2015

Informed Search for Learning Causal Structure

Brian J. Taylor
University of Massachusetts - Amherst

INFORMED SEARCH FOR LEARNING CAUSAL STRUCTURE

A Dissertation Presented

by

BRIAN J. TAYLOR

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

DOCTOR OF PHILOSOPHY

September 2015

College of Information and Computer Sciences

INFORMED SEARCH FOR LEARNING CAUSAL STRUCTURE

A Dissertation Presented

by

BRIAN J. TAYLOR

To my three sons. May this work inspire them to pursue knowledge, but more importantly, to persevere.

ACKNOWLEDGMENTS

I would like to thank my advisor, David Jensen, for his help in leading me through this long and arduous process. I am in debt to him for his insights, his wisdom, and his invaluable editing advice. He helped me foster a speck of an idea into a full body of work.

I also want to express appreciation to Dan Corkill for being a sounding board and his wealth of ideas and comments. He probably considers himself more of an outside observer than a guide, but he has help shaped this research more than he might imagine. To that I am grateful.

I thank the members of my committee for all of their advice, critiques, and patience. Mark Corner influenced several early aspects of this work and fostered my interests in social collaborative systems. Shlomo Zilberstein provided excellent input and feedback on Markov-decision processes and also shaped the direction of my work. Christopher Meacham offered interesting insights from an outside perspective and it was always a delight to discuss my work with him.

I also thank the many members of the Knowledge Discovery Laboratory for their asisstance throughout the years. This includes Matt Cornell, Agustin Schapira, Cindy Loiselle, Deb Bergeron, Marc Maier, Hüseyin Oktay, Matthew Rattigan, Andy Fast, Lisa Friedland, Michael Hay, Jen Neville, David Arbour, Katerina Marazopoulou, Amanda Gentzel, and Phil Kirlin.

Finally and perhaps most importantly, I would also like to thank my wife who endured just as long as I did on my journey. To help me achieve this she eagerly gave up her old life so I could begin a new life and threw her entire support behind me. She provided encouragement when it was needed, criticism when it was warranted, and cheers I desperately needed at the end. In some ways, this dissertation is as much hers as it is mine. I may only be able to repay her dedication to my cause by someday supporting her as she pursues her own doctorate.

ABSTRACT

INFORMED SEARCH FOR LEARNING CAUSAL STRUCTURE

SEPTEMBER 2015

BRIAN J. TAYLOR

B.Sc., WEST VIRGINIA UNIVERSITY

M.Sc., WEST VIRGINIA UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor David Jensen

Over the past twenty-five years, a large number of algorithms have been developed to learn the structure of causal graphical models. Many of these algorithms learn causal structures by analyzing the implications of observed conditional independence among variables that describe characteristics of the domain being analyzed. They do so by applying inference rules, data analysis operations such as the conditional independence tests, each of which can eliminate large parts of the space of possible causal structures. Results show that the sequence of inference rules used by PC, a widely applied algorithm for constraint-based learning of causal models, is effective but not optimal. This is because algorithms such as PC ignore the probability of the outcomes of these inference rules. We demonstrate how an alternative algorithm can reliably outperform PC by taking into account the probability of inference rule outcomes. Specifically we show that an informed search that bases the order of causal inference on a prior probability distribution over the space of causal constraints can generate a flexible sequence of analysis that efficiently identifies the same results as PC. This class of algorithms is able to outperform PC even under uniform or erroneous priors.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **xiii**

LIST OF FIGURES . . . . . **xv**

CHAPTER

1. INTRODUCTION **1**

1.1 Examples of Inference Rule Sequences 2
1.2 Central Ideas of the Dissertation 5

1.2.1 Inference Rules 5
1.2.2 Causal Learning as Search 5
1.2.3 The Benefits of Informed Search 6

1.3 Research Questions 8
1.4 Contributions 9

2. BACKGROUND **11**

2.1 Examples of Inference Rules to Find Causal Models 11

2.1.1 Photobase: An Example of Causal Discovery Using Experimentation 12
2.1.2 Stack Overflow: An Example of Causal Discovery Using Observational Data 13

2.2 Bayesian Networks and d-separation 14
2.3 Causal Interpretation of DAGs 16

2.3.1 Causal Markov Condition 17
2.3.2 Faithfulness 18
2.3.3 Causal Sufficiency 18

2.4 CPDAGs and Markov-Equivalence Classes 18
2.5 Causal Models 20
2.6 Constraints on the Model Space 21

2.6.1 Conditional Independence Tests 22
2.6.2 Edge-Orientation Rules 22

2.7 Constraint-Based Causal Learning 24
2.8 Inference Rules and Causal Inference 26
2.9 Markov Decision Processes 27
2.10 Related Work 29

3. INFERENCE RULES 30

3.1 Inference Rule Representation 31
3.2 Inference Rule Examples 33

3.2.1 Observational Inference Rules 33

3.2.1.1 The Conditional Independence Rule (CIRule) 33
3.2.1.2 The Conditional Dependence Rule (CDRule) 35
3.2.1.3 Edge-Orientation Rules 36

3.2.2 Interventional Inference Rules 39
3.2.3 Representing Inference Rules in General 41

3.3 Simplifying Assumptions 43

4. CAUSAL LEARNING AS A MARKOV DECISION PROCESS 44

4.1 Causal Search and Finding Inference Rule Sequences with Causal Plans 49
4.2 Policies to Solve the Causal MDP 53
4.3 An Architecture to Execute Causal Search 55

4.3.1 GeneratePlan 55
4.3.2 FindNextInferenceRule 56
4.3.3 ChooseOneInferenceRule 58
4.3.4 RuleIsValid 59
4.3.5 ExecutePlan 60

5. UNINFORMED POLICIES FOR SOLVING THE CAUSAL MDP 62

5.1 The PC Policy 62
5.2 Examples of PC Causal Plans 67
5.3 Other Examples of Uninformed Policies 72

5.3.1 The Random Policy 72
5.3.2 The SGS Policy 73
5.3.3 Pearl's Independence Graph Policy 74

6. OPTIMAL CAUSAL SEARCH AND OPTIMALITY OF THE PC POLICY 75

6.1 Causal Model Oracle 75
6.2 The Optimal Policy 76

6.2.1 Optimal Causal Search for Short Inference Rule Sequences 77
6.2.2 Solving Optimal Causal Search for Short Inference Rule Sequences 82

6.3 Evaluating the PC Policy 89
6.4 Ignorable Inference Rules 92
6.5 New Policy: Augmented PC 96
6.6 Further Improvements to the PC Policy 99

7. INFORMED SEARCH OF CAUSAL STRUCTURE 100

7.1 Policies for an Informed Search 100
7.2 The Posterior Policy 101
7.3 An Informative Prior over the Model Space 103

7.3.1 Implementation of an Approximate PMF 105

7.3.1.1 Rejection Sampling 108
7.3.1.2 The Prior Index 109

7.3.2 Updating the Prior into a Posterior 111

7.4 Details of The Posterior Policy 112

8. EVALUATION OF THE POSTERIOR POLICY 114

8.1 Comparing Policies on Expected Cost 114
8.2 Comparing Search Policies on Correctness 116
8.3 Performance of the Posterior Policy 117

8.3.1 Performance with an Uninformative Prior 117

8.3.1.1 Performance Evaluation 117
8.3.1.2 Example of the Posterior Inference Rule Sequence Under an Uninformative Prior 125
8.3.1.3 Computational Trade-Offs 130

8.3.2 Performance with an Informative Prior 131

8.3.2.1 Specification of the Informative Prior 131
8.3.2.2 Performance Evaluation 133
8.3.2.3 Example of the Posterior Inference Rule Sequence Under an Informative Prior 141
8.3.2.4 Computational Trade-Offs 144

8.3.3 Performance with a Misinformative Prior 146

8.3.3.1 Specification of the Misinformative Prior 146
8.3.3.2 Performance Evaluation 147
8.3.3.3 Example of the Posterior Inference Rule Sequence Under an Misinformative Prior 154
8.3.3.4 Computational Trade-Offs 157

8.3.4 Performance as a Function of the Size of M 159

9. CONCLUSION 163

9.1 Summary of Contributions 163
9.2 Implications 164
9.3 Areas of Future Work 164

9.3.1 Faster Search By Sampling Inference Rules 165
9.3.2 Additional Inference Rules 165
9.3.3 Causal Search with Probabilistic Constraints 166
9.3.4 Development of New Policies 167

BIBLIOGRAPHY 168

LIST OF TABLES

Table **Page**

6.1 The Number of Models True for Each Inference Rule. 83

6.2 The Number of Models True After Applying $CIRule(b, [], z)$. 84

6.3 The Number of Models True After Applying $CIRule(b, [x], y)$. 84

6.4 A Comparison of Optimal MDP Solution Approaches. 87

6.5 Sampled Model and *CIRule* Counts Across Evaluated Domains. 90

6.6 Expected Costs from Using PC, Augmented PC, and Optimal Policies. 99

8.1 The Expected Costs for the PC, Posterior, and Optimal Policies on an Uninformative Prior. 119

8.2 Probabilities of Models in a 3-variable Domain With an Uninformative Prior After Posterior Selects Inference Rules. 127

8.3 The Ranking of Available Inference Rules by the Posterior Policy With an Uninformative Prior. 128

8.4 Expected Times in Seconds to Identify an Inference Rule Sequence Across 3- to 10-variables With an Uninformative Prior. 130

8.5 The Informative Prior Used for Evaluation. 133

8.6 Probabilities of Models in a 3-variable Domain With an Informative Prior After Posterior Selects Inference Rules. 142

8.7 The Ranking of Available Inference Rules by the Posterior Policy With an Informative Prior. 143

8.8 Expected Times in Seconds to Identify an Inference Rule Sequence Across 3- to 5-variables With an Informative Prior. 145

8.9 The Misinformative Prior Used for Discussion. 154

8.10 Probabilities of Models in a 3-variable Domain With a Misinformative Prior. 155

8.11 The Ranking of Available Inference Rules by the Posterior Policy With a Misinformative Prior. 155

8.12 Probabilities of Models in a 3-variable Domain With a Misinformative Prior After Posterior Selects Inference Rules. 156

8.13 Expected Times in Seconds to Identify an Inference Rule Sequence Across 3- to 5-variables With a Misinformative Prior. 158

8.14 Expected Costs for the Posterior Across Different Sizes of M. 160

LIST OF FIGURES

Figure **Page**

1.1 Abstract Examples of Choosing Actions Within the Set of Potential Conditional Independence Tests. 3

1.2 Expected Number of Inference Rules of the PC Policy vs. the Posterior Policy. 5

2.1 A Bayesian Network for the Photobase Domain. 15

2.2 Four Causal Structures That Give Rise to a Statistical Association Between X and Y. In (d) X and Y Have Been Conditioned on W. 16

2.3 CPDAG example of the Photobase Causal Model. 19

2.4 The Markov-Equivalence Class for the Photobase CPDAG. 20

2.5 The Collider Detection Rule is Used to Orient *Coverage* and *PhotosCount* as Parents of the *Leaderboard*. 23

2.6 The Known Non-Collider Rule Relies on the *Chain* of Variables in a Path to Orient *PhotosCount* to the *Leaderboard*. 23

2.7 The Cycle Avoidance Rule Preserves Acyclicity by Orienting *CollaborationType* to *Leaderboard*. 24

3.1 Formalized Representation of an Inference Rule. 32

3.2 The CIRule Formal Specification. 34

3.3 The CDRule Formal Specification. 35

3.4 The Collider Detection Rule Formal Specification. 37

3.5 The Known Non-Collider Rule Formal Specification. 38

3.6 The Cycle Avoidance Rule Formal Specification. 39

3.7 The Interventional Inference Rule Formal Specification. 40

3.8 Graphical Structure Representing a Verma Constraint. 42

4.1 The State Space for a Simple 3-variable Domain. 51

4.2 The Four Possible Edge States in the Graphical Representation of the State of Knowledge. 51

4.3 The Causal Plan for the Example 3-variable Domain. 52

5.1 An Example of the PC Policy Computing Transition Probabilities of Available Actions. 65

5.2 How the PC Policy Should Compute the Transition Probabilities of Available Actions. 66

5.3 PC Causal Plan for 3-var Domain Example 68

5.4 PC Causal Plan for 3-var Domain Example, Part 2 of 3. 69

5.5 PC Causal Plan for 3-var Domain Example, Part 3 of 3. 70

5.6 PC Causal Plan for 3-var Domain With a Specific Goal. 71

6.1 The Optimal Causal Plan for a 3-var Example. 78

6.2 An Example of the Optimal Policy Computing Transition Probabilities of Available Actions. 79

6.3 A Causal Model Example From a 5-var Domain. 80

6.4 The Markov-Equivalence Class Containing the Test Causal Model. 81

6.5 An Example of Alternative Models That Entail the Conditional Independencies Matching the Test Causal Model. 82

6.6 Performance of the Different Optimal MDP Solutions. 88

6.7 Comparison of the PC Policy With the Optimal Policy. 91

6.8 X and Y are d-separated Given $\mathbf{Z}$. 92

6.9 Five Cases Where $CIRule(X, [Y], Z)$ is a Redundant Inference Rule. 94

6.10 How PC Searches for Neighbors Between X and Y. 97

6.11 The Case Where Y is Still Considered as a Separator of X and Z by PC. 97

6.12 Comparison of the PC and Augmented PC Policies With the Optimal Policy. 98

8.1 Comparison of PC, Posterior, and Optimal Causal Policies on an Uninformative Prior. 118

8.2 Comparison of PC, Posterior, and Optimal Causal Policies on an Uninformative Prior on Domains of Size 3-5 Variables. 119

8.3 Percent Improvement of the Posterior Policy Over the PC Policy in the 3-variable Domain. 120

8.4 Percent Improvement of the Posterior Policy Over the PC Policy in the 4-variable Domain. 120

8.5 Percent Improvement of the Posterior Policy Over the PC Policy in the 5-variable Domain. 121

8.6 Model Structure for the 3-variable Domain in which PC outperforms the Posterior. 122

8.7 Model Structures for the 3-variable Domain in which PC and Posterior Perform the Same. 122

8.8 Model Structures for the 3-variable Domain in which the Posterior outperforms PC. 122

8.9 Model Structures for the 4-variable Domain in which PC outperforms the Posterior. 123

8.10 Model Structures for the 4-variable Domain in which PC and Posterior Perform the Same. 123

8.11 Model Structures for the 4-variable Domain in which the Posterior outperforms PC. 124

8.12 Example Model to Demonstrate the Inference Rule Sequence Found by the Posterior Policy When Given an Uninformative Prior. 125

8.13 The Markov-Equivalence Class Containing a Set of Models Consistent With the Data That Cannot Be Distinguished Further by PC. 126

8.14 Comparison of PC, Posterior, and Optimal Causal Policies on an Informative Prior. 134

8.15 Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 3-variable Domain. 136

8.16 Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 4-variable Domain, 0- to 3-edges. 137

8.17 Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 4-variable Domain, 4- to 6-edges. 138

8.18 Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 5-variable Domain, 0- to 4-edges. 139

8.19 Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 5-variable Domain, 5- to 10-edges. 140

8.20 Example Model to Demonstrate the Inference Rule Sequence Found by the Posterior Policy When Given an Informative Prior. 141

8.21 Specification of the Misinformative Prior. 147

8.22 Comparison of PC, Posterior, and Optimal Policies Against the Posterior on a Misinformative Prior. 148

8.23 Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 3-variable Domain. 149

8.24 Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 4-variable Domain, 1- to 3-edges. 150

8.25 Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 4-variable Domain, 4- to 6-edges. 151

8.26 Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 5-variable Domain, 1- to 5-edges. 152

8.27 Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 5-variable Domain, 6- to 10-edges. 153

8.28 Comparison of the Posterior Under Different Sizes of M....................161

CHAPTER 1
INTRODUCTION

The goal of causal discovery algorithms is to construct causal models in the form of graphical models where directed edges between variables X and Y imply X is a direct cause of Y. Causal discovery can be described as a process of eliminating potential causal models from a large space of such graphical structures, each of which could explain the probabilistic dependencies found in a given domain. These algorithms differentiate among causal models by applying inference rules which generate constraints to which potential models must adhere. Specifically, an inference rule takes as input a set of known constraints and statistical information on the data in a domain such as from a conditional independence test to produce an output in the form of additional constraints.

Algorithms for causal discovery can be conceptualized as a search across a state space in which states represent sets of known constraints. Actions in this space represent the inference rules that can be applied. The end result of an algorithm is the identification of a sequence of inference rules from a given start state to a final state which may describe a small class of causal models which best fit the domain. Each inference rule in the sequence has a cost and more efficient sequences arrive at the same final state but do so with lower cost. Cost becomes highly important when an algorithm has to choose between different inference rules where some may be very accurate but expensive and others may be less accurate but relatively inexpensive.

Current algorithms such as the PC algorithm approach finding a sequence of inference rules with a rigid consistent routine. These algorithms are said to have a fixed uninformed policy. Uninformed policies do not update the selection of inference rules based on the constraints learned. The same order of inference rules is applied to each and every domain. This often leads to inefficient sequences.

This dissertation demonstrates that modifying the selection of inference by using the learned constraints to guide an algorithm's choices leads to more efficient sequences of inference rules. This approach is a dynamic informed policy that allows a learning algorithm to choose freely from among a set of rules rather than applying them in a predefined order.

1.1 Examples of Inference Rule Sequences

Assume we are attempting to identify a causal model from a four-variable domain represented by W, X, Y, and Z shown in Figure 1.1. We can make use of a conditional independence test, an inference rule that provides statistical analysis indicating if two variables are associated. In a causal model, an edge between two variables indicates association while the orientation of the edge indicates the causal direction. In a four-variable domain, there are 24 conditional independence tests because, for each of six unique pairs of variables (e.g., W, X), there are four potential separating sets that may explain any association between the pair (i.e., [], [Y], [Z], [Y, Z]). The learning algorithm should find a minimal separating set, the smallest set that establishes conditional independence for two variables, to aid later inferences about the direction of causal dependence.

For the example model, there are two conditional independence tests that would evaluate as $True$ identifying that (W, X) and (Y, Z) are not associated and thus wouldn't have an edge between them in the causal model. In the state space shown in Figure 1.1, we represent conditional independence tests that evaluate as $True$ in green blocks and those that evaluate as $False$ in red blocks. Grey indicates an unapplied test. The number in each block is the number in an ordered sequence that an algorithm would apply the test.

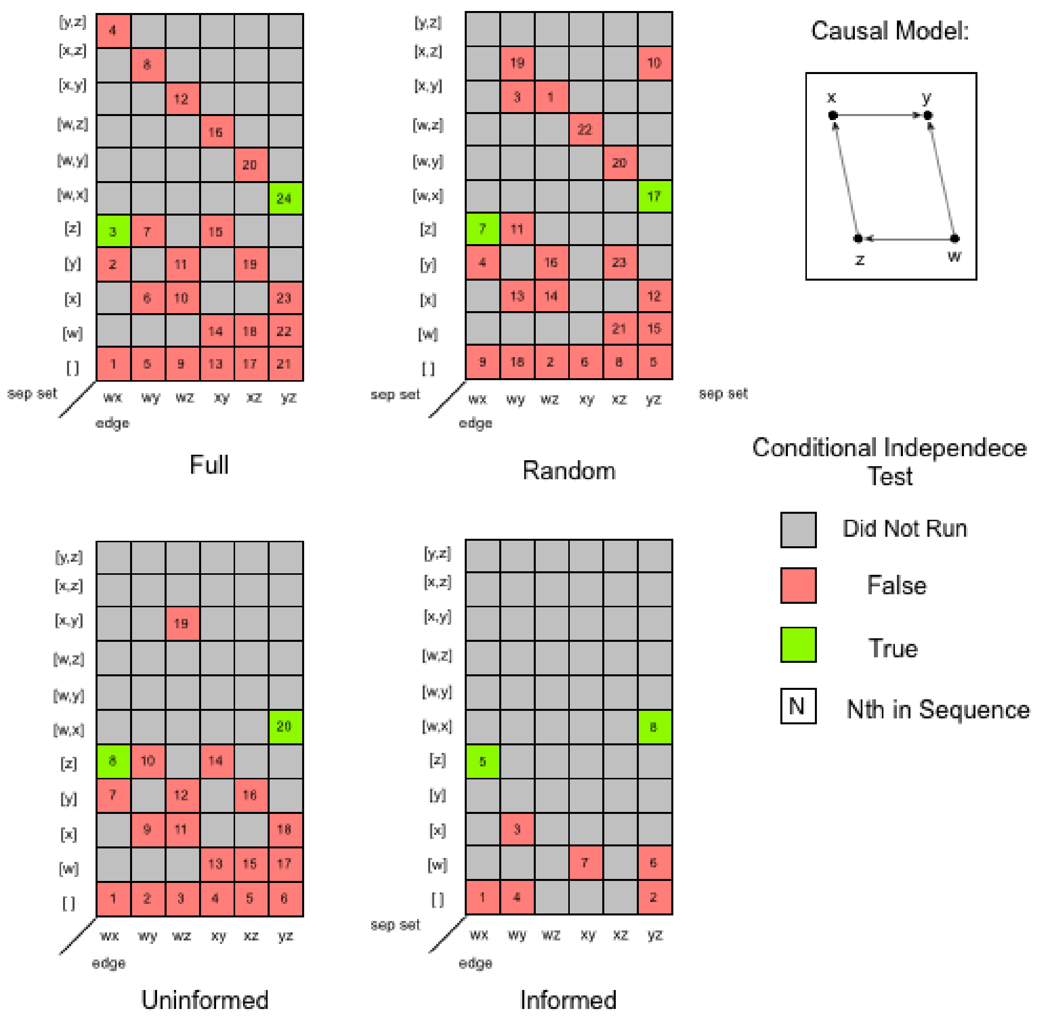


Figure 1.1. Abstract Examples of Choosing Actions Within the Set of Potential Conditional Independence Tests.

An algorithm can be designed to apply any conditional independence test in any order desired. Naively the *full* approach could be developed that applies every conditional independence test. This applies redundant conditional independence tests as there is no need to test if $[Y, Z]$ conditionally separates (W, X) if it is already known that $[Z]$ is a separating set. A *random* approach would walk through the space of inference rules and pick and apply conditional independence tests at random. But there is no guarantee that the minimal separating set is found since tests were chosen randomly. For example two variables that are marginally independent might also be conditionally independent given every potential separating set.

The *uninformed* policy represents the class of current constraint-based approaches to learning. They utilize no information about the constraints they have learned to change their behavior and thus they are uninformed. These approaches systematically apply conditional independence tests in a predefined order only skipping tests that cannot be applied or those that are unnecessary given the results of other tests. These approaches are much more efficient than the *full* or *random* approach.

An alternative, the *informed* policy, selects conditional independence tests based on the knowledge gained from applying the tests. By doing so it can choose across the space of tests instead of following a predefined order. The *informed* approach can choose those tests that eliminate alternative models faster than any other approach leading to shorter inference rule sequences. While conditional independence tests are cheap to apply, if these were other forms of inference rules that required large cost or effort, such as conducting experiments, shorter sequences would lead to faster and cheaper causal discovery.

The number of conditional independence tests available for a domain is exponential in the number of variables. As the space of conditional independence tests grows, so too does the number of inference rules an uninformed search must make. The PC algorithm, the most widely used constraint-based algorithm for inferring a causal model, relies largely on conditional independence tests. This dissertation demonstrates that even for small domains PC, which has been proven sound and complete in learning a causal model, has an exponential explosion in its inference rule sequence. An alternative informed search is presented called the Posterior policy that greatly reduces the exponential growth to arrive at the same model as does PC.

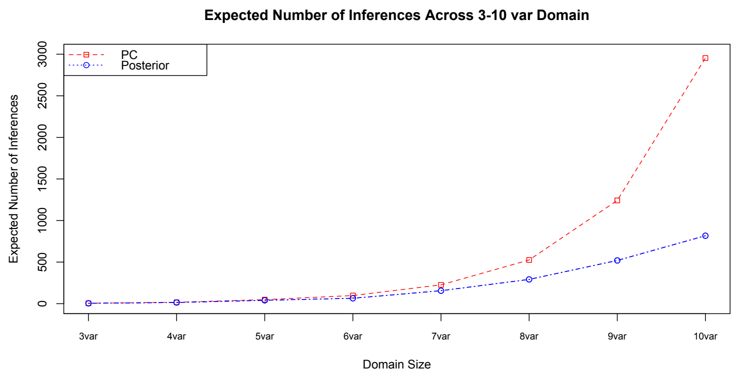


Figure 1.2. Expected Number of Inference Rules of the PC Policy vs. the Posterior Policy.

1.2 Central Ideas of the Dissertation

This work introduces a new formal framework for understanding and reasoning about the inference rules that algorithms use to infer causal structure. Three central ideas comprise this dissertation.

1.2.1 Inference Rules

The actions taken to infer causal knowledge by inference rules can be formally specified. Applying a statistical test to data in order to learn a constraint on the causal model that generated the data is a specific case of a more generalizable inference rule taken to find a constraint on the space of potential causal models. Examples of inference rules include conditional independence tests and rules applied to existing causal knowledge such as orienting the edges of a model by using conditional independence constraints.

1.2.2 Causal Learning as Search

Causal learning can be conceptualized by viewing it as a Markov Decision Process (MDP) composed of states within this space described by a set of constraints. Each state then implicitly defines a set of potential causal models which are consistent with the data for a domain. The causal MDP is solved when a policy is found that tells what action, that is which inference rule, to take in each state.

The overall learning algorithm attempts to solve for the causal MDP by performing a search for inference rules in order to move within a space of constraints to find a set of causal models which best describe the data for a domain.

1.2.3 The Benefits of Informed Search

Current algorithms for constraint-based learning of causal models solve the causal MDP using uninformed search. Uninformed search ignores the knowledge gained from applying inference rules. To an uninformed search, each domain is treated like every other domain and the sequence of inference rules used to discover a causal model is applied in the same order. Expressed another way, uninformed search has one and only one way to conduct inference. It is a fixed procedure.

