaum. Matching with doses in an observational study of a media campaign against drug abuse. *Journal of the American Statistical Association*, 96(456):1245–1253, 2001.

[60] Subramani Mani and Gregory F. Cooper. Causal discovery algorithms based on Y structures. In *NIPS 2006 Workshop on Causality and Feature Selection*, 2006.

[61] N. Mantel. Chi-square tests with one degree of freedom: Extensions of the Mantel-Haenszel procedure. *Journal of the American Statistical Association*, 58(303):690–700, 1963.

[62] N. Mantel and W. Haenszel. Statistical aspects of the analysis of data from retrospective studies of disease. *The Challenge of Epidemiology: Issues and Selected Readings*, 1(1):533–553, 2004.

[63] AE Maxwell. Comparing the classification of subjects by two independent judges. *The British Journal of Psychiatry*, 116(535):651–655, 1970.

[64] Q. McNemar. *Psychological statistics*. Wiley New York, 1969.

[65] C. Meek. Causal inference and causal explanation with background knowledge. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 403–410, 1995.

[66] Glenn W. Milligan. A computer program for calculating power of the chi-square test. *Educational and Psychological Measurement*, 39(3):681–684, 1979.

[67] Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning*, pages 552–559, August 2003.

[68] Frederick Mosteller. A k-sample slippage test for an extreme population. *The Annals of Mathematical Statistics*, 19(1):58–65, March 1948.

[69] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, pages 32–38, 1957.

[70] J.D. Nielsen, T. Kocka, and J.M. Pena. On local optima in learning Bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 435–442, 2003.

[71] T.D. Nielsen and F.V. Jensen. On-line alert systems for production plants: A conflict based approach. *International Journal of Approximate Reasoning*, 45 (2):255–270, 2007.

[72] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[73] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, 1982.

[74] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffman, 1988.

[75] Judea Pearl. *Causality*. Cambridge University Press, 2000.

[76] Judea Pearl and T.S. Verma. A theory of inferred causation. In *Proceedings of the 2nd International Conference on Knoledge Representation and Reasoning*, 1991.

[77] Justus H. Piater, Paul R. Cohen, Xiaoin Zhang, and Michael Atighetchi. A randomized ANOVA procedure for comparing performance curves. In *Proceedings of the 15th International Conference on Machine Learning*, pages 430–438, 1998.

[78] Thomas Richardson and Peter Spirtes. Ancestral graph Markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

[79] P.R. Rosenbaum and D.B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[80] Richard Scheines. An introduction to causal inference. In *Causality in Crisis?* Univ. of Notre Dame Press, 1997.

[81] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6 (2):461–464, 1978.

[82] William R. Shadish, Thomas D. Cook, and Donald T. Campbell. *Experimental and Quasi-experimental Designs for Generalized Causal Inference*. Houghton Mifflin Company, 2002.

[83] M. Singh and M. Valtorta. Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12(2):111–131, 1995.

[84] P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceedings of the first international conference on knowledge discovery and data mining*, pages 294–299. Menlo Park, CA: AAAI, 1995.

[85] Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. Report CMU-PHIL-15, Carnegie Mellon University, August 1990.

[86] Peter Spirtes, Christopher Meek, and Thomas Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In Clark Glymour and Gregory F. Cooper, editors, *Computation, Causation and Discovery*, pages 211–252. MIT Press Cambridge, MA, USA, 1999.

[87] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search*. MIT Press, 2nd edition, 2000.

[88] H. Steck. On the use of skeletons when learning in Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 558–565, 2000.

[89] H. Steck and V. Tresp. Bayesian belief networks for data mining. *Proceedings of the 2nd Workshop on Data Mining und Data Warehousing als Grundlage Moderner Entscheidungsunterstutzender Systeme*, pages 145–154, 1996.

[90] A. Stuart. A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika*, 42(3-4):412–416, 1955.

[91] M. Studeny. An algebraic approach to structural learning Bayesian networks. In *Proceedings of the 11th International Conference IPMU*, pages 2284–2291, 2006.

[92] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. *Proc. of the Twenty-first Conference on Uncertainty in AI*, pages 584–590, 2005.

[93] Jin Tian, Azaria Paz, and Judea Pearl. Finding minimal d-separators. Technical Report R-254, UCLA Computer Science Department, February 1998.

[94] Ioannis Tsamardinos and Laura E. Brown. Bounding the false discovery rate in local Bayesian network learning. In *Proceedings of the 23rd AAAI Conference on Artifical Intelligence*, 2008.

[95] Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct casual relations. In *Proceeding of ACM SIGKDD 2003*, 2003.

[96] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, March 2006.

[97] S. van Dijk, L.C. van der Gaag, and D. Thierens. A skeleton-based approach to learning Bayesian networks from data. In *Proc. of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 2838, 2003.

[98] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

[99] Xianchao Xie and Zhi Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9:459–483, March 2008.

[100] Raanan Yehezkel and Boaz Lerner. Bayesian network structure learning by recursive autonomy identification. *Journal of Machine Learning Research*, 10: 1527–1570, 2009.