But domains are incredibly complex and the number of DAGs that can represent a data set is super-exponential in the number of variables derived from the data [45, 6]. The number of conditional independence tests also grow exponentially as the domain size increases as there are $2^{(N-2)}$ possible conditional independence tests on a domain of N variables. It is not surprising then that learning a causal model has been proven to be an NP-complete problem [5].

Since uninformed search cannot discern among conditional independence tests, it can do no better than random guessing as to which tests may constrain the model space the most. Thus the fixed procedures produced by uninformed search amount to exhaustive application of conditional independence tests. Under an aggressive assumption of 1ms per conditional test, it would take $1e^{19}$ years in the worst case to identify a model for a domain with 100 variables!

Uninformed search works in most research settings because it assumes that the conditional independence tests are simple and fast. In general this is impractical and will not hold with the development of new, more powerful, statistical tests. There is much active research into new conditional independence tests based on the use of kernel methods, including reproducing kernel Hilbert spaces[16, 19]. The Kernel-based Conditional Independence test is another example that works with continuous random variables and offers reduced sensitivity to large dimensions of conditioning variables [58]. These conditional independence tests can identify conditional independencies which Chi-square tests cannot but they can be more computationally expensive.

It would not be feasible to exhaustively apply these newer independence tests. However uninformed search cannot discern which, if any, conditional independence test it should apply early on to constrain the model space. Thus uninformed search can do no better than random guessing as to which tests to apply and in which order, which is why they utilized a fixed approach.

The alternative presented is the use of informed search. Informed search uses the constraints in the state to identify a probability of transitioning from one state of knowledge to the next for each available inference rule. Using this probability it can decide which inference rules, such as conditional independence tests, will constrain the model space more and thus informed search can choose those inference rules which will ultimately lead to fewer rules applied overall.

Informed search can also utilize a prior probability distribution over the set of known constraints to inform inference rule selection. Priors can easily come from domain expertise or previous analysis and the prior can be quite useful in guiding informed search as they can better indicate the likelihood of inference rule output. Informed search can then choose to apply inference rules more likely to constrain the model space. After every inference rule, informed search can then update its knowledge transforming the previous state's prior into a posterior leading to shorter, less expensive, sequences of inference rules compared to uninformed search.

1.3 Research Questions

The principal question this dissertation seeks to answer is:

Is causal discovery made substantially more efficient by explicit reasoning about the order of application of inference rules?

This question is decomposable into smaller research questions.

- What is a method by which we can assess the benefits and costs of an inference rule given a learning goal?
- How can we search across the space of constraints to identify a sequence of inference rules that uncovers causal models?
- Given a set of inference rules, do different sequences of these rules produce the same causal knowledge with different expected costs?
- How do we determine what constitutes an optimal sequence given learning constraints?
- How optimal are current constraint-based learning algorithms?
- Can we obtain a significant and meaningful improvement in the efficiency of learning a causal model by utilizing informed causal search that is guided by prior knowledge over set of known constraints?

1.4 Contributions

Six contributions are made in this dissertation.

First, the introduction of a formalization of an inference rule describing when inferences can be derived, the actions necessary to apply the inference rule, and the outcomes or conclusions produced by the inference rule.

Second, causal learning is recast as a Markov Decision Process (MDP) composed of a set of states describing constraints and transitions between the states that occur when applying inference rules. Algorithms solve the causal MDP by identifying a policy that specifies an action to take in each state. They are differentiated in the ways they determine a policy. A component of finding a policy is the way in which these algorithms update the transition probabilities of actions across the state space.

The process of solving the causal MDP is referred to as *causal search* because learning algorithms search across states of constraints by choosing a sequence of inference rules. In this perspective of learning, algorithms use their policies to find paths through the space from the initial state to some goal state (typically, a Markov-equivalence class of model structures). These paths are *inference sequences.*

Third, a general framework for executing different learning algorithms is implemented that allows their behavior to be observed. This framework is designed to be independent of policy so any learning algorithm can be used. This architecture allows policies to be easily implemented and evaluated. To facilitate this architecture the policy of a learning algorithm is represented by a *causal plan* that specifies the actions taken at each state and handles the contingencies over the possible outcomes from applying that action.

Fourth, the idea of *uninformed search* and *informed search* for causal structure is introduced. Algorithms with a fixed transition probability conduct an *uninformed search* across the state space. These algorithms take no advantage of the outcomes of derived inference.

Alternatively, algorithms that utilize *informed search* update transition probabilities based on what they learn.

Fifth, in order to compare informed search against uninformed search the *Optimal policy* to learning causal structure is developed and the PC algorithm is converted into the *PC policy*. The Optimal policy selects the most efficient sequence of operators that moves from some initial state of causal constraints to the most descriptive state reachable in the state space. This policy represents what a policy can do if given perfect information about the domain for which it is learning a causal model. A policy to reflect the behavior of the PC algorithm is created, and this policy allows determination of the efficiency of PC as well as comparison of the performance of PC against other algorithms.

Finally, one kind of informed search, the *Posterior policy*, is presented that takes as input a prior over the constraint space. This prior is used to compute an initial transition probability between states of constraints. The Posterior policy uses Bayesian updating of this prior based on the outcomes of applying inference rules which in turn is used to update the transition probabilities across the state space. This leads to improved performance over PC. In particular the PC policy identifies inference rule sequences of length exponential in the size of the domain while the Posterior policy identifies inference rule sequences that are much smaller.

CHAPTER 2
BACKGROUND

This background gives two examples of inference used to discover the underlying causal model of two different domains. The representation of the causal model is then formally described as a Bayesian network and how these graphical representations can be causally interpreted is explained. Partially oriented graphical models and how they represent an entire class of causal models that are Markov equivalent are described.

Current algorithms for constraint-based learning of causal models build up a set of constraints to learn Markov-equivalence classes of models that explain a domain. Examples of common inference rules used by these algorithms are given and the important details of the most often used constraint-based algorithm, the PC algorithm is explained.

The foundations for causal search are shown by explaining how applying inference rules is like taking an action that transitions the state of knowledge from one set of constraints to another. An important concept in understanding causal search is the MDP and the basics behind MDPs is explained. Solutions to the MDP known as policies will form the core of this work.

2.1 Examples of Inference Rules to Find Causal Models

Choosing and applying inference rules in order to constrain the model space and learn the graphical model can be demonstrated through two examples. The first example demonstrates the power of experimental analysis where direct manipulation of variables is possible. The second example demonstrates the power of inference that can be derived from observational data.

2.1.1 Photobase: An Example of Causal Discovery Using Experimentation

In earlier work a system called Photobase was developed to experimentally evaluate if causal questions within collaboration systems could be answered. Photobase was a small mobile phone application that allowed people to take photos across the University of Massachusetts campus, to share those photos with others, and to rate photos. This experiment investigated if there were ways people could be encouraged to contribute and participate and if how people behave within a collaboration system could be understood. Photobase was designed to help address questions such as "what causes people to take photos?" and "what causes people to take better photos?"

A randomized experiment was used in which individuals using Photobase were randomly assigned to either a control or a treatment group. There were two treatment groups each with different collaboration perspectives. Some people were told they were working in a competitive environment, one where they could earn points based on performance. Some people were told they were part of a coordinated effort where everyone was given a set of assignments. The control group were given no explicit perspective and only told that they were part of this overall photo sharing system. The effect measured was a set of variables all meant to represent aspects of participation. These included number of photos taken, the frequency of photo taking, and how users rated their content as well as the content from others.

The results of experiments demonstrated that the type of collaboration influenced the number of photos that were taken. Specifically, competitive users took fewer photos than coordinated users and those in the control group. It was also found that competitive users over inflate the value of their contributions indicating a direct cause from the type of collaboration to how people self rate. When photos from a single specific region on campus were deliberately omitted, only the people who were coordinated and told they were responsible for taking photos in this area actually went there and took photos. It was very unlikely for anyone else to take those photos. This indicates that coordination was a direct cause for improved data coverage in the experiment.

2.1.2 Stack Overflow: An Example of Causal Discovery Using Observational Data

Stack Overflow is a collaborative environment where users ask and answer questions related to computer technology. Stack Overflow is highly successful, and site administrators and others are understandably interested in determining which incentives (e.g., badges, points, and levels) encourage user participation. Fortunately, Stack Overflow makes a vast amount of data on daily operation freely available. This means that researchers have gigabytes of collected observational data on users, the posts in the systems, voting of popularity on these posts, etc.

In some earlier work, we applied conventional social science methods to study causal dependence within Stack Overflow [38]. For example, one type of inference rule that we applied to the Stack Overflow data set was the matched quasi-experimental design [47, 4]. The matched design identifies pairs of units where one has received a treatment and the other has not but who are otherwise similar. A unit is an object within the domain such as a user in the system. Thus a matched pair of units may be two people or two objects of the same type. The validity of the conclusions drawn from a matched design improves as the matched pairs of units become more alike.

The matched design was used to answer one of several causal questions: "Does posting a high-quality answer for a particular question cause other users to stop providing answers for that question?" This answer would be very interesting to developers of a collaborative system because it investigates if participation is impacted by perception. If high-quality answers to any question are limited, then once a high quality answer appears, further participation may decline because people may feel they cannot provide anything meaningful to the conversation. This may discourage users from participating and reduce the overall success of the system.

With this matching design, a positive increase in the answer rate was identified for the first 10 minutes around the occurrence of the high-quality answer and then no meaningful effect from that time onward. Perhaps surprisingly, having a high-quality answer actually increases the answer rate when compared to the comparison group! Detailed inspection of the data revealed that posting an answer may take on average 5 to 10 minutes. When a new question is asked, it can generate a burst of activity. Often there is at least one other user who is in the process of typing an answer around the time the eventually accepted answer is also being written. It can be concluded then that it was not that high-quality answers increased the answer rate, but that a number of users are simultaneously typing their answers for the question, one of which becomes the high-quality accepted answer.

2.2 Bayesian Networks and d-separation

For a given domain, characterized by a set of variables V, a causal graphical model describes all direct causal dependencies between pairs of variables $X, Y \in V$. Those dependencies can be represented using a directed, acyclic graph (DAG), also known as a Bayesian network [41].

Bayesian networks are DAG representations of a high-dimensional joint probability distribution P over a set of variables, V. Formally a ***Bayesian network*** $B = \{G, \theta\}$ consists of an acyclic, directed graph $G = \{V, E\}$ containing vertices and edges and a set of conditional probability distributions, θ. An edge E between two variables X and Y indicate a dependence between the variables. The parents $pa(v)$ of a variable $v \in V$ are the set of variables with directed edges pointing to the variable 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 [41]. An example of the Photobase domain as a Bayesian network is shown in Figure 2.1.

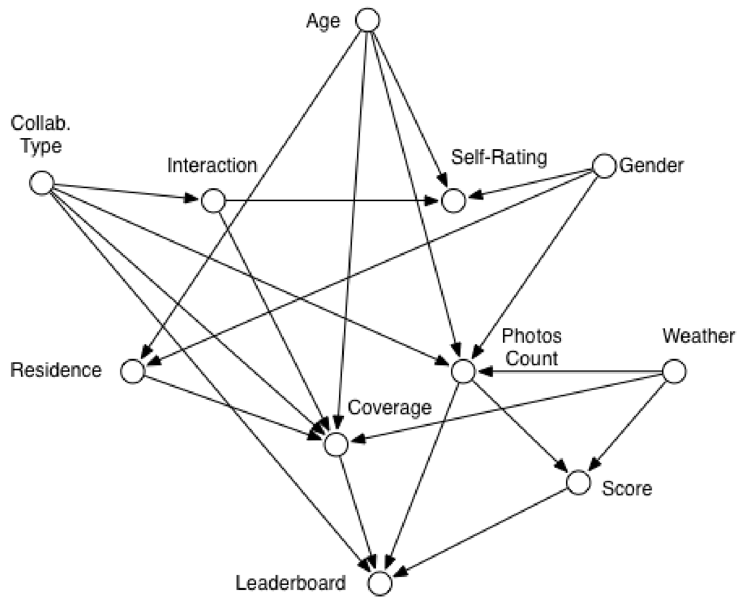


Figure 2.1. A Bayesian Network for the Photobase Domain.

The graphical structure encodes a set of ***conditional independence*** assumptions: each variable $v \in V$ is conditionally independent of its non-descendants given its parents, $pa(v)$. The correspondence between conditional independence relations and certain graphical structures is summarized by the ***d-separation criterion***.

Definition (from Pearl [40, 41]) A path p is said to be d-separated (or blocked) by a set of nodes $\mathbf{Z}$ if and only if

1. p contains a chain $x \longrightarrow z \longrightarrow y$ or $x \longleftarrow z \longleftarrow y$ or a fork $x \longleftarrow z \longrightarrow y$ such that the middle node z is in $\mathbf{Z}$, or

2. p contains an inverted fork (or collider) $x \longrightarrow z \longleftarrow y$ such that the middle node z is not in $\mathbf{Z}$ and such that no descendant of z is in $\mathbf{Z}$.

A set $\mathbf{Z}$ is said to d-separate X from Y if and only if $\mathbf{Z}$ 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 and describe the graphical scenarios that entail these independence relationships. When nodes in a DAG are d-separated, they are conditionally independent; when they are d-connected, they may be dependent.

2.3 Causal Interpretation of DAGs

There are four graphical structures representing the causal structures that can lead to the association between X and Y shown in Figure 2.2.

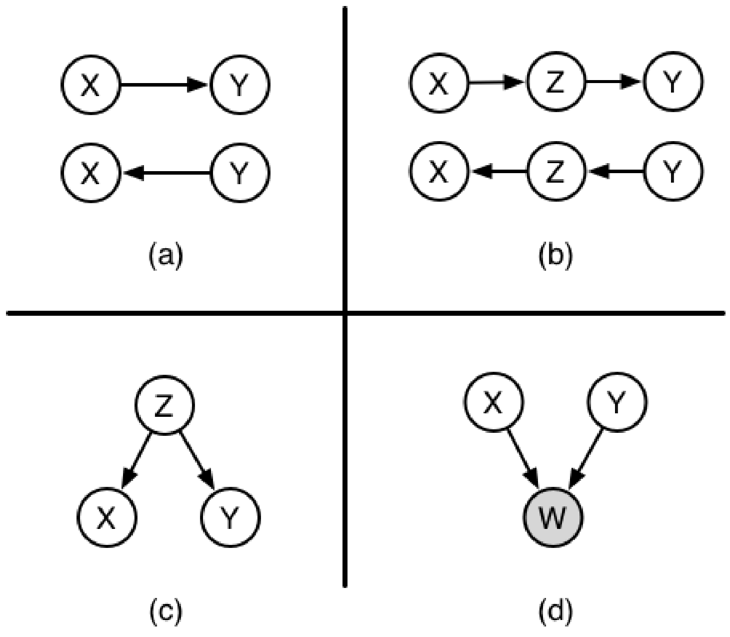


Figure 2.2. Four Causal Structures That Give Rise to a Statistical Association Between X and Y. In (d) X and Y Have Been Conditioned on W.

It could be that X is a direct cause of Y (Figure 2.2(a)). In this case, one may set the value of X and have a direct influence on the value of Y. It could also be that Y is a direct cause of X (Figure 2.2(a)). If this is true, one can affect the value of X by setting the value of Y.

It may be that some variable Z is part of a chain between X and Y (Figure 2.2(b)). When there is a chain where X is a direct cause of Z and Z is a direct cause of Y, then the value of Y is indirectly affected by setting X. Likewise, when there is a chain where Y is a direct cause of Z and Z is a direct cause of X, the value of X can be indirectly affected by setting Y.

A third possibility is that some other variable Z is a common cause of both X and Y meaning that Z is a direct cause for both of them (Figure 2.2(c)). This structure implies that setting X has no effect and instead it is Z that must be set. There may also be a variable W that is a common effect of both X and Y (Figure 2.2(d)). If conditioned on W, or a descendant of W, then a dependence may be induced between X and Y, known as Berkson's paradox, or "explaining away." Under this causal structure no value of X will have any effect on Y.

When certain assumptions hold, the directed edges in a Bayesian network can be read as causal structures allowing a causal interpretation of the graph [41, 42, 50]. These assumptions enable d-separation to serve as a method to directly relate observable conditional independence and the underlying structure of the causal model. Because the PC algorithm relies on these assumptions, these assumptions will also be inherited by this work.

2.3.1 Causal Markov Condition

A directed acyclic graph G over V and a probability distribution P satisfy the Causal Markov Condition if and only if for every $X \in V$, X is independent of $V \backslash (Descendants(X) \cup Parents(X))$ given the Parents(X) [22]. The Causal Markov Condition means that if two variables X and Y are distributionally dependent, then there is a causal relationship between them.

2.3.2 Faithfulness

P is faithful to G if and only if there are no conditional independence relations true in P that are not entailed by the Causal Markov Condition applied to G [51]. This assumption can be informally thought of as explaining that all of the conditional independencies in G are due to the true underlying causal model rather than to some chance distribution of the data in D.

Researchers also tend to make a third assumption that allows interpretation of the causal model as being complete.

2.3.3 Causal Sufficiency

V is causally sufficient if and only if for every pair of variables $X, Y \in V$, any common cause of X and Y, Z is also found in V. Causal sufficiency can only be achieved if there are no latent or hidden variables Z that are common causes of X and Y [57].

2.4 CPDAGs and Markov-Equivalence Classes

Many current methods for constraint-based learning of causal models produce a representation of the Markov-equivalence class of causal models called a completed partial DAG (CPDAG) [6]. The CPDAG is interpreted as a representation of an entire class of causal DAGs because it constrains the space of causal models but does not exactly specify it. The CPDAG contains edges that are either directed or undirected.

Each directed edge in a CPDAG is a specification that the valid causal model must contain the same edge oriented in the same way. However, any undirected edges between two variables X and Y in the CPDAG can be oriented as $X \longrightarrow Y$ or $X \longleftarrow Y$. Any DAG from this equivalence class represented by the CPDAG is indistinguishable from any another DAG within the set because they all describe the exact same set of conditional independence relations and thus the set is referred to as a Markov-equivalence class.

Formally, assume $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are two DAGs across variables V. G_1 and G_2 are ***Markov equivalent*** if for every disjoint X, Y, and $\mathbf{Z} \in V$, X and Y are d-separated by $\mathbf{Z}$ in G_1 iff X and Y are d-separated by $\mathbf{Z}$ in G_2.

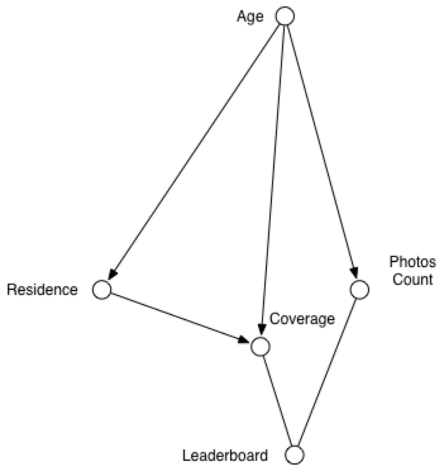


Figure 2.3. CPDAG example of the Photobase Causal Model.

An example of a CPDAG within the Photobase domain is shown in Figure 2.3. Assume that *Age* is intrinsic and nothing in the domain is causal for it. One possible set of causal dependencies is that *Age* is causal for *Residence* (i.e., freshman live on campus), *Age* is causal for *Coverage* (i.e., younger people walk across campus more), and that *Age* is causal for *PhotosCount* (i.e., older students take more photos.) Where people live may be found to be causal for *Coverage* as well. It may be possible to establish that the presence of a *Leaderboard* ranking individual performance is statistically associated with *Coverage* and further to establish a causal dependence between *Leaderboard* and *Coverage*, but the direc-

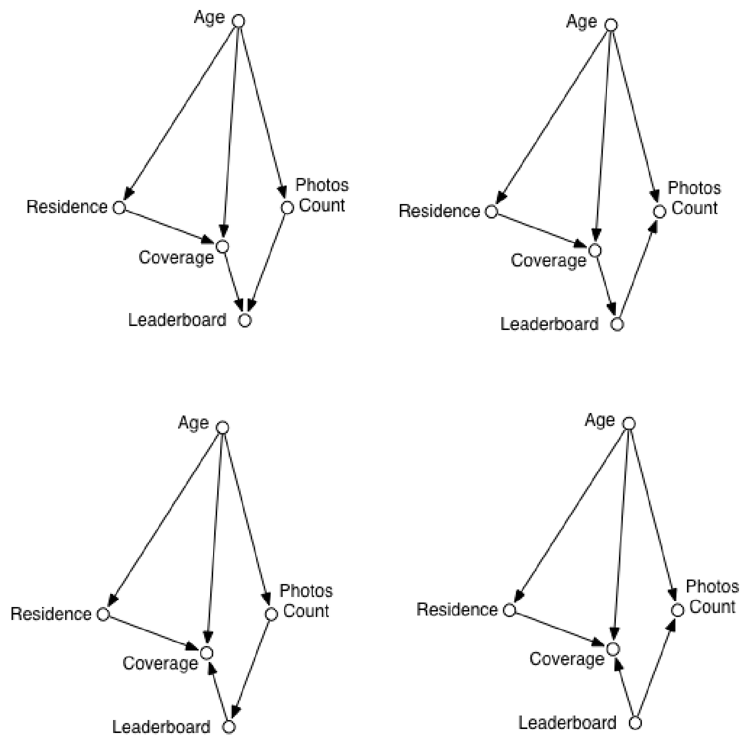


Figure 2.4. The Markov-Equivalence Class for the Photobase CPDAG.

tionality of the causal dependence may not be clear from the data. The same may hold for *PhotosCount* and *Leaderboard*.

Under such a situation, there are four possible causal models in this Markov-equivalence class shown in Figure 2.4.

2.5 Causal Models

Causal models are found within many disciplines including the social sciences [25], climate change [31], genomics [56], understanding election results [1], and identifying the causal relations between poverty and employment [17].

There are an exponential number of DAGs that can represent causal models across a domain of N variables as given by Robinson's equation to compute the number of DAGs [45]:

$$a(N) = \sum_{i=1}^{N} -1^{i+1} \binom{N}{i} 2^{i(N-i)} a(N-1) \tag{2.1}$$

For example with a 3-variable domain, $N = 3$, there are a potential 25 causal models. For 4-variables, there are 543, 5-variables: 29281 and so on. The goal in uncovering the most-consistent causal model for the domain is very much a process of eliminating alternative models from the model space by discovering constraints that models must satisfy. Constraints can be derived from many sources, including from the results of conditional independence tests that establish that two variables are not associated, the outcome from applying an experimental design such as an A/B test [27, 28], or inferred through logical reasoning over other existing constraints [9, 35].

2.6 Constraints on the Model Space

As the constraints on the model space are identified, the set of potential causal models that explain the data are reduced. If two variables X and Y are found marginally independent then models that have X a cause of Y or Y a cause of X need no longer be considered.

A large class of causal discovery algorithms are constraint-based and directly depend on the causal faithfulness assumption to identify conditional independencies and dependencies between variables X and Y in V and thus to identify the edges between X and Y in the causal model G [49]. Constraint-based approaches have been proven to be correct and sound in the sample limit [42, 50]. These algorithms can exhaustively search over all pairs of variables constructing every possible conditioning set to uncover the skeleton of the causal model. A second phase then uses edge-orientation rules such as Meek's rules [34] to identify directionality on the edges for as many pairs of variables as possible.

2.6.1 Conditional Independence Tests

Let X and Y be random variables. Denote the joint density of (X, Y) as $p(x, y)$, the marginal density of X as $p(x)$, the marginal density of Y as $p(y)$, and the conditional density of X given $Y = y$ as $p(x|y)$. Informally, X and Y are marginally independent, denoted by $X \perp\!\!\!\perp Y$, if knowing the value of Y provides no information about the probability distribution of X. Formally, the mathematical expression of this property in terms of densities is $p(x, y) = p(x)p(y)$ [10]. Let Z be a random variable. Write $X \perp\!\!\!\perp Y|Z$ to denote that X and Y are conditionally independent in their joint distribution given $Z = z$, for any value of z. This can be written as the mathematical expression $p(x, y|z) = p(x|z)p(y|z)$.

There are many different tests that can be used to assess if X and Y are marginally or conditionally independent. The Pearson Chi-Squared Test is often used to test for marginal independence between two variables X and Y[18]. The class of Cochran-Mantel-Haenszel (CMH) tests look for association of (X, Y) given a conditioning variable set Z [55].

2.6.2 Edge-Orientation Rules

Once the undirected graph has been identified, Meek's edge-orientation rules based off of d-separation criteria are used to direct the dependencies. These include rules for edge orientation called collider-detection, known non-collider detection, and cycle avoidance. Each orients edges in the DAG based on the learned independence constraints. Technical details can be found in section 3.2.1.3.

Collider detection specifies the conditions where one can orient the direction of a causal dependence as $X \longrightarrow Z \longleftarrow Y$. This structure is referred to as an *inverted fork*. An

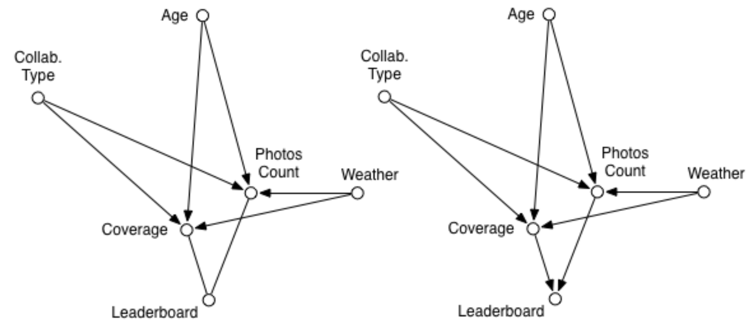


Figure 2.5. The Collider Detection Rule is Used to Orient *Coverage* and *PhotosCount* as Parents of the *Leaderboard.*

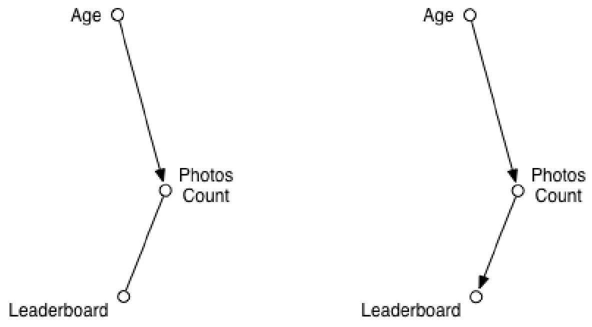


Figure 2.6. The Known Non-Collider Rule Relies on the *Chain* of Variables in a Path to Orient *PhotosCount* to the *Leaderboard.*

example of the result from applying the collider detection rule is shown in Figure 2.5. Since *Leaderboard* is not in the set of variables that d-separate *Coverage* and *PhotosCount*, then they must both be parents of *Leaderboard* or else there would be an edge between *Coverage* and *PhotosCount*.

The known non-collider rule identifies a situation where a causal chain occurs from X through Z to Y. In the example shown in Figure 2.6, if *Age* is causal for *PhotosCount*, and there is no edge between *Age* and *Leaderboard*, then *PhotosCount* must be causal for *Leaderboard.*

Because directed acyclical graphs are used to represent causal models, if it is discovered that X is directly causal for Y, then any directed path from X to Y through Z must be similarly directed. If not, then acyclicity would be violated.

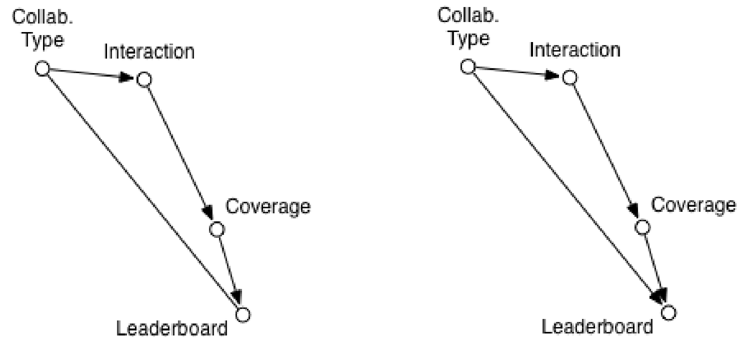


Figure 2.7. The Cycle Avoidance Rule Preserves Acyclicity by Orienting *CollaborationType* to *Leaderboard*.

2.7 Constraint-Based Causal Learning

The dominant group of constraint-based approaches to the automated learning of a causal model is the PC algorithm [50] and its family of extended variations. The PC algorithm is composed of two phases: a phase that identifies constraints from statistical associations between pairs of variables by applying independence tests and an edge-orientation phase that applies logic over the constraints to identify that causal directions between related variables that entail the learned constraints [50]. PC has been shown to be sound and complete and has been extended into additional algorithms to relax assumptions of the original algorithm or apply to different data representations. These include the FCI algorithm [50], the CPC algorithm [43], the RFCI algorithm [8], and the RPC algorithm [33].

PC and its extensions assume a fixed-form solution to inferring a causal model and approach every domain as if it were the same. The first step of these algorithms can be thought of as ordering every possible conditional independence test from simplest to most complex followed by applying every test in order.

Phase1: Skeleton Construction

```
1. Initialize graph $G$ to be a complete, undirected graph over the vertex set $V$.
2. $n \longleftarrow 0$
3. While $n \leq 2^{|V-2|}$:
4.     For ordered pair of variables $(X, Y)$ in $G$:
5.         If Adjacencies($X$, $G\backslash\{Y\}$) has cardinality $\geq n$:
6.             For each subset $\mathbf{Z}$ of Adjacencies($X$, $G\backslash\{Y\}$) where $|\mathbf{Z}| = n$:
7.                 If $X$ and $Y$ are d-separated given $\mathbf{Z}$:
8.                     Delete edge $(X, Y)$ from $G$
9.                     Record $\mathbf{Z}$ in SepSet($X$, $Y$) and SepSet($Y$, $X$)
10.    $n \longleftarrow n + 1$
```

Phase 2: Edge Orientation

```
1. For each triple of vertices $X$, $Y$, $Z$:
2.     If the pair $X$, $Z$ and the pair $Y$, $Z$ are each adjacent in $G$:
3.         If the pair $X$, $Y$ are not adjacent in $G$:
4.             [Collider Detection] Orient $X$ - $Z$ - $Y$ as $X \longrightarrow Z \longleftarrow Y$ if and only if $Z$ is
   not in Sepset($X$ , $Y$)
```

```
5. For each triple of vertices $X$, $Y$, $Z$:
6.     If $X \longrightarrow Z$:
7.         [Known Non-Collider Detection] If $Y$ and $Z$ are adjacent and undirected and $X$ and $Y$ are not adjacent:
8.             Orient $Z$ - $Y$ as $Z \longrightarrow Y$
9. For each pair of vertices $X$, $Y$:
10.    [Cycle Avoidance] If the pair $X$, $Y$ are adjacent in $G$ and there is a directed path from $X$ to $Y$:
11.        Orient $X$ - $Y$ as $X \longrightarrow Y$
```

In many cases the result of PC's edge-orientation is a CPDAG.

2.8 Inference Rules and Causal Inference

Inference can be thought of as composed of basic atomic operators for reasoning about the posterior probability distribution of the causal model space. These operators can be represented by inference rules that describe when constraints on the model space hold. For brevity within this text causal inference rules are referred to as inference rules or rules where the context is clear. Inference rules are also referred to as being actions that can be taken that correspond to actions in the traditional view of AI planning.

The efficiency of any automated learning technique, such as the PC algorithm, depends on the inference rules it applies, and the primary inference rule used by these algorithms is the conditional independence test. However these are a limited set as there are many different inferences that can be used to construct causal models. For example the incorporation of prior knowledge can be encoded into an initial Bayesian network to help refine the causal structures learned on the observational data [23]. Prior knowledge can also be used to assist in orienting causal directions.

2.9 Markov Decision Processes

One framework for characterizing the process of applying inference rules is a Markov decision process (MDP). An MDP is a specification of a sequential decision problem for a fully-observable environment with a Markovian transition model and additive rewards or costs [46]. Causal learning can be described as an MDP because the inferences can be treated as actions that when applied, transition from one set of constraints to another.

A description of an MDP in terms of action costs [21] is:

- MDPs have a finite set of states.
- Each state s has a set of actions $A_i(s)$ available in that state.
- There is an expected immediate cost, $C_a(s)$, of taking action a in state s.
- A terminal state is a state where no action can cause a transition into another state.
- The immediate cost of any action in a terminal state is zero.
- The goal is to reach a desired state, which may be a terminal state, with the minimum expected cost.

A complete solution for an MDP, called a ***policy***, specifies the actions an algorithm should take for any state it reaches [46]. A policy is denoted by π, and $\pi(s)$ is the action recommended by the policy in state s. Policies are the output of a learning algorithm and the same algorithm applied to different domains will generate different policies.

If the algorithm has a complete policy, no matter the outcome of any action, it will always know what to do in any given state. The quality of the policy is measured by its expected utility. This work measures utility in terms of cost, C. The lower the cost, the higher the utility. An optimal policy, π^*, is a policy that yields the highest expected utility under any environment.

There are several methods to solving an MDP. Dynamic programming is an approach to solving an MDP that works by evaluating the utility of the entire state space. Two dynamic programming techniques are the value-iteration and policy-iteration algorithms though both have computational challenges in the super-exponential state space of causal learning.

The Bellman Equation defines the utility of a state as the immediate cost for that state plus the expected discounted utility of the next state when the agent or search chooses the optimal action [46].

$$U(s) = C(s) + \gamma \min_a \sum_{s'} T(s, a, s')U(s') \tag{2.2}$$

where $T(s, a, s')$ is the transition function describing the next state s' from state s found after applying action a, $C(s)$ is the cost to reach state s, and $U(s')$ is a measure of the utility of state s'. The value-iteration solution uses a Bellman equation at each state and updates the utility of each state by starting with initial values and propagating utilities to neighbor states until the equations reach a point of stability. At that point, starting from the initial state it becomes easy to decide the best action at every state and determine the optimal policy.

An alternative is the real-time dynamic programming (RTDP) algorithm that only performs policy-iteration for those states reachable from a start state [2]. This approach is potentially useful to causal learning as the learning task is an optimal solution to the shortest-path problem from a given start state, not from every possible state.

2.10 Related Work

The active learning of causal networks is similar to the idea of using a policy to solve the causal MDP. Active learning attempts to differentiate among the causal models in the Markov-equivalence class by combining interventional analysis with observational analysis. The core challenge that active learning attempts to solve is how to decide which intervention should be applied next given the state of current causal knowledge. There are three approaches to solving this problem: the information-theoretical approach, the decision-theoretical approach, and the game-theory approach.

The information-theory approaches make use of a scoring metric, typically an estimate of the loss of entropy, to discern the value of knowledge about a single undirected edge in the graph [53]. A decision-theoretic method relies on the utility of each intervention. While information theory ranks decisions by their entropy, preferring choices with lower entropy, decision theory ranks decisions by their utility and chooses the option of the highest utility. The utility function is intricate and can be based on several factors including a measure of the number of edges which are oriented as a consequence of performing the intervention [35] or the cost to perform an intervention. When all actions cost the same, the utility function is a special case of the entropy criteria and the approach is very similar to the information theoretic approach [37].

A third approach is to view the causal discovery task within game-theory where there are two players: Nature and the scientist [11, 12]. Nature and the scientist can be seen participating in a zero-sum game where the pay-off for Nature is the number of experiments necessary for the scientist to discover the causal model balanced by the loss to the scientist for their effort. This approach results in mixed-strategies where interventions are randomly selected under the assumption that all interventions are possible [36]. However, in general interventions may be expensive and a practical approach would be to search through the potential set of interventions and select the ones with the highest value to carry out.

CHAPTER 3
INFERENCE RULES

Once an inference becomes atomic, learning algorithms can be designed to use inference rules in any order that facilitates causal discovery. Experience from the manual discovery of causal structure from Photobase and Stack Overflow highlighted the need to carefully select which inference rules to apply and in what order.

The process taken to infer causal knowledge can be formally represented by inference rules. For example, applying a statistical test to data in order to learn a constraint on the causal model that generated the data is a specific case of a more generalizable action. Premises can be used in conjunction with these actions to produce inferences that constrain the space of causal models. Many different actions can be used including interventions [11, 12], quasi-experimental designs [26] and observational studies. Not all inference rules need an action. Consider those rules that can reason over existing causal knowledge such as orienting the edges of a model using the results from previously executed conditional independence tests.

Inference rules include the specification of the necessary actions required to evaluate the rule, definitions of the conditions when the rule can be applied, and the constraints generated as an output of the rule. In this representation, the inference rules are modular allowing an algorithm to determine which rules are available to be applied, to assess the costs and benefits of each rule, and then to select and apply the rules in any desired order. An additional benefit is that once formalized the inference rules become modular components that can be added or removed as needed without requiring that the algorithm be re-designed.

This is an improvement over current algorithms that are designed to work specifically with only a small subset of inference rules.

Constraint-based techniques such as the PC algorithm have a predefined order of inference rules composed from a limited set of independence tests and edge-orientation rules. Each conditional independence test for a domain is ordered from simplest to most complex. These algorithms then proceed to apply each of those tests in order. Systematically applying every rule will eventually uncover the causal model, but when there are $2^{(N-2)}$ possible conditional independence tests on a domain of N variables, it quickly becomes computationally impractical.

The problem of efficiency exists because these algorithms do not change the selection of inference rules based on past inference results. They approach learning as if rules were not modular and for these algorithms learning is simply a chain of successive and fixed inference rules. The order of inference rules they consider results in being the same for every domain.

3.1 Inference Rule Representation

Describing the action of learning an inference as a formal rule enables generalization of the actions taken to infer causal relationships. Analysis becomes modular by encapsulating inference and can be combined with flexible reasoning mechanisms enabling a novel approach to learning.

Formally, a causal inference rule is an inference rule whose antecedents are structural (variables, temporal, etc.), causal, statistical, and whose consequent is a probabilistic or deterministic constraint on the space of causal models. In this dissertation, only deterministic constraints will be considered.

A visual representation of an inference rule is depicted in Figure 3.1.

In this representation antecedents are decomposed into two parts: input constraints and steps. The input constraints correspond to assumptions, describe necessary relationships

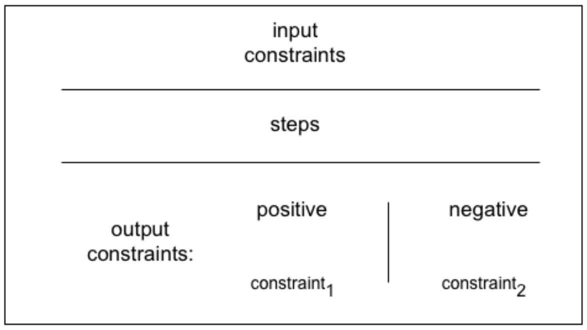


Figure 3.1. Formalized Representation of an Inference Rule.

between variables, or specify temporal conditions on the data. These are constraints that would exist in a knowledge base and require no more than a check for their presence and truth value.

As soon as the input constraints are valid, the inference rule can be applied. During application is when the steps in the rule can be taken. The steps are activities that generate additional information necessary to determine the output of the rule. Steps can be activities such as performing a statistical test, obtaining missing information through data gathering, or to wait for an event such as monitoring a metric until it satisfies a criteria.

It is after all steps are completed that the rule generates outputs that constrain the space of potential models explaining the domain. There are one of two possible output states[1]: True (positive) and False (negative). If the output of an action evaluates to True, then all of the positive output constraints are added into the knowledge base. If any of the actions evaluate to False, the rule generates all of the negative output constraints.

Note that when there are no steps for the inference rule, then as soon as an inference rule input constraints are met, the output is always True. An example of this would be the

[1]In reality it is possible that there could be statistical power issues with a test that might mean you have a third state, an inconclusive result. But PC does not assume this.

edge-orientation rule. Once all of the edge-orientation conditional independence criteria are met, it can immediately be applied to orient the direction of a causal dependence.

3.2 Inference Rule Examples

For all inference rules, define the variable space as $V \supset U, O$ where O are the set of observed variables for which there is data and U are the set of latent or unobserved variables for which there is no data. It is useful to represent V in this manner because it allows for the description of assumptions over O and U.

3.2.1 Observational Inference Rules

Several forms of observational analysis are described in the following sections as inference rules. These include the formal specifications of conditional independence, conditional dependence, and edge-orientation rules.

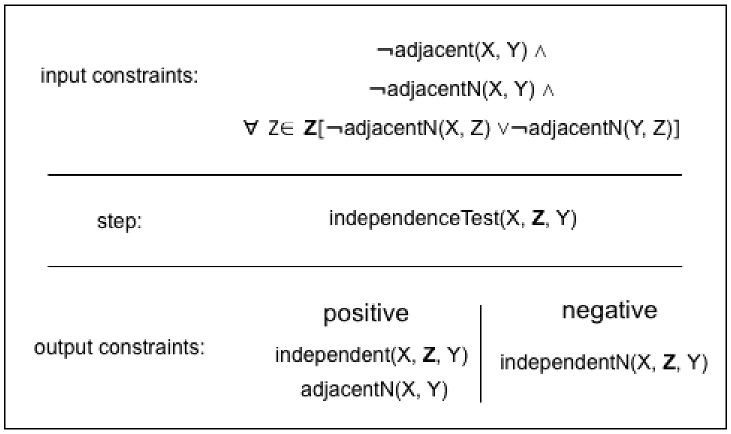
3.2.1.1 The Conditional Independence Rule (CIRule)

The conditional independence tests used by algorithms in constraint-based learning are easily translated into inference rules. The core of a conditional independence test is the introduction of new constraints on the space of causal models based on running an independence test [10]. The conditional independence rule, $CIRule(X, \mathbf{Z}, Y)$ specifies the variables under test, X and Y, and a set of conditioning variables Z.

The input conditions for the $CIRule$ are as follows:

$$\begin{gathered} \neg adjacent(X,Y) \quad \wedge \quad \neg adjacentN(X,Y)) \quad \wedge \\ \forall Z \in \mathbf{Z} \quad \neg(adjacentN(X,Z) \quad \wedge \quad adjacentN(Y,Z)) \end{gathered}$$

where $adjacent(X, Y)$ is a predicate indicating X and Y are connected by an edge in the graphical model indicating they are statistically associated. $adjacentN(X, Y)$ is a predicate indicating X and Y are not connected by an edge in the graphical model.



input constraints:

¬adjacent(X, Y) ∧

¬adjacentN(X, Y) ∧

∀ Z∈ **Z**[¬adjacentN(X, Z) ∨¬adjacentN(Y, Z)]

step: independenceTest(X, **Z**, Y)

output constraints:

positive	negative

independent(X, **Z**, Y)
adjacentN(X, Y)
	independentN(X, **Z**, Y)

Figure 3.2. The CIRule Formal Specification.

The rule is unnecessary when something is known about edge (X, Y), specifically when X and Y are already known to be adjacent in the causal graphical model, identified as $adjacent(X, Y)$ or not adjacent in the graphical model, identified as $adjacentN(X, Y)$. An *adjacency* corresponds to the condition where two variables X and Y are statistically associated or potentially associated.

Each $Z \in \mathbf{Z}$ should be potentially adjacent to either X or Y represented by ensuring each Z is not non-adjacent to X and Y. Otherwise Z cannot be on a directed path between X and Y and cannot explain any possible association. Within the step a general $independenceTest$ is shown but a specific independence test can be specified along with the evaluation criteria for interpreting the test. If the result of the $independenceTest$ is True, then the inference rule generates both a non-adjacency constraint and an independence fact:

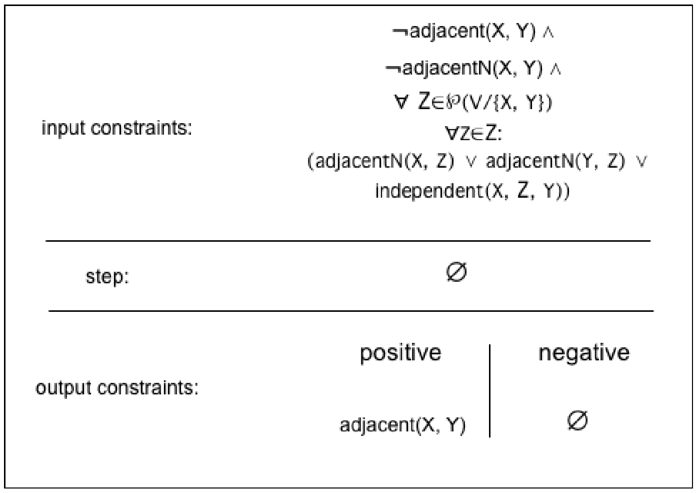
$$adjacentN(X, Y) \wedge independent(X, \mathbf{Z}, Y)$$

If the result of the test if False, then the separating set does not explain the association between X and Y and the output constraint is only a non-independence fact:

$$independentN(X, \mathbf{Z}, Y).$$

3.2.1.2 The Conditional Dependence Rule (CDRule)

The conditional dependence rule, $CDRule(X, Y)$ makes explicit the notion that if no conditional independence rule could d-separate two variables, then there must be an edge between them.



input constraints:

$\neg$adjacent(X, Y) $\wedge$
$\neg$adjacentN(X, Y) $\wedge$
$\forall$ Z$\in\wp$(V/{X, Y})
$\forall$Z$\in$Z:
(adjacentN(X, Z) $\vee$ adjacentN(Y, Z) $\vee$
independent(X, Z, Y))

step: $\varnothing$

output constraints:

positive	negative
adjacent(X, Y)	$\varnothing$

Figure 3.3. The CDRule Formal Specification.

This inference rule, while not explicitly mentioned, is a component of constraint-based algorithms that assert an edge between two variables after all conditional independence test across those two variables have been exhaustively applied without finding an independence [50]. Implicitly after every potential conditional independence rule has been applied on a pair of variables, if no separating set has been found, and under the assumptions of causal sufficiency, the pair of variables are considered to be adjacent. This is represented by the $CDRule(X, Y)$ rule with the input conditions:

$$\neg adjacent(X, Y) \quad \wedge \quad \neg adjacentN(X, Y) \quad \wedge$$
$$\forall \mathbf{Z} \in \wp(V/X, Y)$$
$$\forall Z \in \mathbf{Z} \quad (adjacentN(X, Z) \quad \wedge \quad adjacentN(Y, Z))$$

$$\vee \quad independentN(X, \mathbf{Z}, Y)$$

where $\wp(V/X, Y)$ denotes the power set of the variables in the domain minus X and Y. This inference rule has an empty step and generates the adjacency constraint once it becomes valid:

$$adjacent(X, Y).$$

3.2.1.3 Edge-Orientation Rules

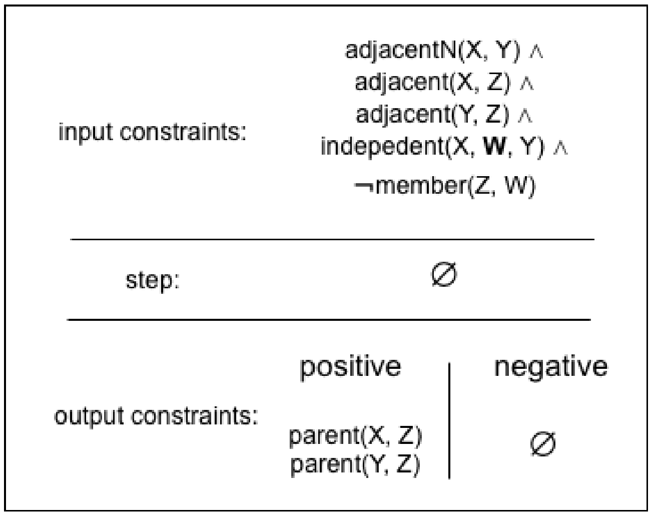
Constraint-based learning algorithms rely on rules such as collider-detection, known non-collider detection, and cycle avoidance to orient edges based on the learned independence constraints. These edge-orientation rules are derived from d-separation rules that identify the conditional independencies entailed by a graph [39].

Each edge-orientation rule can be translated into a separate inference rule since each edge orientation rule works in a manner consistent with the way an inference rule is executed. Their input conditions require certain adjacency constraints and conditional independence facts and these inference rules output parent constraints that identify the direction of the causal dependence between two variables. Since the edge-orientation inference rules only rely on constraints that may already exist in the causal knowledge base, they can be considered as simple rules that can be immediately executed once they are valid.

Collider Detection Rule (ColDetRule)

Collider detection identifies inverted forks between three variables [50]. The rule is as follows: given a triple of vertices X, Y, Z, and knowledge that X and Y are conditionally independent by a separating set $\mathbf{W}$ but that Z is not in this separating set, orient the edges as

$X \longrightarrow Z \longleftarrow Y$. If the edge were to be oriented in any other way, Z would be in the



input constraints:	adjacentN(X, Y) ∧ adjacent(X, Z) ∧ adjacent(Y, Z) ∧ indepedent(X, **W**, Y) ∧ ¬member(Z, W)	
step:	∅	
output constraints:	positive	negative
	parent(X, Z) parent(Y, Z)	∅

Figure 3.4. The Collider Detection Rule Formal Specification.

separating set. The collider detection inference rule, $ColDetRule(X, Z, Y)$ has as the input conditions:

$$adjacentN(X, Y) \wedge adjacent(X, Z) \wedge adjacent(Y, Z)$$
$$\wedge independent(X, \mathbf{W}, Y) \wedge \neg(Z \in \mathbf{W}).$$

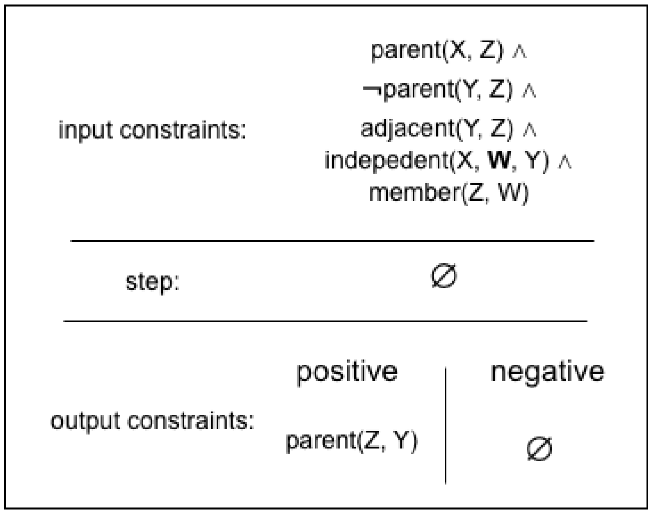
The step of this inference rule, as with all edge-orientation inference rules, is empty. Once the collider detection inference rule becomes valid, it can immediately be applied to generate the output constraints:

$$parent(X, Z) \wedge parent(Y, Z).$$

Known Non-Collider Rule (KNCRule)

The known non-collider rule identifies a chain of edges oriented in one direction between two variables X and Y [50].

Given a triple of vertices X, Y, Z, and knowledge that $X \longrightarrow Z$ and X and Y are conditionally independent by a separating set $\mathbf{W}$ and that Z is in this separating set, orient



input constraints:	parent(X, Z) ∧ ¬parent(Y, Z) ∧ adjacent(Y, Z) ∧ indepedent(X, **W**, Y) ∧ member(Z, W)	
step:	∅	
output constraints:	positive: parent(Z, Y)	negative: ∅

Figure 3.5. The Known Non-Collider Rule Formal Specification.

the edges as $X \longrightarrow Z \longrightarrow Y$. The known non-collider inference rule, $KNCRule(X, Z, Y)$ has as the input conditions:

$$independent(X, \mathbf{W}, Y) \wedge parent(X, Z) \wedge Z \in \mathbf{W}$$

and the output constraints:

$$parent(Z, Y).$$

Cycle Avoidance Rule (CARule)

Causal models are represented by directed acyclical graphs. The edge-orientation rules preserve acyclicity and include a cycle avoidance rule: given a pair of vertices X, Y, and knowledge that there is a directed path, P, $p_i, ..., p_j, ..., p_k$ from X to Y, orient the edge as $X \longrightarrow Y$ because the alternative orientation creates a causal cycle that violates the assumption of a directed acyclic graph [50].

The Cycle Avoidance inference rule, $CARule(X, Y)$ has as the input conditions:

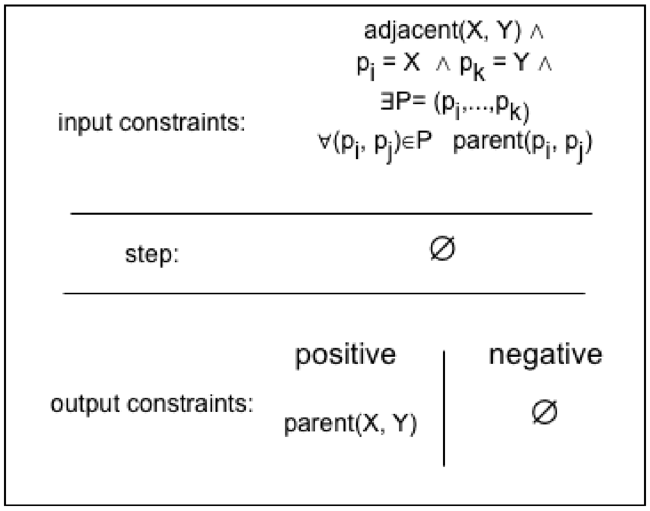


Figure 3.6. The Cycle Avoidance Rule Formal Specification.

$$adjacent(X,Y)$$
$$\wedge p_i = X \wedge p_k = Y$$
$$\wedge \exists P = (p_i, ...p_k)$$
$$\wedge \forall (p_i, p_j) \in P \quad parent(p_i, p_j)$$

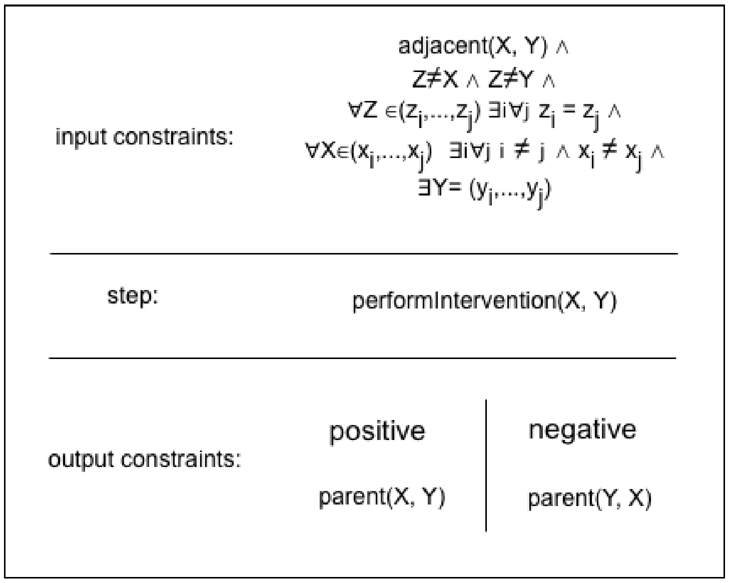
and the output constraints:

$$parent(X,Y).$$

3.2.2 Interventional Inference Rules

There are a very large number of interventional analyses available to infer a causal dependence. Many researchers often make use of the "perfect" interventional rule to facilitate theoretical reasoning of constraint-based algorithms [14].

The intervention represents a simple experimental design for which complete control over all variables in the domain is assumed. This kind of rule is referenced within the active learning of causal model community frequently [37, 13]. This type of intervention is often used during learning after a causal structure has been found but where causal direction is unknown on at least one edge (X, Y). The perfect intervention is then used to determine the orientation of the edge (X, Y) and then subsequently on any other non-oriented edges. The formal representation for the inference rule is given in Figure 3.7.



input constraints:

$\text{adjacent}(X, Y) \wedge$
$Z \neq X \wedge Z \neq Y \wedge$
$\forall Z \in (z_i, \ldots, z_j)\ \exists i \forall j\ z_i = z_j \wedge$
$\forall X \in (x_i, \ldots, x_j)\ \exists i \forall j\ i \neq j \wedge x_i \neq x_j \wedge$
$\exists Y = (y_i, \ldots, y_j)$

step: performIntervention(X, Y)

output constraints:

positive	negative
parent(X, Y)	parent(Y, X)

Figure 3.7. The Interventional Inference Rule Formal Specification.

A single treatment variable X is assigned a known setting, severing the causal influences from all parent variables, to observe the effect on a single outcome variable, Y. For simplicity, assume the intervention, $Int(X, Y)$, is only applied when X and Y are known to be adjacent in the causal model: $adjacent(X, Y)$. The step of this rule is an intervention where the value of X is assigned each possible value in its domain to observe the values on Y. A marginal independence test is applied on X and Y to determine if they are associated. If so a parent constraint is applied: $parent(X, Y)$. If not, then since X and Y are known to be associated, it must be that the causality runs in the opposite direction: $parent(Y, X)$.

The perfect interventional inference rule is provided as an example of both how an intervention or experimental design can be encoded and how they can be formally represented as a rule described in the literature. It is very useful to show how an existing activity to discover an inference can be represented as a rule that can be used by an algorithm. Beyond that it is not a practical inference rule because it makes some strong assumptions. First it assumes that all interventions are possible when in practice they rarely are. Second, it assumes that control of other variables is complete and that there are no latent variables to be concerned about.

3.2.3 Representing Inference Rules in General

A large number of actions taken to find inferences can be represented using this formalism. Randomized experiments are commonly used to control for potential common causes of both the treatment and outcome variables by using random assignment of the value of the treatment variable [7]. It can be represented similar to the "perfect" interventional inference rule with the addition of a few more predicates to identify *control* and *treatment*. The *interrupted time series* (ITS) inference rule is an observational rule that tests if a treatment variable, treated as an event, has any influence on an outcome variable across time. ITS can be represented formally when predicates are included for discussing time and aggregation across data.

New inference rules can also be accommodated with this representation. In some cases inference rules are refined and improved against weak assumptions made on the data. The strength of formally representing rules is such that new rules can be incorporated into learning as they become available rather than developing wholly new algorithms based around the new rule.

The graph shown in Figure 3.8 is called a Verma Graph and helps identify a constraint on the causal model when a latent variable L is present [54]. A new inference rule based on dormant independence is a special kind of Verma constraint where a conditional independence is implied based on identifying interventional distributions within the set of observational data [48].

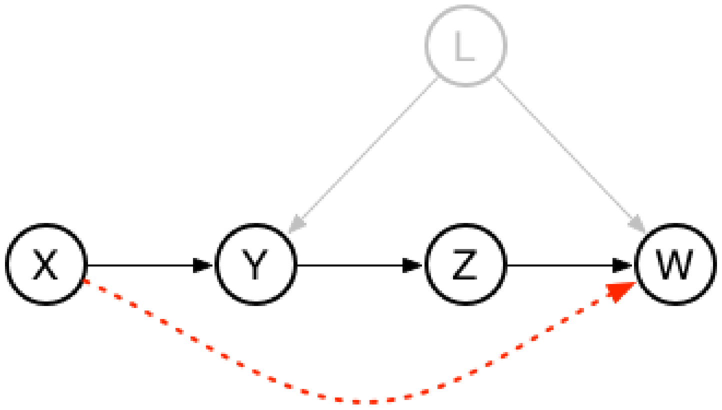


Figure 3.8. Graphical Structure Representing a Verma Constraint.

Most constraint-based algorithms assume propositional data [30]. But when the data is relational, new rules are available [32]. For example the PC algorithm was extended to incorporate the use of relational representations in a new technique, relational-PC (RPC) [33]. As with PC, RPC uses conditional independence tests but searches for variables across the relational representation. RPC constrains the potential space differently than does PC because it does not allow the existence of a relationship to be causal for variables on the entities involved in the relationship. Another new technique made possible by working on relational data representations is the relational blocking design where blocking refers to the method of conditioning based on entity. If the entity that is blocked on contains hidden or latent variables, then relational blocking can eliminate the entire class of variables as potential common causes [44].

3.3 Simplifying Assumptions

Consider the choice of conducting an independence test to be a separate action and include it as an inference rule that generates either an $independent(X, \mathbf{Z}, Y)$ or an $independentN(X, \mathbf{Z}, Y)$ constraint on the model space. For simplicity assume these actions are components of any inference rule that relies on them and would otherwise be able to execute save for the missing independence information. Formally, whenever an inference rule such as the conditional independence rule is considered, this would just be represented as a combination of an independence test followed by the action of the conditional independence rule. But from a practical perspective these steps can be combined into the single action of applying the conditional independence rule.

Also note that this work focuses on uncovering the structure of the causal model and does not address parameterization. Once the CPDAG representing a Markov-equivalence class of models is found by an algorithm, the probability distribution of a random variable given its parents or direct causes can be represented by a conditional probability distribution (CPD) [29]. There are many ways to represent this CPD including tables [24], linear equations, and decision trees [3].

CHAPTER 4
CAUSAL LEARNING AS A MARKOV DECISION PROCESS

Constraint-based causal learning is the process by which the space of potential causal models is pared down. The actions taken, that of applying inference rules, result in constraints that eliminate models inconsistent with the obtained knowledge. Alternatively, the sets of constraints that are learned identify a Markov-equivalence class of models that are most likely to explain the domain.

Because we can treat a set of constraints as a 1:1 mapping with set of models, inference rules can be viewed as atomic operators that allow a transition from one set of constraints to another. With this perspective, causal learning is a problem of choosing one action from a set of actions available on each set of constraints in order to uncover a set of causal models consistent with the domain. It can be described as a search across the state space of constraints resulting in a sequence of inference rules. If each rule has an associated cost, then inferring a causal model can be treated as an MDP.

The initial state of causal knowledge, the set of available inference rules, and the transitions between states based on application of those rules implicitly define the state space of the learning problem. A directed network or graph describes this state space where nodes are states of constraints and the edges between nodes are the transitions due to applying actions. A path in the state space is a collection of states connected by inferences. The goal in causal learning is the identification of efficient sequences of inference that find a state with the most consistent models for the domain from some initial state of constraints.

While the problem is assumed to be fully-observable, the outcomes from applying inference rules are probabilistic. The specification of the probabilistic nature of the actions is called the transition model $T(s, a, s')$ that describes the probability of reaching state s' from state s after applying inference rule, action a. The transition model is assumed to be Markovian where the probability of reaching s' from s depends only on s and the available actions and the probability does not depend on the history of earlier states of causal knowledge.

The causal learning MDP is specified as such:

State: s

A state for the causal learning MDP is defined as a tuple composed of (1) a set of constraints on the model space and (2), a set of inference rules available within this state.

State: {constraints}x{available inference rules} (a 2-dimensional, super-exponential, space)

The set of available inference rules is considered a part of the state to prevent algorithms from repeating the same inference rules which have no effect. Without maintaining this set, the transitions across the state space must include self-loops from one state back to itself. As self-loops indicate no new constraints have been found, they can be eliminated by removing from the set of available inference rules all rules which have already been applied.

Each state implicitly defines a set of causal models which need not be a Markov-equivalence class.

Initial State: s_0

The initial state is a set of initial constraints as described by any prior knowledge of the domain. The initial state has an empty inference sequence.

Actions: A

The set of inference rules in this domain and allowed by the system. For example the PC algorithm would only allow conditional independence rules, conditional dependence rules, and edge-orientation rules.

Cost: $C_a(s, s')$

The cost associated with applying inference rule $a \in A$ to reach state s' from state s. Costs are bounded negative values to encourage a policy to reach the goal state quickly because applying unnecessary rules accrues more negative costs than executing a minimum set. Costs can be measured by factors such as time and money.

Terminal states

Terminal states have no available inference rules and thus a terminal state cannot transition to another state. Terminal states can be considered *stuck states* where the algorithm can no longer proceed because it has exhausted the set of inference rules. A terminal state may represent the smallest set of Markov-equivalent models consistent with the domain. When terminal states do not represent a Markov-equivalence class, the set of available inference rules has been overly constrained.

Goal(s): {constraints}x{*} where * represents a wild card.

A goal is used to describe a desired set of constraints that helps guide an algorithm in the selection of its rules. When specifying a goal, there is no need to include a specification on the set of available inference rules left over. In many cases the set of available inference rules in a goal state is either empty or contains inference which would transition out of the goal state if they were applied. For these reasons a wild card is used to indicate the set of available rules can be ignored.

Goals need not specify the entire causal model be learned. For example a subgraph within the causal model might suffice. A goal could specify a desire to know something about a particular edge, all edges from a particular variable, all edges in the model, or something in-between. Example Goals:

$G1 = \{\text{all pairs}\}\text{x}\{*\}$.

$G1$ specifies that the entire set of constraints is desired, and consequently, to identify the smallest Markov-equivalence class most consistent with the domain. The use of the wild card simply means that the set of inference rules used to arrive at this goal is unimportant. $G1$ by definition is a terminal state.

$G2 = \{(X, Y)\}\text{x}\{*\}$.

$G2$ specifies that a constraint on a specific edge is desired. $G2$ is not necessarily a terminal state. Such a goal would be useful if one were interested in understanding a relationship between two variables such as "Is smoking a direct cause of cancer?"

$G3 = \{(X, \#)\}\text{x}\{*\}$.

$G3$ specifies that knowledge of all constraints involving X is desired. As with $G2$, $G3$ does not necessarily specify a terminal state. It could be described as "Why might someone smoke and if they did smoke, what does it directly cause?"

Within an algorithm, the goal for a specific constraint can be represented by using adjacency, non-adjacency, or parent constraints to describe the knowledge of interest. If only interested in the skeleton information on an edge (X, Y), that goal can be specified to contain either $adjacent(X, Y)$ or $adjacentN(X, Y)$. If orientation was also desired, then only the term $parent(X, Y) \vee parent(Y, X)$ need be specified.

Transition Model: $T(s, a, s') = p_{sas'}$

The transition model defines the probability $p_{sas'}$ of transition from a state s to a state s' if action a was taken. Each inference rule has one of three outcomes: True, False, and inconclusive. When a rule evaluates to True or False, it transitions to some state s'. Inconclusive outcomes are defined as self looping from state s back to state s. The transition model is also defined such that if action a was already taken, repeating a results in a self-loop back to state s, $T(s, a, s) = 1$.

For example, assume an initial state, s_0 in a 3-variable system, with the following available inference rules:

$$CIRule(X, [], Y),$$
$$CIRule(X, [], Z),$$
$$CIRule(Y, [], Z),$$
$$CIRule(X, [Z], Y),$$
$$CIRule(X, [Y], Z),$$
$$CIRule(Y, [X], Z)$$

and a prior over the constraints defined as:

$$p_{xy} = P(X \longrightarrow Y) : 0.8, \quad P(X \longleftarrow Y) : 0.1, \quad P(X \quad Y) : 0.1$$
$$p_{xz} = P(X \longrightarrow Z) : 0.7, \quad P(X \longleftarrow Z) : 0.0, \quad P(X \quad Z) : 0.3$$
$$p_{yz} = P(Y \longrightarrow Z) : 0.0, \quad P(Y \longleftarrow Z) : 0.2, \quad P(Y \quad Z) : 0.8$$

where the prior on each edge is independent of any other edge. Then this prior can be used to compute the transition of executing any action a:

$$T(s_0, CIRule(X, [], Y), s_1) = 0.1$$
$$T(s_0, CIRule(X, [], Y), s_2) = 0.9$$
$$T(s_0, CIRule(X, [], Z), s_3) = 0.3$$
$$T(s_0, CIRule(X, [], Z), s_4) = 0.7$$
$$\ldots$$

where:

s_1: $\{independent(X, [], Y), adjacentN(X, Y)\}$ x $\{CIRule(X, [], Y)\}$

s_2: $\{independentN(X, [], Y)\}$ x $\{CIRule(X, [], Y)\}$

s_3: $\{independent(X, [], Z), adjacentN(X, Z)\}$ x $\{CIRule(X, [], Z)\}$

s_4: $\{independentN(X, [], Z)\}$ x $\{CIRule(X, [], Z)\}$

Inference Rule Sequences

Once a policy is found for an MDP, it specifies a single action for every state. The set of actions from an initial state to some goal or terminal state form a *Markov chain.* In this dissertation, the Markov chain for a causal MDP identifies a sequence of inference rules that are referred to as an *inference rule sequence.*

Because the decision problem is sequential, the inference rule sequence can be described by a cost function expressing the efficiency of the solution. When the learning algorithm chooses an action, it incurs the cost of that inference rule, $C_a(s, s')$. The cost of the entire inference rule sequence then is the sum of the costs incurred as the algorithm searches for a goal state.

$$TotalCost = \sum_{s=s_0}^{G} C_a s, s'$$

The use of cost rather than reward for the MDP is a generalization of search problems. The algorithm's goal is to solve the learning problem as quickly as possible with a minimal amount of cost.

4.1 Causal Search and Finding Inference Rule Sequences with Causal Plans

In this section, we describe a class of algorithms that formulate the task of learning a causal model as an MDP and then solve for that MDP. We call this process *causal search.* Causal search explores the state space of a causal MDP as it develops a policy with a goal of finding an inference rule sequence that reaches a Markov-equivalence class of models that best fit a domain.

The state space of a causal MDP can be represented with an AND/OR graph. If self-loops from a state back to itself are ignored, then the state space can be represented as an AND/OR tree, a special case of the AND/OR graph. Figure 4.1 gives an example of a partial state space for a three variable domain. For the purposes of brevity the available inference rules are limited to marginal independence tests and only show some of the graph.

Actions are placed inside the OR nodes. Instead of showing the constraints, the AND nodes show a graphical representation of the ramifications of the constraints that are learned. Figure 4.2 explains how to decipher the state representation.

If it is known that an edge exists in the model between two variables X and Y, the edge is shown as a solid black line. This graphical structure corresponds to the $adjacent(X, Y)$ logical predicates used in the specification of the inference rules. Likewise, f a constraint is learned that X and Y are independent, $adjacentN(X, Y)$, the edge (X, Y) is removed from the graph. If nothing is known about an edge, a grey dashed line is used. This corresponds to the logical statement: $adjacent(X, Y) \vee adjacentN(X, Y)$.

Note that if an edge has a single arrow, then orientation is known. Directed edges correspond to $parent(X, Y)$ or $parent(Y, X)$ depending on the orientation. If an edge is bi-directional, then the constraints do not specify causal direction and either direction is possible.

Because an inference rule can evaluate to True or False and transition to one of two states, an algorithm must consider a graph of the state space reachable from the start state and have a policy that specifies which action to take in each state. This view of the state space is called a *causal plan.* The causal plan informs the algorithm where it is and the policy tells it which action to take as it follows the causal plan.

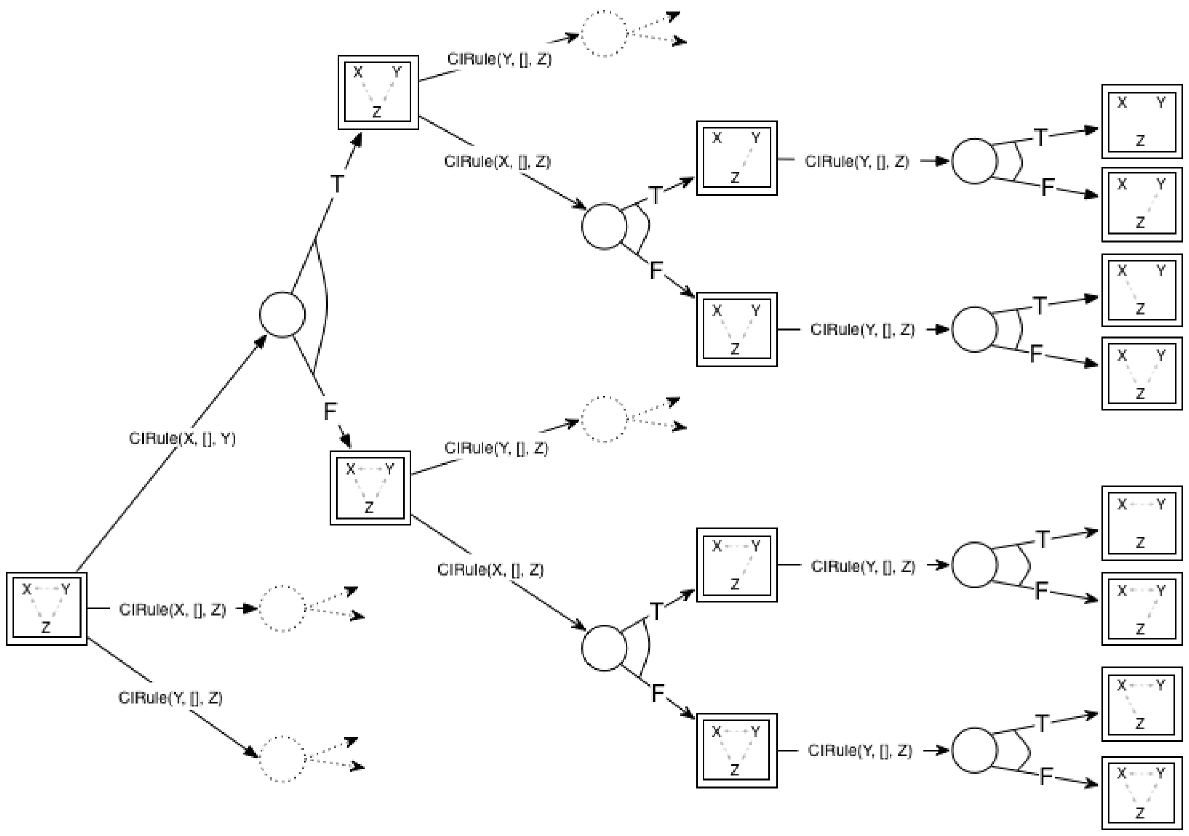


Figure 4.1. The State Space for a Simple 3-variable Domain.

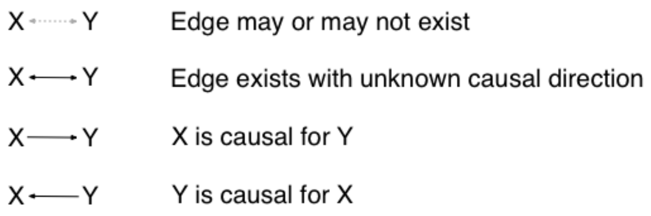

X Y Edge may or may not exist
X Y Edge exists with unknown causal direction
X Y X is causal for Y
X Y Y is causal for X

Figure 4.2. The Four Possible Edge States in the Graphical Representation of the State of Knowledge.

Once an action is taken by an algorithm, it no longer considers any state except those reachable from the current state. This means a policy need not be specified for the full

AND/OR tree but rather can be specified only for reachable nodes. In this way a causal policy discovers a *partial solution graph* represented by a *partial causal plan*.

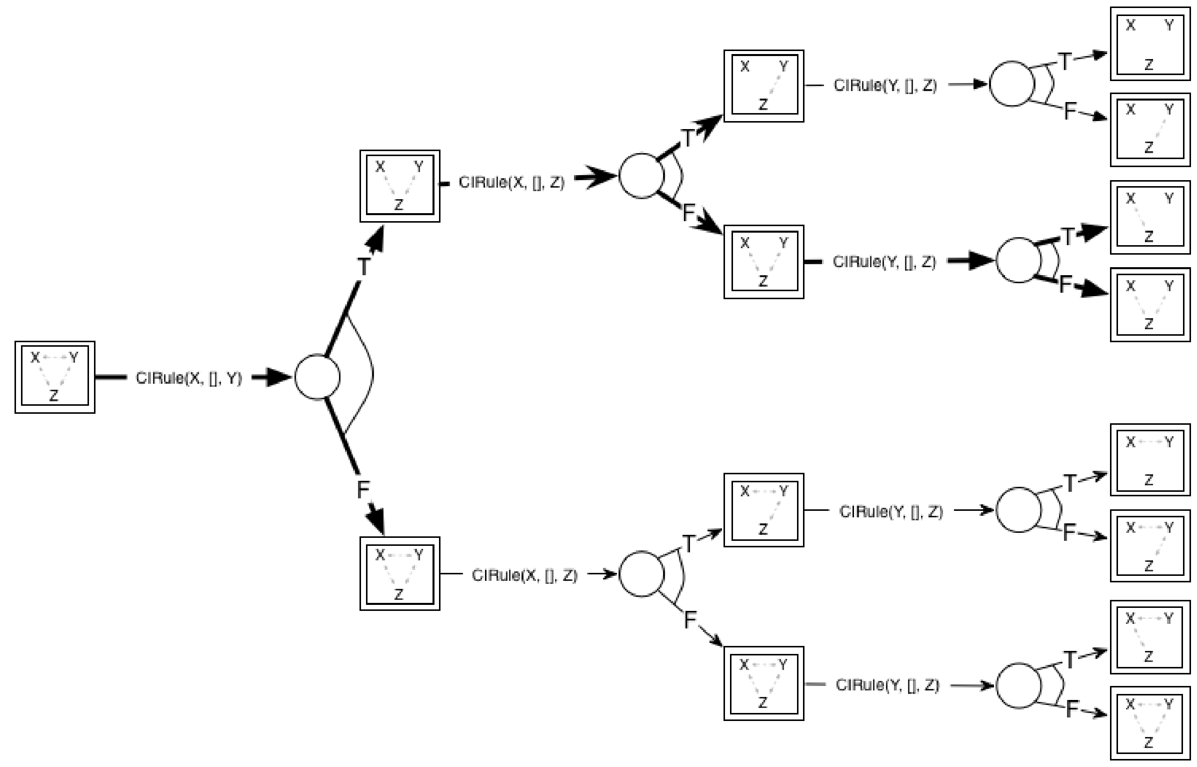


Figure 4.3. The Causal Plan for the Example 3-variable Domain.

Causal policies differ in the causal plans they identify. A causal plan for Figure 4.1 is shown in Figure 4.3.

Causal search finds the causal plan according to the causal policy and then executes the plan subsequently creating the *inference rule sequence*. Assuming that X and Y are marginally independent, that X and Z are not marginally independent, and that Y and Z are not marginally independent then the dark path in Figure 4.3 represents the solution to the causal plan found by the policy. On executing the plan and following the branches from the OR nodes, the learning algorithm would identify the inference rule sequence:

$$CIRule(X, [], Y), CIRule(X, [], Z), CIRule(Y, [], Z).$$

4.2 Policies to Solve the Causal MDP

An MDP is solved by finding a policy π that specifies which action to take in every state s. Well known methods to solving an MDP are the value-iteration and policy-iteration algorithms. Within the state space of causal learning, applying these solutions will have issues with computational intractability.

With even a small number of inference rules available, as the number of variables in the domain increases, the size of the causal plan can grow exponentially. For example, with the simple approach of the PC algorithm, the policies must consider a conditional independence test for every pair of variables and a power set of the remaining $V - 2$ variables as potential separating sets. With V variables, this represents $V(V-1)2^{(V-2)}$ conditional independence tests. In the worst case, the domain is a fully-connected causal model and every conditional independence test must be applied. This means that the causal plan reaches a minimum height of $V(V-1)2^{(V-2)}$ before it even includes the edge-orientation rules.

To address this problem, policies can be designed that choose one action followed by discovery of the next action. The high-level steps of a policy are:

1. Identify the set of available inference rules from a state.
2. For each inference rule, determine the reachable states for each possible outcome of the rule.
3. Rank the inference rules.
4. Choose the highest ranking inference rule and apply it.
5. Repeat steps 1-4 until a terminal state is reached.

Inference rule ranking provides the information used in the policy to make a decision for the next action. It is during ranking that the algorithm sorts the rules in decreasing order of benefit. Different algorithms will implement different methods to perform the ranking.

Uninformed search such as the PC algorithm have an implicit ranking embedded within their learning procedure. This kind of ranking criteria is unable to respond to the constraints learned and adjust the subsequent ranking of rules using this information. Informed search can rely on the transition probabilities across the state space to help accuracy in ranking. With improved ranking of inference rules, a policy can choose an action that tries to minimize the total cost of inference.

Based on how the PC algorithm works, the steps have been refined to include an additional activity that is called *inference rule preference.* This allows an algorithm to specify a subset of inference rules to focus on during the ranking stage. It aids in reducing computational complexity and provides a mechanism to allow the creation of a policy that implements the PC algorithm.

Preference allows for the algorithm to narrow down the set of rules it considers helping to reduce computational complexity. Establishing a preference can be as simple as identifying a hierarchy of inference rules based on some criteria such as cost. A policy need not utilize a preference and could consider all rules equally.

1. Identify the set of available inference rules from a state.

   Identify an ordering of rules by preference.

2. For each inference rule, determine the reachable states for each possible outcome of the rule.

3. Rank the inference rules.

4. Choose the highest ranking inference rule and apply it.

5. Repeat steps 1-4 until a terminal state is reached.

4.3 An Architecture to Execute Causal Search

An architecture was created to implement causal search. It uses a policy to create and then execute a causal plan. The architecture works directly on the causal plan and so it is indifferent to the underlying policy and how this policy is found. This allows the evaluation of different policies. The architecture executes the causal plan and follows the branches corresponding to the outcomes of the action until it executes the complete inference rule sequence.

Our architecture that performs causal search is composed of five core methods: GeneratePlan, FindNextInferenceRule, ChooseOneInferenceRule, RuleIsValid, and ExecutePlan. The pseudocode for each method follows.

4.3.1 GeneratePlan

The GeneratePlan method is an entry point to identifying the causal plan. A maxDepth can be specified to control how deep the plan is before it is returned. This way a partial plan can be generated even if the goalState hasn't been found yet. The method FindAllRules is not included in the pseudocode as this method is specific to each policy. Given a ruleList which can constrain the rules that are generated, FindAllRules will return the set of inference rules available to the search algorithm for planning.

Empty startStates and goalStates are handled by assuming a default value that for this method is a startState of $s_0 = \emptyset$ meaning that nothing is known about the domain and a goalState $G = \{\text{all pairs}\}$. After setting up the parameters, the method FindNextInferenceRule is called. This method will recurse from the startState to build up the plan.

Algorithm 1 GeneratePlan(ruleList, variables, prior, startState, goal, maxDepth)

```
rules ← FindAllRules(variables, ruleList)
if startState == NULL then
    startState ← $\emptyset$
end if
if goal == NULL then
    goal ← ALL PAIRS
end if
plan ← FindNextInferenceRule(rules, startState, goal, variables, prior, 0, maxDepth)
return plan
```

4.3.2 FindNextInferenceRule

The core of the developing the causal plan is the method FindNextInferenceRule.

Starting from an initial state, FindNextInferenceRule will maintain a set of inference rules that are not invalid. This set is composed of *goodRules*, which are inference rules that can be immediately applied, and *potentiallyGoodRules* that are not currently valid but may later become valid such as edge orientation rules waiting for the conditions in which they apply.

From among the set of *goodRules* ChooseOneInferenceRule identifies one rule for the next step in the causal plan. The state composed of known constraints with corresponding **true** or **false** values is passed into ChooseOneInferenceRuleṄote that *goodRules* may be an empty set which occurs when the set of valid inference rules have been exhausted. This can happen if a goalState is found but it can also happen when no other inference rules are available or all of those remaining inferences are invalid.

Once an inference rule is chosen, the method will recurse by identifying the next level in the plan by following the True and the False branches from the rule. Situations where the inference rule is valid but are unable to be applied are not considered here for brevity. The FindNextInferenceRule searches to a depth specified by maxDepth or until a

Algorithm 2 FINDNEXTINFERENCERULE(rules, state, goal, variables, prior, depth, maxDepth)

```
if state = goal or depth > maxDepth then
  return None
end if
plan ← ∅
goodRules ← ∅
potentiallyGoodRules ← ∅
for r ∈ rules do
  (valid, invalid) ← RULEISVALID(r, state)
  if valid then
    goodRules ← goodRules ∪ rule
  else if ¬ invalid then
    potentiallyGoodRules ← potentiallyGoodRules ∪ rule
  end if
end for
if |goodRules| = 0 then
  return None
end if
r ← CHOOSEONEINFERENCERULE(goodRules, state, variables, prior)
plan.rule() ← r
goodRules ← goodRules ∩ r
for truth ∈ [True, False] do
  newState ← state ∪ getOutputConstraints(r, truth)
  subplan ← FINDNEXTINFERENCERULE(goodRules ∪ potentiallyGoodRules, newState,
  goal, variables, prior, depth+1)
  if subplan then
    plan.branch(truth) ← subplan
  end if
end for
return plan
```

goalState has been found. By traversing the True/False branches that result from choosing an inference, the causal plan that is developed is the partial solution to the AND/OR tree.

4.3.3 ChooseOneInferenceRule

ChooseOneInferenceRule implements the policy. As mentioned before, the implementation of a causal search policy is composed of two parts: inference rule preference and inference rule ranking. Preference allows for the policy to narrow down the set of inference rules it considers. Within the preference set, ranking distinguishes the rules in decreasing order of benefit. ChooseOneInferenceRule identifies the preference, performs the ranking, and then chooses the highest ranking rule from the most preferred set of rules.

Algorithm 3 ChooseOneInferenceRule(rules, state, variables, prior)

```
highestPreference ← 0
preferenceSet ← ∅
for r ∈ rules do
   preference ← preference(r, —variables—)
   if preference > highestPreference then
      highestPreference ← preference
      preferenceSet ← r
   else if preference == highestPreference then
      preferenceSet ← preferenceSet ∪ r
   end if
end for
rankedSet ← rank(preferenceSet, prior, state, variables)
rule ← rankedSet[0]
return rule
```

The method $preference(r)$ allows the policy to identify if one type of inference rule is preferred over another. One method to implementing the preference is as a rule-based decision that has hard-coded a preference value for each inference rule in the system. Information about the domain such as domain size can be passed into the preference method to help determine maximum preferences. ChooseOneInferenceRule maintains the highestPreference value it has seen and the set of inferences that have that preference value.

After identifying the set of highest preferred inference rules, ChooseOneInferenceRule calls the *rank* method to order the set according to the benefit of each inference rule. Each policy has its own ranking method. If rules have the same ranking, the order within this subset of will be random. ChooseOneInferenceRule returns the first element of the rankedSet which is the highest ranking, highest preferred inference rule.

By decoupling the architecture from the policies, ChooseOneInferenceRule allows for any policy implementation because it simply calls the *preference* and *rank* methods. This means that the architecture is not connected to the policy and different policies can be developed and used to identify causal plans without having to change the design or implementation of the system.

4.3.4 RuleIsValid

The RuleIsValid method performs a check against the knowledge base to determine if the constraints for an inference rule are all **true**.

Algorithm 4 RuleIsValid(rule, state)

```
constraints ← getConstraints(rule)
notMissing ← true
for c ∈ constraints do
  if ¬ constraintCheck(c, state) then
    if missingConstraint(c, state) then
      notMissing ← false
    else
      return (false, true)
    end if
  end if
end for
return (notMissing, false)
```

getConstraints gets the set of input constraints associated with the inference rule. The method constraintCheck verifies if each constraint is found in the knowledge base represented here as the *state*. If it isn't, then a check is made to see if the constraint is simply missing as opposed to being **false**. If the constraint is **false**, the method returns that the rule is invalid. If the constraint is simply missing, the remaining constraints are checked in case one of those may be **false**. If all constraints are **true** and none are missing in the *state* then the inference rule is valid. If no constraint is **false** but at least one is missing, the inference rule is still not valid but it isn't invalid.

By distinguishing the reasons that an inference rule is invalid, RuleIsValid can return a tuple (valid, invalid). The three possible outcomes are that an inference rule is valid and not invalid (**true**, **false**), an inference rule is not valid but also not invalid (**false**, **false**), and an inference rule is not valid and invalid (**false**, **true**). The second case allows the causal planner to keep inference rules that are not valid because of missing constraints rather than being **false**. If the planner were to remove inference rules that were missing constraints, it would lose the ability to apply rules such as edge orientation and conditional dependence rules.

4.3.5 ExecutePlan

The ExecutePlan method takes the causal plan and any data that is already available to the system and proceeds through the causal plan following the branches of the plan based on the outcomes of the applied rule.

Algorithm 5 ExecutePlan(plan, data)

```
constraints ← ∅
ruleSequence ← ∅
while plan do
    rule ← plan.rule()
    (truth, newConstraints) ← rule.execute(data)
    constraints ← constraints ∪ newConstraints
    ruleSequence ← ruleSequence ∪ rule
    plan ← plan.branch(truth)
end while
return constraints, ruleSequence
```

Since the causal plan is an AND/OR tree, as the inference rule that is selected at each OR node is executed, the output constraints are added back into the knowledge base. The sequence of inference rules are updated and the path through the AND node is followed based on the truth of the inference. The execution continues until no further steps are left in the plan and ExecutePlan returns both the inference rule sequences and the constraints that define the causal models that remain in the model space.

CHAPTER 5

UNINFORMED POLICIES FOR SOLVING THE CAUSAL MDP

There are two kinds of policies to solving the causal MDP: uninformed policies and informed policies. The designation is based on the amount of information utilized by the individual policies. Prior to this work, all existing causal learning algorithms fell into the uninformed policy category. This includes the well known PC algorithm. These algorithms approach the causal MDP with a fixed transition probability across the state space.

5.1 The PC Policy

The PC policy implements the behavior of the PC algorithm but as a policy. The output of the PC policy is the PC causal plan. When the causal plan is executed, it generates the same set of inference rule sequences that the traditional PC algorithm would identify.

The high-level steps for the PC policy are:

1. Identify the set of available inference rules from a state.

   Prefer inferences in the following order:

   a. conditional independence tests in order of increasing separating set size,

   b. conditional dependence tests, and

   c. edge-orientation rules.

2. For each rule in the highest available preference set, determine the reachable states for each possible outcome of the inference.

3. Rank the set of highest preferred rules randomly.

4. Choose the highest ranking inference rule and apply it.

5. Repeat steps 1-4 until a terminal state is reached.

The PC policy relies on a small set of inference rules: conditional independence tests and edge-orientation rules. It also implicitly relies on the conditional dependence inference rule because the PC algorithm assumes that after exhausting the space of conditional independence tests, if a pair of variables (X, Y) are not found to be conditionally independent, they must in fact be dependent. Policies in general need not be so constrained, but to maintain consistency with the PC algorithm the limited set of rules is maintained. As with all causal search policies, the core of the PC policy is defined by its preference and its ranking methods.

In essence the PC algorithm is a bottom-up algorithm that seeks to exhaust the set of marginal independence tests before it considers conditional independence. Then it tries every valid conditional independence test of separating size one before separating size two and so on. These are followed by applying the set of conditional dependence and finally PC will apply the set of edge-orientation rules.

To mimic the PC algorithm, the PC policy places higher preference on the CIRules in order of their separating set size, followed by the conditional dependence tests, and then the edge-orientation rules.

The preference on conditional independence is defined as the maximum preference minus the size of the separating set. The $domainSize$ is passed into this method as the number of variables in the domain to identify a maximum preference. With this calculation marginal independence, which have the form $CIRule(X, \emptyset, Y)$, have the highest preference, $maxPreference$ since $|\emptyset| = 0$. The separating set size can be no larger than $|V| - 2$ thus conditional independence rules with the largest separating set have a preference of two, being preferred above conditional dependence.

Algorithm 6 PCPolicyPreference(rule, domainSize)

```
maxPreference ← domainSize
preference ← -1
for r ∈ rules do
  if type(r) == ColliderDetectionRule then
    preference ← 0
  else if type(r) == KnownNonColliderRule then
    preference ← 0
  else if type(r) == CycleAvoidanceRule then
    preference ← 0
  else if type(r) == CIRule($x$, $Z$, $y$) then
    preference ← $maxPreference - |Z|$
  else if type(r) == CDRule($x$, $y$) then
    preference ← 1
  end if
end for
return preference
```

Once all conditional independence tests on a pair of variables (X, Y) have been applied, the conditional dependence rule becomes valid. By default the CDRule will receive a preference score of one meaning it has a higher preference than the edge orientation rules. Applying the conditional dependence rules establishes the skeleton of the causal model.

After application of the conditional dependence rules, the edge-orientation rules have the next highest preference. This is captured by giving them a preference of zero. Within the set of edge orientation rules, all rules have the same preference meaning they can be applied in any order.

The PC policy is an uninformed policy because it never updates the transition probabilities of its available actions, so it never updates its inference rule ranking. Instead it assumes a fixed uniform probability across the outcomes of rules.

As an example, consider the scenario in Figure 5.1. States represented by s_2, s_4, s_6, and s_8 are not terminal states while states s_1, s_3, s_5, and s_7 are terminal states. Also remember that terminal states are desirable because they represent the policy has found a class of

Markov-equivalent models consistent with a domain. For this example, the model most consistent with the domain can be found in the Markov-equivalence class in state s_3.

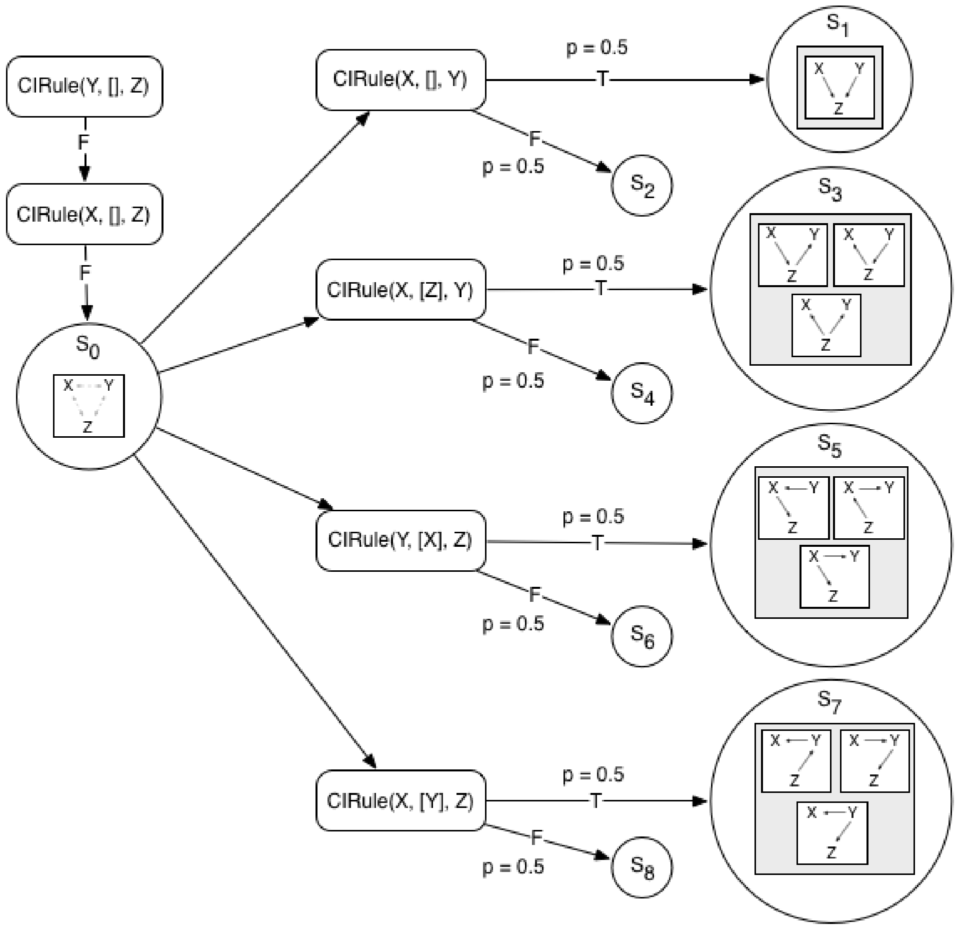


Figure 5.1. An Example of the PC Policy Computing Transition Probabilities of Available Actions.

Assume that the PC policy has already applied two rules: $[CIRule(Y, [], Z), CIRule(X, [], Z)]$ in that order and arrived at a state s_0. In state s_0, the PC policy has four conditional independence rules available: $CIRule(X, [], Y)$, $CIRule(X, [Z], Y)$, $CIRule(Y, [X], Z)$, and $CIRule(X, [Y], Z)$.

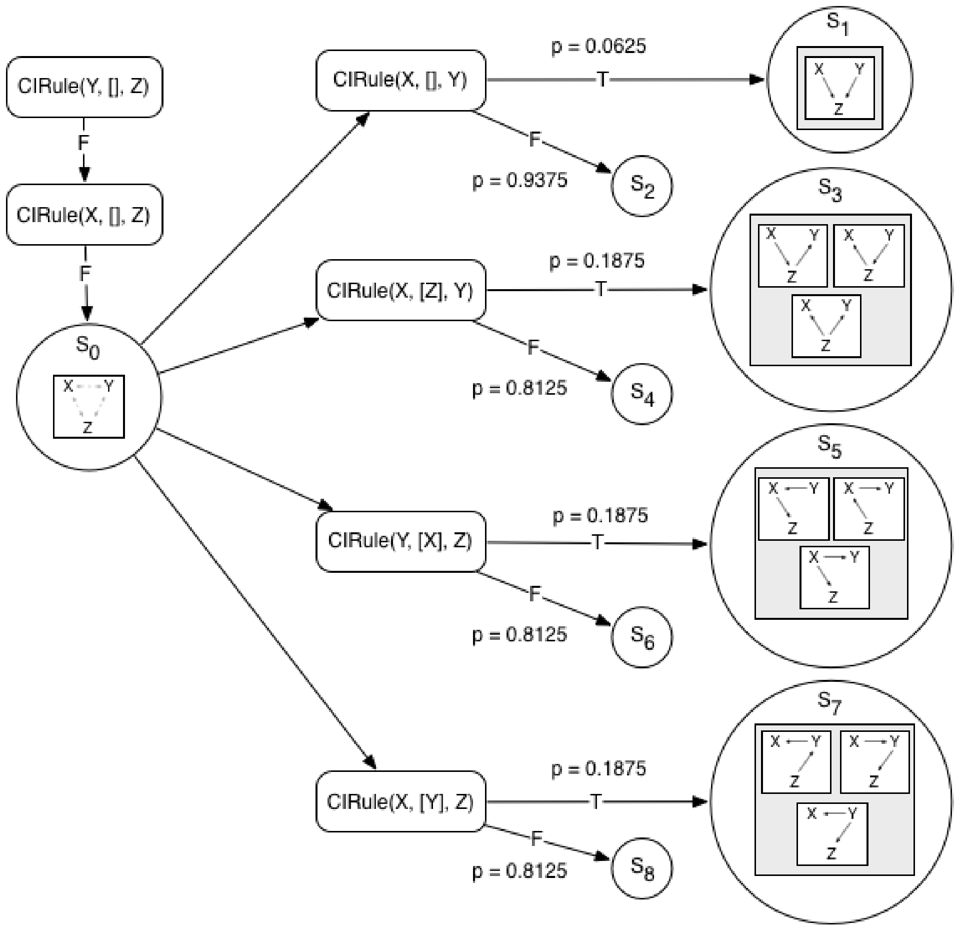


Figure 5.2. How the PC Policy Should Compute the Transition Probabilities of Available Actions.

Based purely on the preference criteria, the PC policy would choose rule $CIRule(X, [], Y)$ above the others. But PC does not know in advance whether application of $CIRule(X, [], Y)$ will result in reaching state s_1, a terminal state, or state s_2 a non-terminal state. For PC, the possibility of reaching either state is uniform. The uniform probability is consistent for every action PC might consider from state s_0 as can be seen when following the actions of $CIRule(X, [Z], Y)$, $CIRule(Y, [X], Z)$, or $CIRule(X, [Y], Z)$.

If the PC policy were more informed, it might see the transition probabilities as shown in Figure 5.2. While the preference criteria would have the PC policy still choose rule $CIRule(X, [], Y)$, it would at least know it would be more likely to end up in state s_2 a non-terminal state than it would in state s_1, a terminal state.

The implementation of the uninformed PC policy is to rank rules within a preference set equally and thus to choose one rule from within this set uniformly at random.

Algorithm 7 PCPolicyRanking(preferenceSet, prior, state, variables)

```
  rankedSet ← ∅
  rankedSet ← randomShuffle(preferenceSet)
  return rankedSet
```

The PC algorithm does use an ordered set of variables to test for adjacencies. For example if an edge from variable X to variable Y is tested marginally and a subsequent conditional test is applied, the edge (X, Y) will be tested and not (Y, X). This order can be used within the ranking method to maintain a consistent order within the separating sets. For a domain a lexicographical naming convention on the variables, an ordering over the names can be used to maintain the consistency with PC.

5.2 Examples of PC Causal Plans

Figures 5.3-5.5 demonstrate a causal plan generated by the PC policy for a 3-variable domain. In this example the goal is to learn as much about the causal model as possible. This is represented in the example as a causal model over 3-variables with grey-dashed bi-directional edges using the same notation from Figure 4.2. However in this causal plan the fringe nodes represent Markov-equivalence classes and those states are expanded to show each model that could occur in that state rather than use the graphical representation.

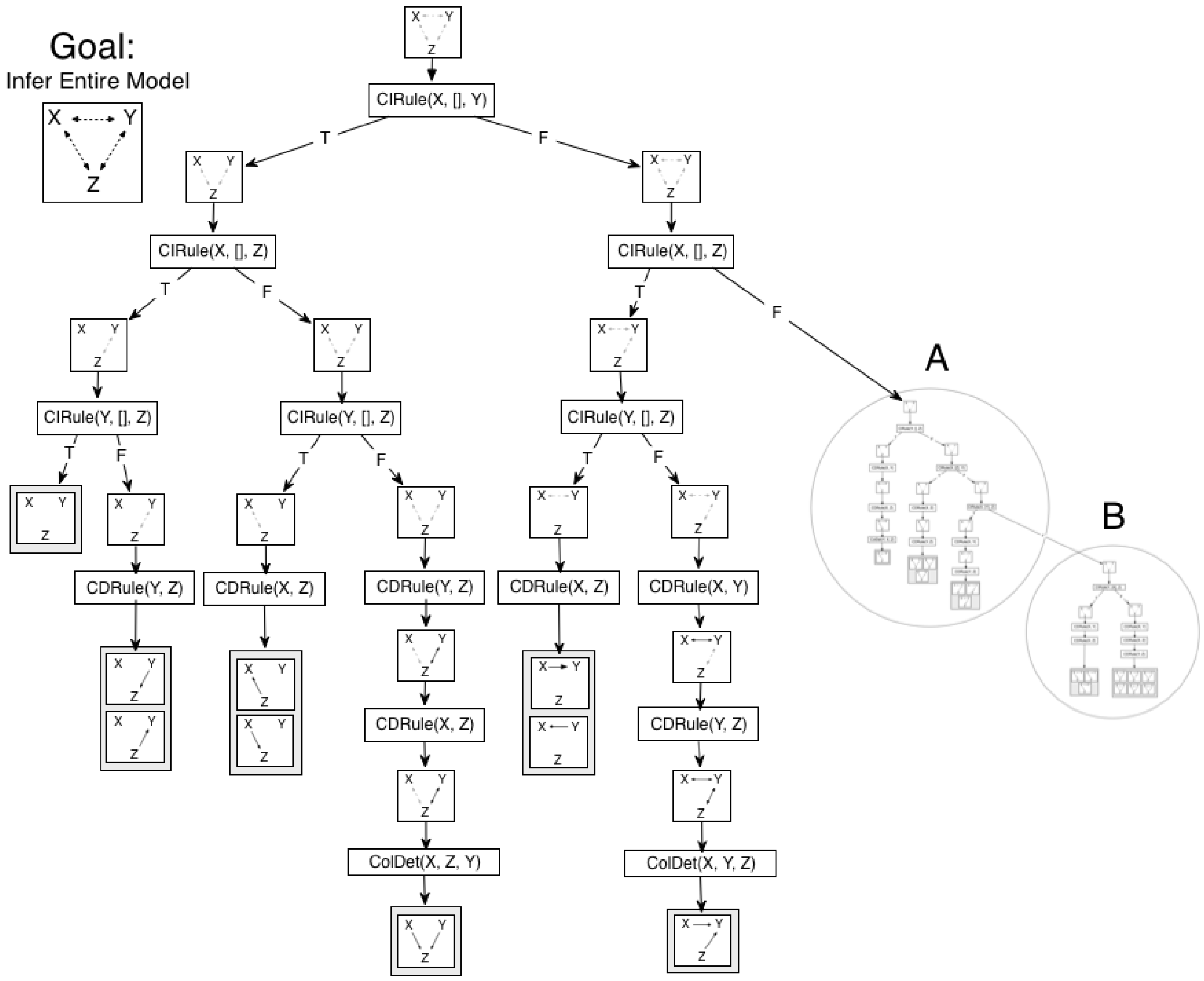


Figure 5.3. PC Causal Plan for 3-var Domain Example

The causal plan starts at the root node and continues through the AND/OR tree, applying the selected rules and traversing down the branch that corresponds to the outcome of the rule. The execution path through the causal plan produces the inference rule sequence.

As done in Figure 4.1 only the choice that is ultimately selected by the policy is shown instead of showing every potential rule at the AND nodes. Without this simplification, multiple figures would be required to show the complete causal plan. Even in a 3-variable

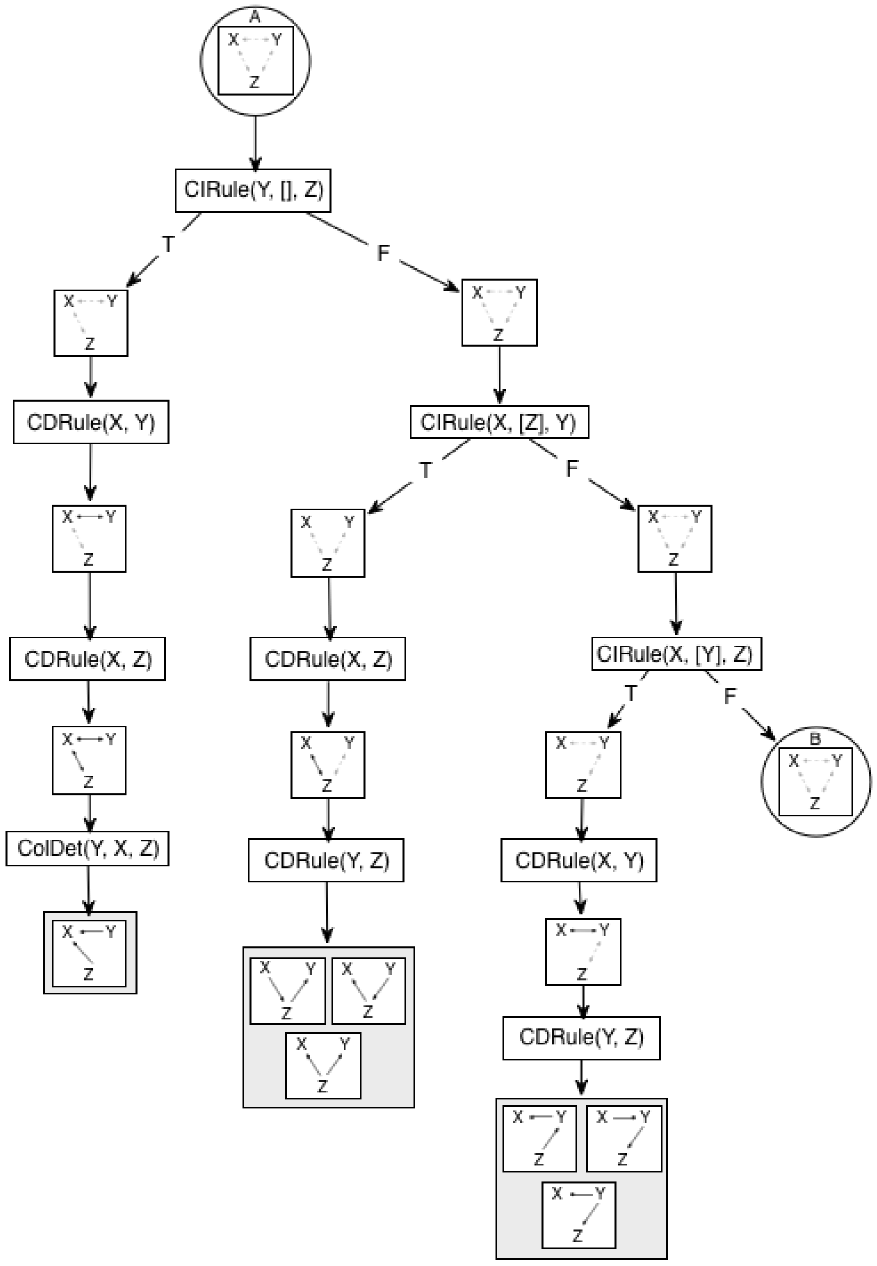


Figure 5.4. PC Causal Plan for 3-var Domain Example, Part 2 of 3.

domain with only conditional independence and conditional dependence rules, the root node

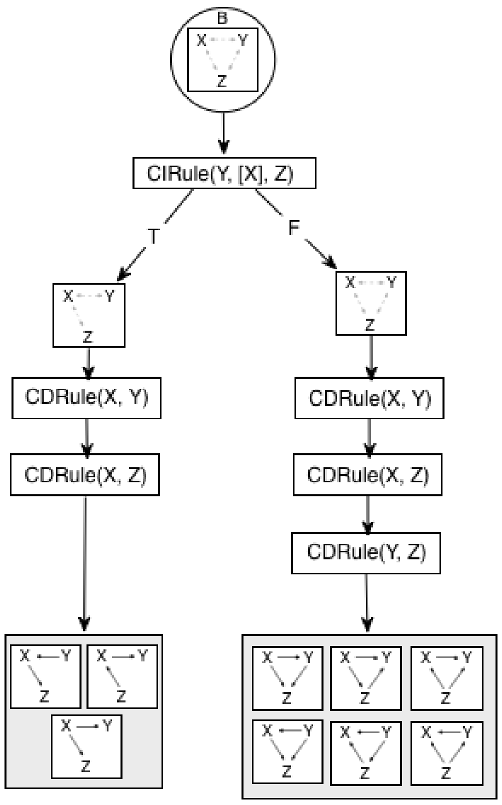


Figure 5.5. PC Causal Plan for 3-var Domain Example, Part 3 of 3.

would have nine different rule branches. We've also simplified the OR nodes by dropping the rounded nodes and just using the rule.

For the simple example it is clear that depth of the causal plan can become a problem, especially in terms of computational memory and processing. This reinforces the decision that a one-step iterative approach of find the next rule, apply that rule, and then repeat saves on computational requirements. The problem of computation grows worse as more rules are available and as the domain size increases.

In the 3-variable example only one edge-orientation rule applies: the collider detection rule. The known non-collider detection doesn't gain enough information from conditional independence alone to ever apply in a 3-variable domain while the cycle avoidance will only apply if causal orientation is known on two of the three edges and a third edge is present.

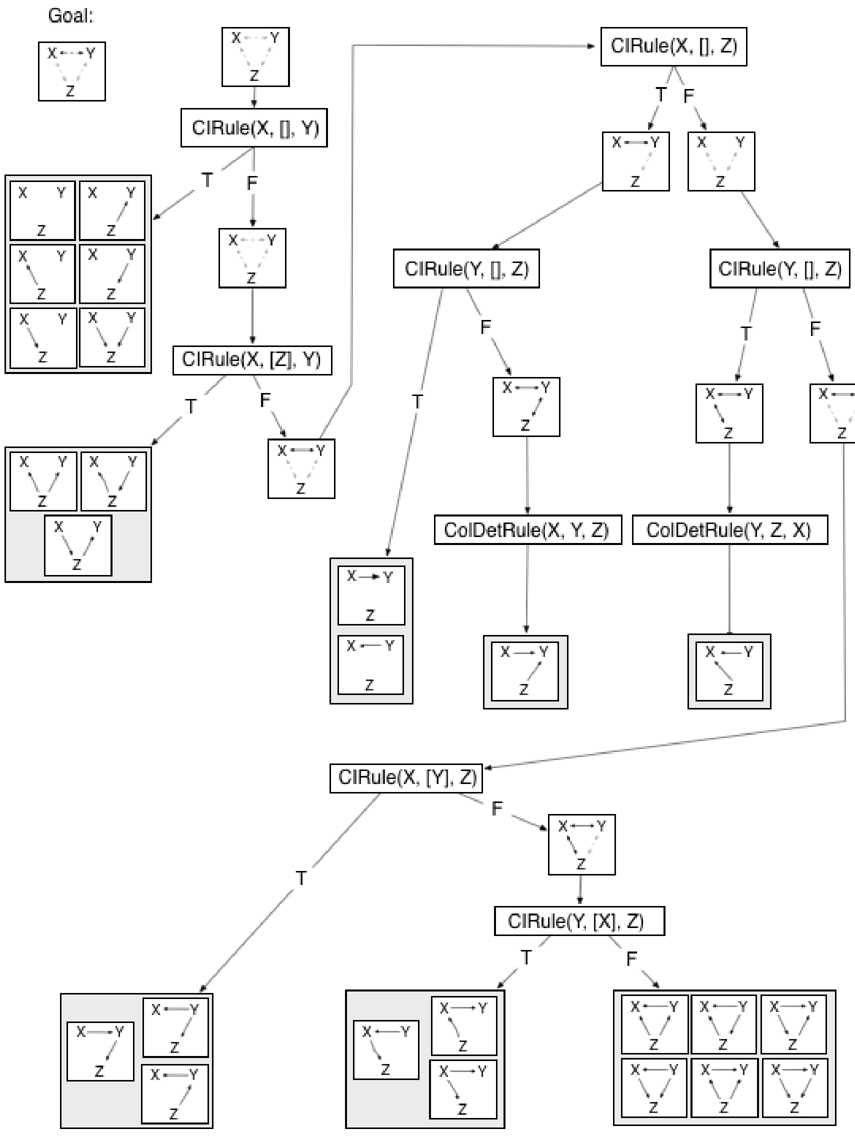


Figure 5.6. PC Causal Plan for 3-var Domain With a Specific Goal.

To demonstrate the PC causal plan, Figure 5.6 shows the PC policy output when given a specific goal. In this case, the goal is to learn as much as it can about a certain edge, (X, Y).

This goal directs the policy to applying rules with preference to rules that will achieve the goal. And since the PC policy doesn't necessarily care about information on any other edge, once it has determined everything it can on edge (X, Y), it stops applying inference rules.

5.3 Other Examples of Uninformed Policies

We've chosen the PC policy as an example of an uninformed policy because it is has been widely studied, extended, and continues to be a basis for comparisons in the causal learning community. PC has been shown to be sound and complete [50]. And extensions of PC focus on reducing potential orientation errors that can arise from failures to satisfy some of the causal assumptions such as causal sufficiency and causal faithfulness. For those extensions, skeleton learning is the same but the criteria to orient an edge differs leading to alternative edge orientation rules.

5.3.1 The Random Policy

The simplest and most-naive uninformed policy is the Random policy that randomly applies rules. The Random policy is much worse in terms of execution time than the PC policy. If Random search can exhaustively apply all inference rules, then the random policy will eventually uncover the same model that the PC policy will, but it will do so with a much more inefficient inference rule sequence.

The Random policy relies on no preference and no ranking.

Algorithm 8 RandomPolicyPreference(rule, domainSize)

```
  return 1
```

Algorithm 9 RandomPolicyRanking(preferenceSet, prior, state, variables)

```
  rankedSet ← ∅
  rankedSet ← randomShuffle(preferenceSet)
  return rankedSet
```

The consequence of this is that all inference rules are applicable as soon as they become valid. However this does not necessarily mean an inference rule has a meaningful benefit. For example all conditional independence tests are immediately valid at the start of learning. This means the Random policy is free to choose any $CIRule$ and so conditional independence tests across the same variable pair $(X,\ Y)$, can all be chosen as the first rule. A successful application of a $CIRule(X, \mathbf{Z}, Y)$ where $\mathbf{Z}$ is the set of all other variables, $V \setminus X, Y$, does not mean that any other $CIRule$ would not also be successful. In fact, application of that rule cannot rule out marginal independence or independence due to any subset of $Z \in \mathbf{Z}$.

If the Random policy were to use the same set of available rules as does the PC policy, it would be forced to apply every conditional independence rule. It must do so or run the risk of missing the minimal separating set for two variables. While it can certainly identify the causal model skeleton, and perhaps in a smaller inference rule sequence than the PC policy, it cannot guarantee that it identifies the proper orientation of the edges in the model. This puts it at a distinct disadvantage to the PC policy.

We could eliminate the difficulty that the Random search strategy experiences by introducing a new causal inference rule based upon algorithms that identify a minimal separating set from a non-minimal separating set [52]. This new inference rule could have its input constraints specified using paths and a combination of algorithm 3 found in [52]. But that changes the set of inference rules beyond those considered by PC and inhibits fair comparison, so for this reason, the Random policy is not considered further.

5.3.2 The SGS Policy

A precursor to the PC algorithm is the SGS algorithm that utilizes the same rules as does PC but does not order the conditional independence tests by separating set size [50]. Instead SGS applies every possible conditional independence test for every pair of variables and never makes use of pruning unnecessary tests. This leads to the worst-case performance of the PC algorithm and its easy to see why SGS would not do very well on sparse causal models. It is also easy to see that the Random policy is essentially the SGS policy but less principled.

5.3.3 Pearl's Independence Graph Policy

Pearl introduced a learning algorithm based on finding the independence graph first as an alternative of the SGS algorithm [50]. Just as PC does, this algorithm initially assumes a fully-connected graph and chooses as its first inference rule to test the (in)dependence of variables (X, Y) conditioned on all other variables $V \backslash X, Y$. Note that this is similar to the random policy except that it begins with the largest conditioning set. If X and Y remain dependent conditioned on all other variables, an undirected edge connects them. The resulting skeleton is referred to as the independence graph.

X and Y may still be independent, conditioned on some subset of the set of variables adjacent to them in the independence graph. In the causal DAG being inferred, the parents of any variable form a maximal complete subgraph referred to as a clique. X and Y can be determined to be independent given a subset of a clique including either X or Y.

The algorithm must first find the cliques in the graph and then try to condition on every possible subset to determine dependence. Identification of the cliques in the graph and then conditioning on all members of the maximal clique, leads to worse performance than the PC algorithm. This is because the PC algorithm can identify the same (in)dependencies without the need to find cliques and because PC only applies conditional independence tests up to the minimal conditioning set.

CHAPTER 6

OPTIMAL CAUSAL SEARCH AND OPTIMALITY OF THE PC POLICY

Policies which are optimal will infer a causal model in a manner such that no other policy can perform better. The optimality of a policy can be assessed by evaluating its efficiency. Unfortunately, in practice only nature knows the optimal path solution to get from an initial state to a goal state. Instead a method comprised of a causal model oracle and a theoretical policy can be used to evaluate the efficiency of the inference rule sequences generated by the different policies.

6.1 Causal Model Oracle

Real domains provide no clear way of knowing the correctness nor completeness of any learning algorithm. One solution is to use a causal model oracle able to construct a causal model across a domain for evaluation. The oracle also provides the capability of applying rules to this model by responding to queries from a learning algorithm and responding with true/false outcomes for each rule.

With the oracle the different policies can be tested against one another and execute their plans to generate inference rule sequences. Note that for this work the oracle is not used to evaluate policy performance under conditions such as violations to causal sufficiency or faithfulness. This can be incorporated into the oracle if the model is specified to contain latent variables or to even use the selected causal model to generate synthetic data and then apply the inference rules to the synthetic data instead. Instead performance is evaluated under ideal conditions where violations to causal assumptions do not occur.

The oracle is able to randomly generate a causal model over any sized domain. For small domains of three to five variables, the oracle can generate every distinct causal model. For larger domains with a potentially super-exponential number of models, the oracle is specified to generate only a subset of the allowable set of causal models. The subsets for each domain size are fixed ahead of evaluation and re-used when testing each policy.

The concept of the causal model oracle is intuitive. The goal is to find the underlying causal model of a domain by asking questions in the forms of inference rules about the structure of the model. Since the oracle can read the causal model, it has the ability to answer each rule inquiry with the answer as if the rule had been applied to the actual causal model under evaluation. This includes identification of causal direction, if any, corresponding to interventional analysis. For example, one can think of a conditional independence rule $CIRule(X, \mathbf{Z}, Y)$ as a question: "Are X and Y d-separated given separating set $\mathbf{Z}$?"

6.2 The Optimal Policy

There could be many different criteria that one would use to define an optimal inference rule sequence. This work determines optimality as the length of the inference rule sequence. This is primarily because the PC policy relies on a small set of observational analysis that is relatively cheap computationally and monetarily. Efficiency for PC then is the number of inference rules it must use.

The true cost for the PC policy is the total time it takes to apply rules within a domain. Note that many other kinds of inference rules can be included such as interventional analysis, quasi-experimental designs, and observational analysis. But the inclusion of these rules would imply a much more difficult causal learning task than the one PC is typically applied to.

The optimal policy is the policy that produces the lowest expected cost given the initial state and the desired learning goal. The *Optimal Policy* can be implemented by directly relying on the causal model chosen by the oracle. The oracle provides the means to do this and thus the optimal policy can determine the most efficient rules to achieve a desired goal. This is a remarkable advantage for optimal policy and it can determine the exact benefit of each inference rule. It is essentially cheating.

Figure 6.1 is an example of the optimal inference rule sequence on 3-variables. The optimal sequence only contains those CIRules necessary to establish the smallest distinguishable Markov-equivalence class. In this example plan, the models that remain after each CIRule application are shown. To prove the Markov-equivalence class for the model, the Optimal policy only requires three CIRules. There is no other sequence consisting solely of CIRules that could arrive at the Markov-equivalence class with fewer inferences.

The Optimal policy can be thought of as being able to rank the outcomes of rules exactly. For the same example model shown in Figure 5.1, the Optimal policy would compute the transition probabilities as shown in Figure 6.2. It can thus determine that the shortest inference rule sequence from state s_0 would be to apply rule $CIRule(X, [Z], Y)$ as it is the only action with certainty to reach a terminal state.

6.2.1 Optimal Causal Search for Short Inference Rule Sequences

When cost is determined by the length of the inference rule sequence, then the problem for the optimal policy becomes one of selecting rules that maximize benefit while minimizing cost on the way to finding the smallest Markov-equivalence class.

The application of a conditional independence rule may establish that two variables are independent effectively removing an edge from the model. That single piece of knowledge eliminates up to two thirds of models from the model space. Assume each edge (X, Y) has one of three possible states: $X \longrightarrow Y$, $X \longleftarrow Y$, or $X \quad Y$ indicating no dependence.

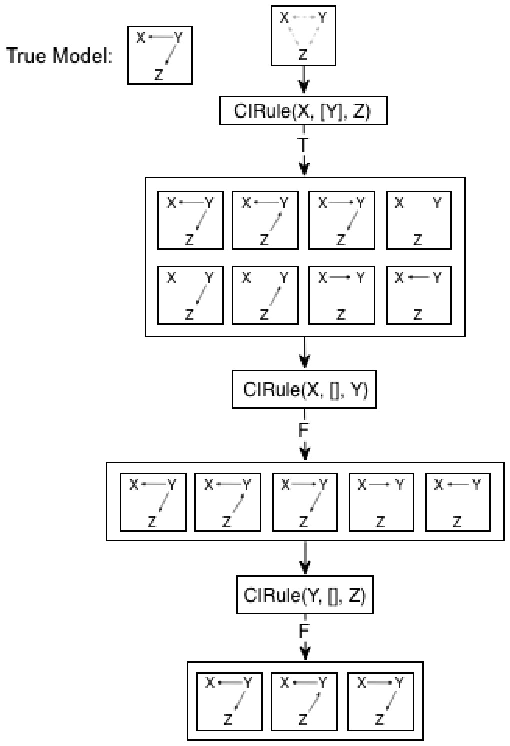


Figure 6.1. The Optimal Causal Plan for a 3-var Example.

The total number of distinct directed graphs is 3^E where $E = V * (V - 1)/2$ and V is the number of variables in the domain. Identifying an independence means two variables are not connected by an edge in the model and eliminates the edge states $X \longrightarrow Y$ and $X \longleftarrow Y$. The number of distinct directed graphs changes to $3^{(E-1)}$. $3^E - 3^{(E-1)} = 3^E * (1 - 3^{-1})$ models are removed. $1 - 3^{-1} = 2/3$. However the number of acyclical directed graphs is smaller than 3^E so at best up to 2/3rds of the model space is eliminated.

If inference rules are limited to apply only on individual edges, as with conditional independence tests, then no inference rule can eliminate more than 2/3rds of the model space. To eliminate more would imply it eliminates more than 2 edge states, but eliminating all three edge states violates the assumptions of the model space in general. This means that

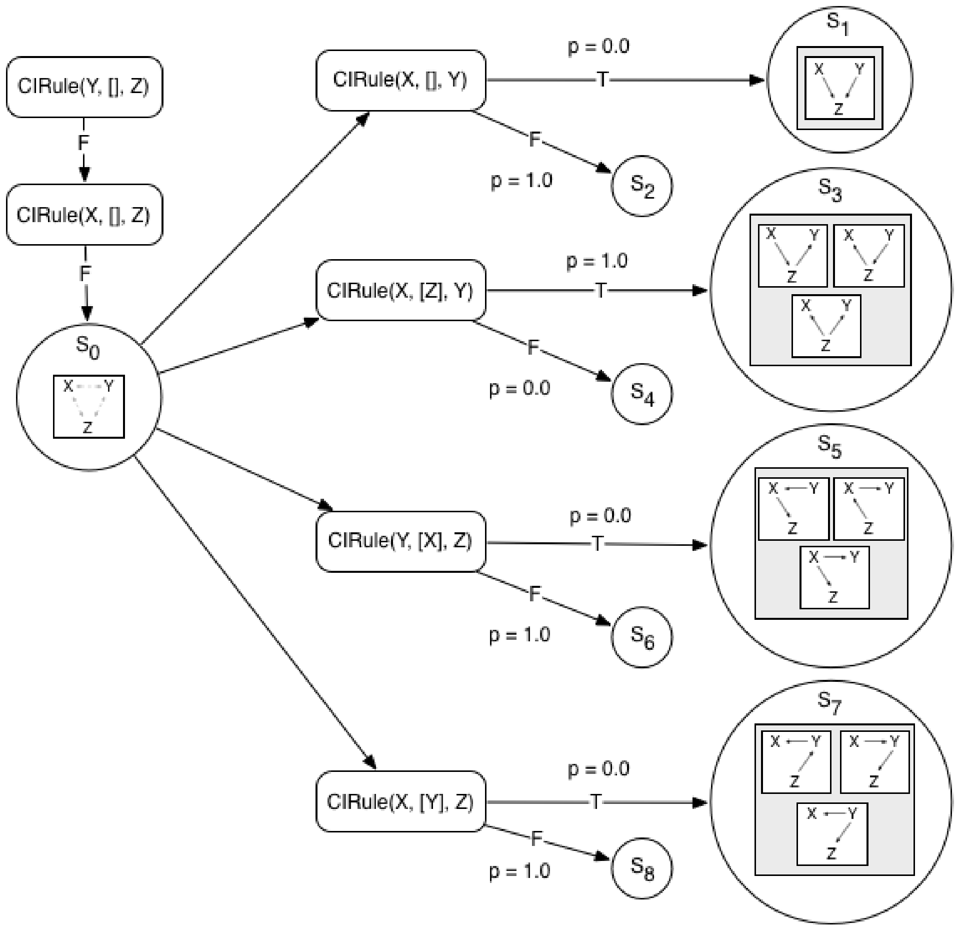


Figure 6.2. An Example of the Optimal Policy Computing Transition Probabilities of Available Actions.

rules such as the conditional independence inference rule or the interventional designs are one of the most informative pieces of analyses that can be applied.

Given a model selected by the oracle for testing, the first step for an optimal policy is to apply all conditional independence inference rules that would be true in the model since each true conditional independence eliminates up to 2/3rds of the alternative models.

The order of the CIRules does not matter as they all must be applied. Applying all the true CIRules first establishes the skeleton of the model but it does not provide enough information to allow for correct orientation of the remaining edges, if the orientations are even possible given the causal skeleton and minimal separating set information. The next step is to eliminate as many other remaining models as possible.

As an example, consider the causal model shown in Figure 6.3.

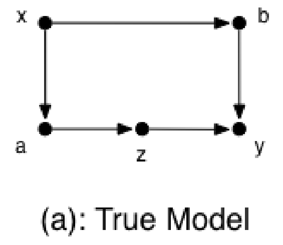


Figure 6.3. A Causal Model Example From a 5-var Domain.

For a 5-variable domain, there are a possible 29,281 causal models. The list of conditional independence inference rules that hold true in the example model, and thus establish its skeleton, are:

$$CIRule(a, [x, z], y)$$
$$CIRule(x, [a], z)$$
$$CIRule(a, [b, z], y)$$
$$CIRule(a, [x], b)$$
$$CIRule(b, [x], z)$$
$$CIRule(x, [a, b], y)$$
$$CIRule(x, [b, z], y)$$
$$CIRule(b, [a], z)$$

There is no need to apply two CIRules on the same pair of variables. This set can then be arbitrarily reduced to:

$$CIRule(a, [b, z], y)$$
$$CIRule(x, [a], z)$$
$$CIRule(a, [x], b)$$
$$CIRule(b, [x], z)$$
$$CIRule(x, [a, b], y)$$

After applying all of these inference rules, the policy is in a state that contains 110 models consistent with the constraints.

..remaining models after applying inference rule $CIRule(a, [b, z], y)$: True, 5940

..remaining models after applying inference rule $CIRule(x, [a], z)$: True, 1036

..remaining models after applying inference rule $CIRule(a, [x], b)$: True, 305

..remaining models after applying inference rule $CIRule(b, [x], z)$: True, 182

..remaining models after applying inference rule $CIRule(x, [a, b], y)$: True, 110

There are two sets of models that remain after the first step. One set contains models that are in the same Markov-equivalence class as the sampled model and cannot be distinguished further. This Markov-equivalence class is shown in Figure 6.4.

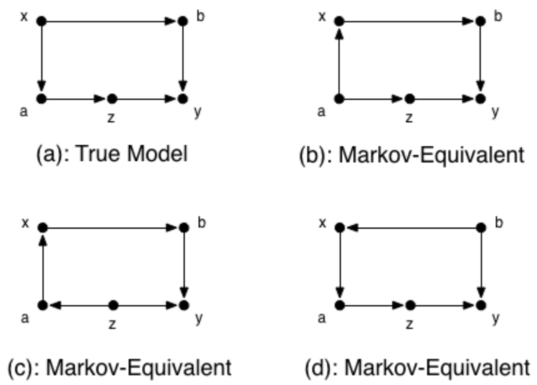


Figure 6.4. The Markov-Equivalence Class Containing the Test Causal Model.

The other set contains models that have the same set of independencies entailed by the applied set of rules, but do so for smaller separating sets. That is each of these non-Markov-equivalent models have at least one $CIRule(X, \mathbf{W}, Y)$ with a smaller separating set $\mathbf{W}$ than the true $CIRule(X, \mathbf{Z}, Y)$ in the model. An example is shown in Figure 6.5 with three causal models that share conditional independencies with the test model but not the corresponding separating sets.

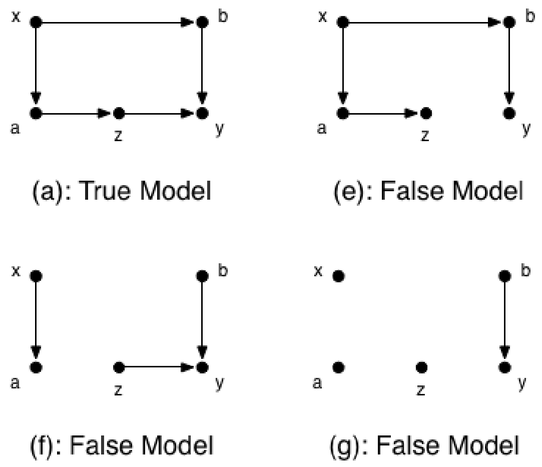


Figure 6.5. An Example of Alternative Models That Entail the Conditional Independencies Matching the Test Causal Model.

6.2.2 Solving Optimal Causal Search for Short Inference Rule Sequences

The Optimal Causal Search at this step is a specialized form of causal learning as an MDP. The initial state for this MDP defines the set of models that remain after applying the true CIRules. The desire is to eliminate all of these alternative models that do not share the same separating sets as the oracle's model and to do so in a way that reaches the model or its Markov-equivalence class in the shortest number of actions. The set of available actions are those CIRules that hold false in the oracle's model but which are true in at least one of the alternative models. And since the shortest path is desired, the cost per each action can be -1.

As an example for the ways the state space changes based on the actions chosen, consider two different choices for the previous 4-variable problem. For all of the CIRules that hold true in this smaller space the number of models for which the rule holds true is counted. This count indicates the number of alternative models that would be eliminated. This is expressed in Table 6.1.

Table 6.1. The Number of Models True for Each Inference Rule.

Inference Rule	Models Holding True	Inference Rule	Models Holding True
CIRule(b, [], z)	92	CIRule(x, [b], y)	27
CIRule(x, [], z)	76	CIRule(a, [z], y)	27
CIRule(a, [], b)	76	CIRule(x, [z], y)	12
CIRule(x, [], y)	67	CIRule(x, [a], y)	12
CIRule(a, [], y)	67	CIRule(a, [x], y)	12
CIRule(y, [], z)	50	CIRule(a, [b], y)	12
CIRule(b, [], y)	50	CIRule(b, [x], y)	5
CIRule(a, [], x)	48	CIRule(b, [z], y)	5
CIRule(b, [], x)	45	CIRule(y, [b], z)	5
CIRule(a, [], z)	45	CIRule(y, [a], z)	5

This list has been shortened by listing only those CIRules that eliminate one or more models. If evaluation of what happens after applying $CIRule(b, [], z)$, the remaining set of CIRules with their counts is shown in Table 6.2. If instead $CIRule(b, [x], y)$ is evaluated first, the result is shown in Table 6.3.

What this shows is that some inference rules constrain the model space more than others. This implies that selecting the best inference rule sequence depends on the benefit of each action and that the benefit can be described by the number of models eliminated.

One ideal solution to this problem is the value-iteration solution. For causal learning MDPs the state utilities are based around cost and not reward. Thus the Bellman Equation

Table 6.2. The Number of Models True After Applying $CIRule(b, [], z)$.

Inference Rule	Models Holding True	Inference Rule	Models Holding True
CIRule(b, [], z)	0	CIRule(x, [b], y)	5
CIRule(x, [], z)	0	CIRule(a, [z], y)	5
CIRule(a, [], b)	0	CIRule(x, [z], y)	5
CIRule(x, [], y)	4	CIRule(x, [a], y)	5
CIRule(a, [], y)	4	CIRule(a, [x], y)	5
CIRule(y, [], z)	4	CIRule(a, [b], y)	5
CIRule(b, [], y)	4	CIRule(b, [x], y)	5
CIRule(a, [], x)	0	CIRule(b, [z], y)	5
CIRule(b, [], x)	0	CIRule(y, [b], z)	5
CIRule(a, [], z)	0	CIRule(y, [a], z)	5

Table 6.3. The Number of Models True After Applying $CIRule(b, [x], y)$.

Inference Rule	Models Holding True	Inference Rule	Models Holding True
CIRule(b, [], z)	92	CIRule(x, [b], y)	27
CIRule(x, [], z)	76	CIRule(a, [z], y)	22
CIRule(a, [], b)	76	CIRule(x, [z], y)	7
CIRule(x, [], y)	67	CIRule(x, [a], y)	7
CIRule(a, [], y)	67	CIRule(a, [x], y)	12
CIRule(y, [], z)	50	CIRule(a, [b], y)	12
CIRule(b, [], y)	50	CIRule(b, [x], y)	0
CIRule(a, [], x)	48	CIRule(b, [z], y)	0
CIRule(b, [], x)	0	CIRule(y, [b], z)	5
CIRule(a, [], z)	0	CIRule(y, [a], z)	5

becomes:

$$U(s) = C(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s') \tag{6.1}$$

The value of the choice made for each state is connected to the immediate cost to apply that action and a fraction of the cost of the next state. For this simple problem, every *CIRule* has a cost of one. Goal states are those states where a fully-oriented model is found, the Markov-equivalence class, or in which no further inference rules are available. Actions to reach goal states would have a cost of one so the utility of the goal state is one. Using the Bellman equation, any state that would reach the goal state via a single action would have a cost of $1 + \gamma$.

When every action has a cost of one, value iteration subsequently identifies the shortest path from the initial state to a goal state since the optimal solution is one that finds the shortest cost path. This means a solution can be found similar to value-iteration within the causal MDP by a depth-first search from the initial state. The search looks for the smallest number of actions required to reach a goal state.

The high-level steps of the Optimal policy are:

1. Identify the set of available inference rules from a state.

2. For each rule, determine the reachable states for each possible outcome of the rule.

3. For each reachable state, compute the cost to reach the state.

4. Continue computing the cost for all subsequently reachable states.

5. Using the Bellman equation, compute the cost for each state.

6. Choose the inference rule that leads to the lowest cost state.

While value-iteration is theoretically a good solution for the optimal search problem, it is impractical. The number of states is potentially super-exponential and it is too computationally difficult to calculate the value for every possible state. If the causal plan is considered to be a k-ray tree where k is the number of inference rules that can be chosen and where h is the length of the optimal inference rule sequence then the number of states to compute is given as:

$$1 + k + k^2 + \cdots + k^h = (k^{(h+1)} - 1)/(k - 1) \tag{6.2}$$

Even if the set of inference rules is small and with simplifications such as constraining the size of k because not all rules are valid at every level in the causal plan, this is still very computationally challenging.

This challenge can be overcome by approximating the value-iteration solution with a greedy-like approach. The hybrid solution performs a depth-first search from the initial state but it only considers some k' of actions instead of every potential action as does the value-iteration approach.

A heuristic is used that calculates the number of models eliminated through each action. The idea is to only consider those k' actions that are most likely going to get to the goal faster and ignore searching across the others. By constraining the size of the set of rules to the most discriminating k' actions, time is not wasted evaluating paths with little chance of being efficient. Simultaneously, considering the best k' actions instead of just the best action alone attempts to reduce the risk of missing an opportunity. But this solution is not completely greedy because $k' << k$.

Using Table 6.1, the pure greedy solution would be to outright apply $CIRule(b, [], z)$ as it has the highest count. Because it holds true in 92 of the remaining models, applying it will reduce the number of alternative models from 110 to 18 which is a huge reduction in the model space. While the pure value-iteration approach would compute the benefit of applying each of the non-zero count CIRules, the hybrid approach will be neither greedy nor completely value-iterative and instead considers only the k=[1, 5] top ranking inference rules.

By setting the value of k the solution can be balanced between being greedy or value-iteration. One could set $k = 1$ for all levels and realize a fully greedy approach. k could also be set to the maximum number of inference rules available at each state realizing the value-iteration solution but even with a domain of size 5-variables this becomes quite computationally long. Experience shows that values of k in the range [2-5] are all that is needed at the higher levels of the search.

Consider an example across a 4-variable domain which contains 543 models. Table 6.4 shows the evaluation of performance of the greedy, hybrid, and value iteration solutions within this domain size to demonstrate the trade-offs. k is limited to at most 5 because evaluation for values greater than 5 did not show any meaningful benefit.

Table 6.4. A Comparison of Optimal MDP Solution Approaches.

Solution Type	Expected Cost	Time (secs)	Solution Type	Expected Cost	Time (secs)
Greedy	5.67	27.4	Hybrid [4-3-2]	5.23	38.6
Hybrid [2-2-1]	5.23	26.7	Hybrid [5-3-2]	5.23	40.7
Hybrid [3-2-1]	5.23	26.9	Hybrid [4-4-2]	5.23	39.4
Hybrid [4-2-1]	5.23	28.3	Hybrid [5-4-2]	5.23	39.5
Hybrid [5-2-1]	5.23	28.9	Hybrid [5-5-2]	5.23	46.6
Hybrid [3-3-1]	5.23	29.6	Hybrid [3-3-3]	5.23	54.8
Hybrid [4-3-1]	5.23	29.8	Hybrid [4-3-3]	5.23	49.5
Hybrid [5-3-1]	5.23	31.0	Hybrid [5-3-3]	5.23	46.9
Hybrid [4-4-1]	5.23	29.8	Hybrid [4-4-3]	5.23	51.1
Hybrid [5-4-1]	5.23	28.2	Hybrid [5-4-3]	5.23	53.8
Hybrid [5-5-1]	5.23	34.2	Hybrid [5-5-3]	5.23	62.5
Hybrid [2-2-2]	5.23	37.4	Hybrid [4-4-4]	5.23	59.5
Hybrid [3-2-2]	5.23	38.7	Hybrid [5-4-4]	5.23	63.5
Hybrid [4-2-2]	5.23	38.0	Hybrid [5-5-4]	5.23	68.3
Hybrid [5-2-2]	5.23	36.0	Hybrid [5-5-5]	5.23	74.0
Hybrid [3-3-2]	5.23	40.7	Value Iteration	5.21	314.7

Hybrid [1st-2nd-All Others] indicates the k-value for the 1st, 2nd, and all other remaining states of the MDP. So Hybrid [5-4-3] indicates that the top five ranking $CIRules$ are considered at the initial state of the MDP; the top 4 ranked rules at any state reachable from the initial state, and the top 3 ranked rules at any state encountered from that point forward. To obtain the data in Table 6.4 each solution 100 was run times for each potential model in the 4-variable domain. The values are the mean costs and times from that evaluation.

Graphically, it is easy to see why the Hybrid approach is preferred.

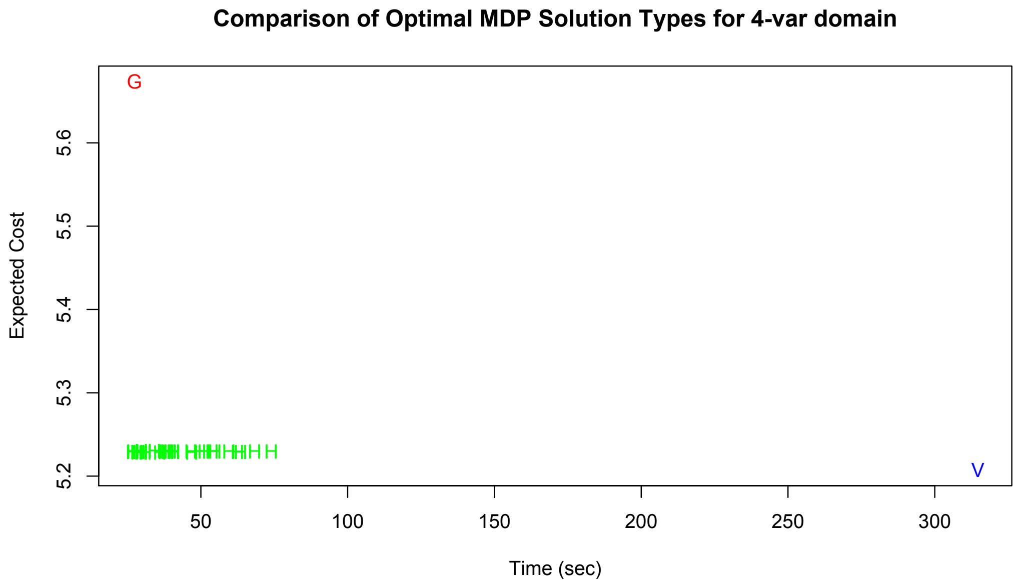


Figure 6.6. Performance of the Different Optimal MDP Solutions.

As can be seen from Figure 6.6, the Hybrid approach gives the better solution for the lowest amount of time. While value-iteration can certainly find the optimal inference rule sequence, it is not necessarily worth the time it takes to do so. Using anything other than $k = 2$ beyond the third level does not seem to improve the output of the algorithm but it can greatly impact its computational requirements. Based on these results, the Optimal Policy is approximated with the Hybrid [3-2-1] solution.

6.3 Evaluating the PC Policy

Evaluation of the PC policy was conducted on domains of size 3- to 10-variables. Evaluation did not proceed beyond 10-variables because the computational complexity grows exponentially and the results up to 10-variables are adequate enough to demonstrate the PC policy performance. *CIRules* were assigned a cost of one and all other inference rules a cost of zero to reflect an evaluation on the total number of conditional independence inference rules used by the policy.

The causal model oracle is used to provide the policies a way to apply inference rules without actually requiring data generation and application of statistical tests across this data. By using an oracle instead of data, we are assuming that conditional independence tests can be made without error. Usage of an oracle means that the evaluation of policies is done in an environment where the assumptions of causal faithfulness, causal sufficiency, and the causal Markov condition hold.

As described in section 5.1 the PC algorithm works in two phases. In phase 1 it uncovers the skeleton of the causal model by applying conditional independence inference rules in order of increasing separating set size until it exhausts the space of valid *CIRules*. By starting with small *CIRules* and then working its way to more complex inference rule, PC can effectively decide the validity of inference rules by the knowledge it gains along the way. After these inference rules, all vertices that have not been found non-adjacent are assumed to be adjacent and the skeleton is simply the set of these edges. The second phase makes use of edge orientation rules, derived from the d-separation criteria, to attempt to orient as many of the edges as possible. The PC policy produces this two phase structure implicitly.

The evaluation works in the following way: Given a domain size, the causal model oracle generates the specified number of models in the domain as a model evaluation set. For domains of size 3- to 5-variables, every single possible model is generated. For domains of size 6- to 10-variables, a subset from all possible models is generated via sampling. The model counts for each domain size is shown in Table 6.5. The subsets are saved to file to allow repeat evaluations and prevent sampling the same model from the model space. Every model is assumed to be equally likely giving a uniform prior over the model space.

Table 6.5. Sampled Model and *CIRule* Counts Across Evaluated Domains.

Domain Size	Sampled Models	Potential Models	*CIRule* Count
3-var	25	25	24
4-var	543	543	384
5-var	29281	29281	10240
6-var	100000	3781503	491520
7-var	15000	$1.1\text{x}10^{9}$	$4.4\text{x}10^{7}$
8-var	5000	$7.8\text{x}10^{11}$	$1.7\text{x}10^{9}$
9-var	500	$1.2\text{x}10^{15}$	$2.47\text{x}10^{12}$
10-var	100	$4.1\text{x}10^{18}$	$1.58\text{x}10^{15}$

For each model in the evaluation set, the model is given to the Optimal policy that uses it as a cheat to guide selecting inference rules. However the PC policy is given no information. It applies inference rules using the oracle to determine the outcomes of each rule and identifies the next step in the PC sequence. For each model the final inference rule sequence is recorded and the cost of the sequence is computed where each *CIRule* has a cost of 1. The expected cost is calculated for each policy by taking the probability of each model, which in this evaluation is uniform, and multiply by each sequence cost. Since all models are equally likely, the expected cost is simply the average number of *CIRules* each policy requires to identify the causal model or its Markov-equivalence class.

Because the space of available inference rules are overly constrained and an oracle is used that does not violate any of the causal assumptions, the Optimal policy identifies the skeleton consistent with the oracle's model and minimal separating sets for each *CIRule*. As the PC policy is a simple, bottom-up approach to uncovering a causal model it will always find the same *CIRules* to be True that the Optimal policy does. Given that both find the same skeleton and the same *CIRules*, they both can apply the same edge-orientation rules which is another reason edge-orientation inference rules are not included in the cost. They do not act in anyway to distinguish the optimality of the PC policy in this evaluation.

The optimality of the PC policy is shown in Figure 6.7.

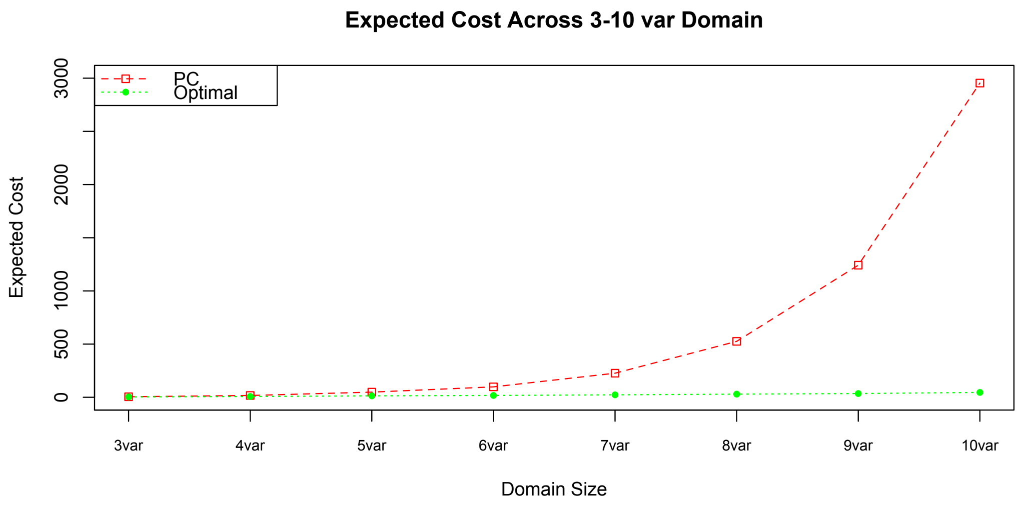


Figure 6.7. Comparison of the PC Policy With the Optimal Policy.

The x-axis shows the number of variables in the domain from 3- to 10-variables. The expected cost across the entire domain is on the y-axis. For every model, the inference rule sequence given by the PC policy and given by the Optimal policy is found. Then using the uniform prior and the total cost for every inference rule sequence, the expected total cost is computed for each domain for each policy.

The PC policy is actually very good for what it does. It is able to avoid $CIRules$ that are unnecessary by ignoring $CIRules$ across two variables that have already been found non-adjacent. It will also ignore inference rules that are inapplicable because those inference rules include separating sets with members that are not neighbors of either X or Y. However as demonstrated in Figure 6.7 the PC policy is still not optimal.

PC has an exponential growth in expected cost as the domain size increases linearly. Because the PC policy is trying *CIRules* with smaller separating sets prior to considering more complex *CIRules*, PC ends up considering several inference rules it does not need to consider. The number of *CIRules* for each domain size is shown in Table 6.5. Since the number of *CIRules* grows exponentially with the number of variables in a domain PC will also experience an exponential growth in the expected number of conditional independence rules it considers.

6.4 Ignorable Inference Rules

One reason that the PC policy is sub-optimal is because PC will consider inference rules that are ignorable given what is has already learned and the d-separation criteria. An ignorable *CIRule* is an inference rule for which the current skeleton and the d-separation criteria means it adds no meaningful additional information. Figure 6.8 represents where X and Y are d-separated given $\mathbf{Z}$.

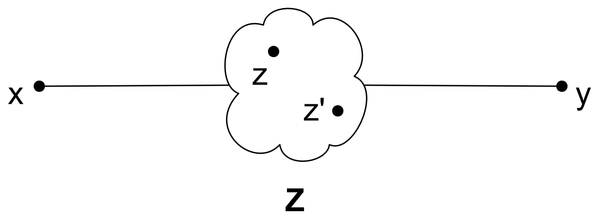


Figure 6.8. X and Y are d-separated Given $\mathbf{Z}$.

There is no graphical structure where Y can d-separate X and any $Z \in \mathbf{Z}$ where some other set **Z'** $\in \mathbf{Z}$ could also not d-separate X and Z. Once $CIRule(X, \mathbf{Z}, Y)$ is known to be True then $\mathbf{Z}$ is either a minimal separating set or it isn't. If it is minimal then $CIRule(X, [Y], Z)$ where $Z \in \mathbf{Z}$ will be False, so it can be ignored. However if $\mathbf{Z}$ is not minimal, but $\mathbf{Z}$ is known to d-separate X and Y, then $CIRule(X, [Y], Z)$ could be True. But in this case there is some other set **Z'** $\in$ **Z** that would also be True so again $CIRule(X, [Y], Z)$ can be ignored.

Theorem 6.4: Given two variables, X and Y, a separating set **Z**, and knowledge that **Z** d-separates X and Y, then testing if Y d-separates X and any $Z \in \mathbf{Z}$ provides no additional benefit than testing if X and Z are d-separated given a subset of **Z'** $\in \mathbf{Z} \backslash Z$. The proof follows.

Definition: Trail $(X, ..., Y)$: A trail is a simple path in a graph between two vertices, X and Y, where the edges along the path may have any orientation.

Definition: Minimal Separating Set: A separating set, **Z**, for two variables, X and Y, provides d-separation for every trail between X and Y. A separating set is minimal if no variable $Z \in \mathbf{Z}$ can be removed without d-connecting X and Y. There may be multiple minimal separating sets $\mathbf{Z}_i$ between two nodes X and Y.

Proof Outline:

Informally, assume that $CIRule(X, \mathbf{Z}, Y)$ is True and pick a $Z \in \mathbf{Z}$ and some **Z'** $\subset$ **Z** where $Z \notin$ **Z'**. Then there will never be a case where $CIRule(X, [Y], Z)$ is True without there also being a case where $CIRule(X, \mathbf{Z'}, Z)$ is also True. So no information is lost by ignoring Y and Y offers no more benefit than another inference rule.

Alternatively, if Y was to d-separate X and some $Z \in \mathbf{Z}$ along a trail T, then there must be some other element of **Z** that also d-separates X and Z along T. If there was not then Z is not a separator because X and Y are d-connected given $\mathbf{Z} \backslash Z$ and since Y lies along a trail from X to Z and d-separates them, Z cannot d-separate X and Y. This means some subset of $\mathbf{Z} \backslash Z$ only needs to be considered and Y can be ignored. The alternative is that Y cannot d-separate X from Z, so it need not be considered because it offers no information which improves the knowledge of the causal skeleton nor orientation of that skeleton.

Formal Proof of Theorem 6.4

Assume $CIRule(X, \mathbf{Z}, Y)$ is True. Apply $CIRule(X, [Y], Z)$ for some $Z \in \mathbf{Z}$. If $CIRule(X, [Y], Z)$ evaluates to True, then one of the following five cases shown in Figure 6.9 must also be true:

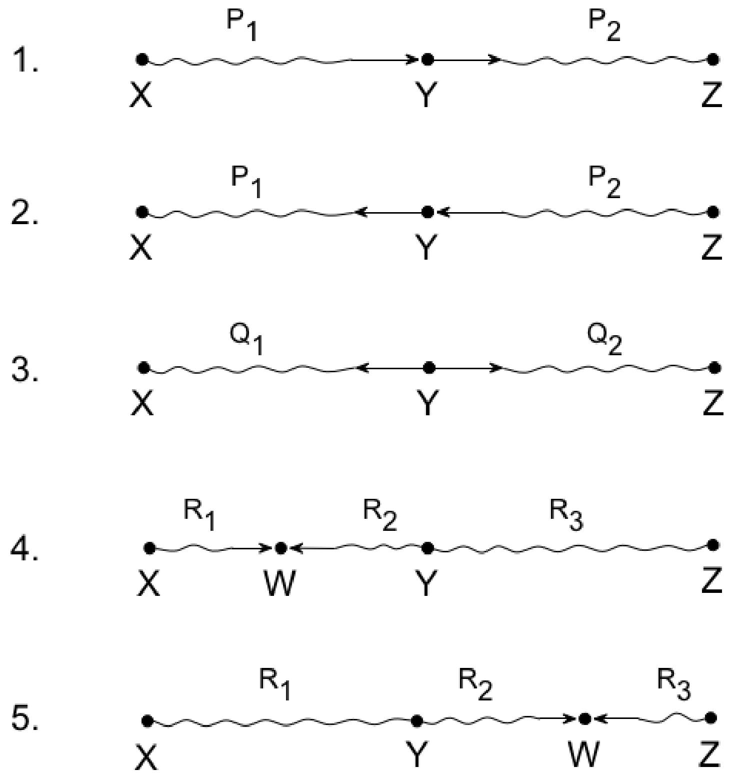


Figure 6.9. Five Cases Where $CIRule(X, [Y], Z)$ is a Redundant Inference Rule.

Case 1 and 2 ($\mathbf{Z}$ is not minimal): The trail $(X, P1, Y, P2, Z)$ contains a chain around Y and by including Y in the separating set for X and Z, d-separates those two nodes along this trail. However, since X is d-separated from Y then some element $Z' \in \mathbf{Z}$ must d-separate X and Y along the trail $(X, P1, Y)$. This same element must also d-separate X from Z along trail $(X, P1, Y, P2, Z)$ so $[Y]$ is not the only minimal separating set for X

and Z along this trail and applying $CIRule(X, [Y], Z)$ is no more informative than applying $CIRule(X, [Z'], Z)$ where $[Z'] \subset \mathbf{Z} \backslash Z$.

Case 3 ($\mathbf{Z}$ is not minimal): The trail $(X, Q1, Y, Q2, Z)$ contains a fork around Y and by including Y in the separating set for X and Z, d-separates those two nodes along this trail. However, since X is d-separated from Y then some element $Z' \in \mathbf{Z}$ must d-separate X and Y along the trail $(X, Q1, Y)$. This same element must also d-separate X from Z along trail $(X, Q1, Y, Q2, Z)$ so $[Y]$ is not the only minimal separating set for X and Z along this trail and applying $CIRule(X, [Y], Z)$ is no more informative than applying $CIRule(X, [Z'], Z)$ where $[Z'] \subset \mathbf{Z} \backslash Z$.

Case 4 and 5: The trail $(X, R1, W, R2, Y, R3, Z)$ contains a collider around W. Since W is not in the separating set $[Y]$, this separating set is able to d-separate X and Z. Note that $[Y]$ is not a minimal separating set. The empty set, $[]$ is a minimal separating set and is able to d-separate X from Z along the trail $(X, R1, W, R2, Y, R3, Z)$, meaning X and Z are marginally independent. Applying $CIRule(X, [Y], Z)$ is less informative than applying $CIRule(X, [], Z)$ where the empty set $[] \subset \mathbf{Z} \backslash Z$. The same argument holds for trail $(X, R1, Y, R2, W, R3, Z)$.

If $CIRule(X, [Y], Z)$ evaluates to False, then one of the following four cases must also be true: $CIRule(X, \mathbf{Z'}, Z)$ is True, $CIRule(X, \mathbf{Z''}, Z)$ is True where $\mathbf{Z''} \in V \backslash \{X, Y, Z\}$ and $\mathbf{Z''}\mathbf{Z'}$, X is a direct cause of Z, or X is a direct effect of Z. In the first and second cases more information is gained by not considering Y and in the third and fourth cases no information is lost by ignoring it. QED.

This theorem is valid even if $\mathbf{Z}$ is not a minimal separating set. That is, if there is some other separating set $\mathbf{Z'}$ smaller than $\mathbf{Z}$ that d-separates X and Y, then Y is still an unnecessary component of a separating set between X and any $Z \in \mathbf{Z}$. This is true because there is no graphical structure such that Y causally blocks X from Z that another member of $\mathbf{Z}$ cannot also block.

Corollary: Given two variables, X and Y, a separating set $\mathbf{Z}$, and knowledge that $\mathbf{Z}$ d-separates X and Y, then Y can safely not be included in any separating set **Z'** when testing if **Z'** d-separates X and any $Z \in \mathbf{Z}$. Once a variable, Y, is shown d-separated from another variable, X, all separating sets that contain Y can be ignored from potentially d-separating X and any member $Z \in \mathbf{Z}$. While it may be that Y is a member of a minimal separating set that d-separates X and Z, this separating set will not be the only minimal separating set, and $\mathbf{Z}$ can be used to help find this separating set.

Corollary: Given two variables, X and Y, a separating set $\mathbf{Z}$, and knowledge that $\mathbf{Z}$ d-separates X and Y, then X can safely not be included in any separating set **Z'** when testing if **Z'** d-separates Y and any $Z \in \mathbf{Z}$. This also means that the inclusion of X can safely be ignored in the separating set of Y and any $Z \in \mathbf{Z}$.

Note that if an algorithm is interested in identifying all of the minimal separating sets, then $CIRule(X, [Y], Z)$ is no longer ignorable. Such a situation arises when one of the causal assumptions could be violated such as causal sufficiency and identification of multiple minimal separating sets could be useful in determining edge orientation.

6.5 New Policy: Augmented PC

The PC policy works by considering the conditional independence inference rules in order of increasing separating set size, eliminating rules that contain non-minimal separating sets by reasoning over the structure learned thus far. This means that if Z is found to be conditionally independent from X and from Y, then there is no need to consider Z as a member of a separating set for X and Y. Graphically this is very simple reasoning and can reduce the space of inference rules that PC considers. Figure 6.10 demonstrates how PC searches across the neighbors of X and Y for potential separating sets.

PC only needs to consider the neighbors A, B, C, and D of X and the neighbors M, N, O, and P of Y. Since these are all components of individual trails between X and Y, there

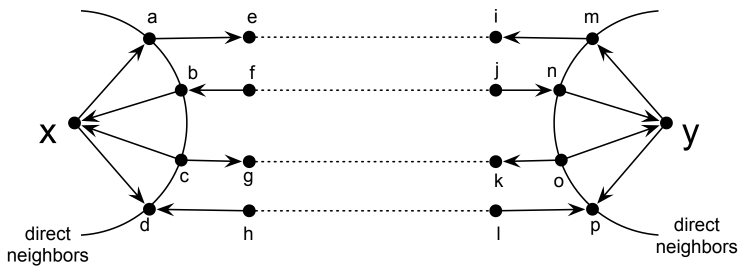


Figure 6.10. How PC Searches for Neighbors Between X and Y.

are only four possible directional structures. Chains and forks are blocked by including the node involved in the structure and colliders are blocked by not including the node in the structure. If some node Q is not a neighbor of X nor Y, then it can be ignored since it is either not on any trail between X and Y or it is on a trail between X and Y but a neighbor of X or Y would block the trail.

However PC fails to identify the ignorable inference rules according to the Theorem 6.4. Once PC identifies that $CIRule(X, \mathbf{Z}, Y)$ is True it knows there is no direct edge between X and Y. But there is still potentially an edge between X and some $Z \in \mathbf{Z}$ and between Y and Z. This means that when the $CIRules$ between X and Z are considered, PC considers Y in the separating set because Y may be connected to Z (see Figure 6.11).

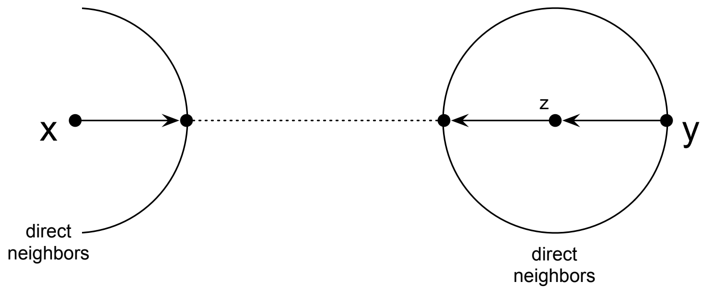


Figure 6.11. The Case Where Y is Still Considered as a Separator of X and Z by PC.

The theorem above explains why one would not need to consider such an inference rule. By failing to ignore these ignorable inference rules, PC applies many more inference rules than it needs.

The PC policy can be extended to correct for this behavior with a policy called the *Augmented PC* policy. Augmented PC will be given a reasoning component to its policy where it identifies $CIRules$ as being ignorable based on the constraints it has already learned. When assessing the benefit of an inference rule, if Augmented PC detects a rule is ignorable, it assigns the rule as having zero benefit. The evaluation verified that the PC policy and the Augmented PC policy identify the same $CIRules$ as being True. This ensures that both identify the same skeleton as well as the same separating sets used for edge-orientation. The only difference then is the inference rule sequences that each can find.

The results are shown in Figure 6.12.

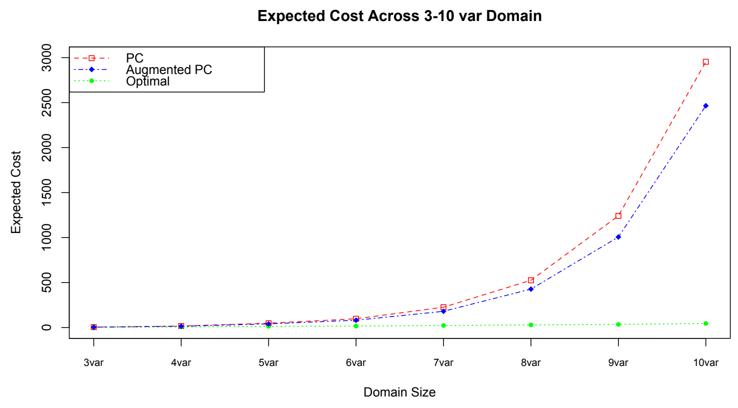


Figure 6.12. Comparison of the PC and Augmented PC Policies With the Optimal Policy.

The Augmented PC policy is able to improve over the PC policy. While the improvement does not eliminate the exponential growth of the inference rule sequence, it does come closer to being optimal.

Table 6.6. Expected Costs from Using PC, Augmented PC, and Optimal Policies.

Domain Size	PC E[cost]	Augmented PC E[cost]	Optimal E[cost]	Models in Domain	Models Sampled
3-var	5.04	4.69	3.36	25	25
4-var	17.01	14.93	7.44	543	543
5-var	48.55	40.93	13.46	29281	29281
6-var	97.92	81.56	17.22	3781503	100000
7-var	226.39	180.87	23.03	$1.1\text{x}10^{9}$	15000
8-var	525.79	427.69	29.60	$7.8\text{x}10^{11}$	5000
9-var	1241.80	1006.48	35.11	$1.2\text{x}10^{15}$	500
10-var	2954.07	2465.56	45.78	$4.1\text{x}10^{18}$	100

Table 6.6 shows the expected costs results across domain size for PC and augmented PC. The improvement over PC grows with domain size. This is due to the exponential increase in the number of $CIRules$ available to PC. Once an inference rule, $CIRule(X, \mathbf{Z}, Y)$ evaluates to True, neither of the two variables X or Y that are conditionally independent of one another should be used in any separating set for the other variable and any member $Z \in \mathbf{Z}$.

6.6 Further Improvements to the PC Policy

Another reason PC is not optimal is that PC will consider all models uniformly likely. This is referred to as an uninformative prior. With this prior, all edge states along every edge in every model, have the same probability. This is a correct assumption when no information on the model space is known.

The Optimal policy is unfairly efficient because it has the oracle's model to guide the selection of inference rules. The Optimal policy knows more about the model space than the PC policy. Rationally then, if a policy knows something more about the model space, it too could choose rules better. While a policy wouldn't ever have the actual causal model, it may have something more than an uninformative prior. With an informative prior over the model space, it would be able to achieve more efficient inference rule sequences.

CHAPTER 7
INFORMED SEARCH OF CAUSAL STRUCTURE

The core of why the uninformed policies are inefficient is a lack of knowledge about the probability of the constraint space and how this probability changes based on what they discover. The PC policy uses a fixed transition probability across the constraint space leading it to incorrectly rank inference rules in terms of their benefit and subsequently the PC policy considers many more actions/states than it actually needs. This inefficiency extends to many other uninformed search policies as many of them such as FCI [50] and Conservative-PC [43] simply extend the PC algorithm.

An alternative is to use the learned constraints to update the transition probabilities from each state. With updated probabilities, the policy can select inference rules which are more likely to find terminal states. Algorithms that incorporate information on the state space are called *informed search* and as will be shown, they can outperform their uninformed counterparts.

7.1 Policies for an Informed Search

Ideally a policy chooses the action that leads to the state with the lowest expected cost. The optimal policy begins by selecting inference rules that constrain the model space the most. For the limited set of rules considered by PC, these are the conditional independence tests that are True. But since an informed policy cannot hope to have the perfect information available to an optimal policy, it can approximate the optimal behavior by computing the transition probabilities for each inference rule. An informed policy can then rank rules in order of evaluating to true and choose those more likely to constrain the model space.

To update the transition probabilities, a policy considers the possible outcomes from each action, a, available in state s. This identifies the set of states $\mathbf{s}'$ reachable from s. Each reachable state, $s' \in$ **s'** defines a set of constraints which entails a class of Markov-equivalent models. Assuming there is a prior over the constraints in state s, then a posterior can be computed over the Markov-equivalence class in s' and a probability for s' can be found. This becomes the probability of the transition for action a into state s' from state s, $T(s, a, s')$.

Informed policies have the following steps:

1. Identify the set of available inference rules from a state.

    Identify an ordering of inference rules by preference.

2. For each rule in the highest preferred set, determine the reachable states for each possible outcome of the rule.

3. For each reachable state, compute a value for the state.

4. Rank inference rules based on the value of the states they can reach.

5. Choose the highest ranking rule and apply it.

6. Repeat steps 1-6 until a terminal state is reached.

7.2 The Posterior Policy

This work presents an informed policy, the *Posterior policy*, that defines the benefit of an inference rule based on how likely it is to constrain the model space. It uses a prior over the initial state s_0 to compute transition probabilities and then updates the probability across the model space through a posterior update. The steps of the Posterior policy are:

1. Identify the set of available inference rules from a state.

   Identify an ordering of inference rules by preference.

2. For each rule in the highest preferred set, determine the reachable states for each possible outcome of the rule.

3. For each reachable state, compute the models consistent with that state.

4. Compute the total probability of the models entailed by the state.

5. Rank the inference rules based on their probabilities of evaluating to True.

6. Choose the highest ranking rule and apply it.

7. Repeat steps 1-6 until a terminal state is reached.

The Posterior policy is similar to the Optimal policy in that it uses the probability over constraints to arrive at a probability over the model space in order to rank the inference rules by likelihood of evaluating to true. By selecting the inference rules that hold true, the policy seeks rules that lead to states with more constraints. Consequently, these actions eliminate larger portions of the alternative models and are more likely to arrive at a terminal state.

The Posterior policy preference criteria is based on an order of increasing complexity or difficulty. It prefers inference rules in the following order:

- Conditional Dependence rules
- Edge-Orientation rules
- Conditional Independence rules
- Quasi-experimental analysis

- Interventional analysis

This order reflects an intuition of cost preference where simple observational analysis is preferred overall. While it could be that an interventional analysis leads to eliminating more of the model space than any set of observational analysis, observational analyses are still far cheaper. Note that the goal in designing the policy is to minimize overall cost. Spending some analysis using observational data before proceeding to more expensive analysis is prudent.

7.3 An Informative Prior over the Model Space

Just as there is a 1:1 mapping between a set of constraints and a Markov-equivalence class of models, there is also a 1:1 mapping between a prior over a set of constraints and a probability mass function (PMF) over the Markov-equivalence class of models consistent with those constraints. Representing a prior over constraints is easy as only a probability for each edge state for every edge in the causal model need be specified. However representing a PMF across the entire model space is very difficult especially when considering that model space size is super-exponential in the number of variables in the domain.

A concise and practical representation of the prior across the model space is needed that facilitates evaluation of the benefits of each inference rule and provides a method by which the prior can be updated into a posterior. Three approaches were identified that provided methods to compute a PMF and maintain a posterior.

Method 1: Keep an exact probability across all possible models in the model space

A knowledge base of deterministic causal constraints combined with an initial prior across those constraints is one possibility that can define a PMF across the model space. As constraints on the model space are found through inferences, the probabilities across the models are updated. When the truth of an inference rule matches the structure of a model, that model becomes more probable. When the truth of an inference rule conflicts with the structure of a model, that model probability goes to zero. This probability specification is referred to as the Exact PMF.

This is a rather naive method because it requires that the probability of every potential causal model in the domain is maintained. It is not compact and in situations where a feature across all models must be evaluates (i.e., the potential validity of a conditional independence rule) it requires enumeration of all models to determine the probability of that feature. This method also requires computing the PMF starting from the initial prior every time and does not calculate the posterior update explicitly.

These problems make this method computationally infeasible given the super-exponential number of models in a domain. The exact PMF is impractical beyond a certain domain size N. Even with $N = 6$, the number of potential models in the domain is nearly 3.8 million. Performance can be improved by considering elastic map-reduce systems, but eventually a limit will be hit where the exact PMF has to be abandoned.

Method 2: Use a prior over structures that is computed against the structure

One approach that is found in Bayesian network learning is to assume a structural form over the models and to compute a probability of the structure [15]. This is how search-and-score techniques typically work. The probability of a model or graph is composed of the product of the probability of the structure for each individual variable in the model. For example, you can treat the family of each variable as independent of all other variables and compute the probability of the family for each variable. The probability of any graph then is the product of all the probabilities of the variables.

$$P(G|D) = product(p(V)) \text{ for all } V \text{ in the model.}$$

More specifically, one can use an MCMC approach that on mixing represents the underlying PMF for the model space. You then sample from this model space as many times as you want in order to compute some feature such as the probability that a conditional independence would hold or not.

$P(G)$ could be computed using deterministic constraints combined with the prior. But the PMF would need to be recomputed every time something is learned and for every model under consideration. It also requires that the models considered are constrained to structures for which priors have been specified rather than specifying a prior over the edges directly.

Method 3: Approximate by keeping a probability across only a set M of models

Rather than maintaining a PMF across the entire model space, a smaller set, M, of models can be maintained along with their probabilities. The set M contains a sample of models from the model space that can be drawn by using the informative prior. As constraints are learned, models from M are removed that conflict with what has been learned and new models are drawn by updating the prior. This method is referred to as the Approximate PMF.

The sampled model set M is used to guide the policy in deciding what inference rules to apply and in what order. If all models are eliminated from M, then a new set of sampled models is found to guide the policy. This new set must not disagree with what has already been learned and so the learned constraints are used to guide finding new models.

7.3.1 Implementation of an Approximate PMF

Method 3 was chosen to approximate the PMF as it is similar to other approaches within Bayesian network learning and it presents an acceptable trade-off between accuracy and computational requirements.

The purpose of the set M is to find models that represent the current state based on the initial prior and using the constraints that have been learned. Ideally M contains the

Markov-equivalence class that is most consistent with the data. Most likely this set M also holds models whose structure implies potential unapplied inference rules that may be true yielding a set of actions that must still be considered. As with the Optimal policy these alternative models are to be eliminated from M by applying inference rules from this set of actions.

The prior on the constraints is defined as so: Each edge (X, Y) has one of three possible states with a probability on each state:

$$P(X \longrightarrow Y),$$
$$P(X \longleftarrow Y),$$
$$P(X \qquad Y).$$

The total sum of the probability on an edge must add up to 1.0. If one state is not possible, it is assigned a probability of 0.0. The prior need not be specified for every edge. When an informative prior is unavailable, a prior without any specification can be used by assuming each edge state has uniform probability.

By using the prior on the constraints a set of models is drawn to approximate the PMF of the model space. The goal is to draw a representative sample of the model space to help the policy identify which rules are more beneficial at finding a terminal state. In drawing samples models with cycles are discarded and models that have already been seen are rejected. There are different ways to sample models for inclusion into M.

One could choose the highest probable models and add models into M in order of decreasing probability. This might lead to focusing too much on an area of the model space missing opportunities to constrain the model space better by a more spread out approach. The initial prior may incorrectly identify a set of models, potentially a Markov-equivalence class, as being highly probable when in fact the most consistent causal model is initially much less probable. By over sampling from the higher probability models, the policy could spend a lot of wasted effort by ignoring those less probable models. However if these high-probable models were incorrect, early sampling would eliminate a large part of the initial model space that would in turn modify the PMF to focus more on the distribution of consistent models.

A method could randomly sample models from the prior using rejection sampling. A model could be sampled by randomly generating a causal structure and then using the prior to compute the model's probability. This generates a set M that is not connected to the initial prior and so isn't truly representative of the PMF defined by it. Instead a more principled approach can be taken to generate an edge state by sampling from the probabilities along each edge. This creates an M that is more representative of the space. Models are then rejected that are duplicates or have a zero probability.

Deciding the size of M can be tricky because all models in M may still be valid, yet the set of valid inference rules across M may have been exhausted. This can happen when M contains a subset of a Markov-equivalence class. In the worst case, M contains a subset of a Markov-equivalence class that contains models that are consistent with the data except for a single edge. Within this Markov-equivalence subset every inference rule can be executed and not a single member of M may be eliminated. So the approximate PMF approach must be able to identify this condition and be prepared to enlarge M to grab more models from the model space.

When going back to resample models, a decision must be made on how many should be re-sampled. The set of models is resampled until either all potential models are exhausted or until a sample size is found at least as big as M and that contains at least one model which has an inference rule that has not previously been evaluated and which may be true. This means M can be adjusted upwards if needed. Of course if there simply are no more models available then the algorithm will detect that it has found a Markov-equivalence class and stop sampling.

Two ways were implemented to perform an approximation to the PMF of the model space: sampling from the model space with rejection and walking a structure called the prior-index. First, model sampling is performed to obtain a set of candidate models M using rejection sampling. Resampling is done as needed to as models are eliminated from M. When the rejection rate gets too high a second, more systematic method, is used that is referred to as the prior-index. The prior index method guarantees all remaining valid models can be found albeit with a computational burden. This burden is hopefully reduced if rejection sampling was done for sufficiently long enough.

7.3.1.1 Rejection Sampling

Given a prior on the model space, sampling can progress edge-by-edge over the constraints and randomly choosing an edge state for each edge. The model is then defined as the set of edge states taken together. The probability of the model is defined as the product of the probability of each edge state.

$$p_{model} = \prod_{i=0}^{[V*(V-1)/2])} p_{edge_i}$$

As models are sampled they are added into the set M. Models are rejected that have already been added into M, for which their probability is 0.0, or which contain a cycle. The membership check is straight forward and eliminates most duplicates. Models with a probability of 0.0 are easily detected because they contain an edge state with probability 0.0 or because the model is invalid given the constraints learned thus far. To identify models with cycle, detection can be done with a topological sorting [20]. After every model is drawn and checked for membership and probability, cycle detection is applied to eliminate cyclical models. If the model is not rejected, the model is added into M and sampling is repeated.

Situations can be encountered where large number of sampled models is rejected making rejection sampling computationally expensive to fill M. If a probability across the model space is such that one model has an incredibly high probability and all others have incredibly small numbers then the high probability model could be sampled over and over. Another situation is one where there are several local maximums along the probability density function that capture several models in the space and so repeated sampling find these frequently but miss the models within the local valleys. Models with very low probabilities may take an incredibly long time to ever find. This situation is especially common for models where at least one edge state has a near-zero probability.

To detect when rejection sampling has become unviable, two metrics are used. The first is to place a limit on the number of bad models in a row that are accepted before abandoning the rejection sampling method. The limit is set to be sufficiently high enough to allow a large number of rejected models but not so high that large amounts of time is wasted looking for a previously unseen model. Currently the limit is configurable with each experiment and values of 100, 1000, or 10000 have been used. The second is to keep track of the rejection rate computed as the number of models rejected divided by the total number of models sampled. This is rate is referred to as the acceptRatio. When the acceptRatio is 0.05 or less indicating that 95% of the sampled models are being rejected, the sampling technique is no longer beneficial.

Sampling continues until either the sample set size M of models is found or until rejection sampling has been determined to be insufficient. This challenge is overcome by falling back to an exhaustive and principled approach to sampling from the model space, the prior index.

7.3.1.2 The Prior Index

The prior index technique is a method that allows exhaustive sampling of every possible model in the model space but with a computational trade-off. The technique works by treating the prior specification as a numerical representation and generates every possible model by incrementing the representation. Specifically it: Assigns each edge state an index of 0, 1, or 2 (assuming the three possible states). It does not matter the order of the edges, though alphabetically makes reading the prior easier. The prior index is initially set as the model with an index of 0 for each edge. For example in a 3-variable domain consisting of variables X, Y, and Z, the initial index corresponds to [0, 0, 0] that could also be referenced as $X \longrightarrow Y$, $X \longrightarrow Z$, $Y \longrightarrow Z$.

The technique then treats the prior index as a number base i where i is the number of edge states and the dimensions of the index is j where j is the number of edges. Each index identifies a model by reading off the number at each digit as an edge state for the model. If the model designated by this index isn't rejected it is added into the sample M. Increment the prior index by one and repeat checking if the current index is a model that can be added into M and incrementing the index until the index is incremented to j^i.

The important component to the prior index method of generating samples is how to determine if the model corresponding to the current index is rejected or saved. There are three criteria that match the criteria used in rejection sampling. First, ignore models already in M. Second, verify if the model contains a cycle and discard any model with a cycle using the same topological sorting cycle detection. Third, verify that the model does not conflict with the current set of found constraints or has a non-zero probability. If a model disagrees with any constraint or is zero probability, discard the model.

An advantage to using the prior index is that it is possible to sort the edge states in order of decreasing probability. In this way, the edge state with index 0 would be the edge state with the highest probability. The initial model, the prior index corresponding to all zeros, then is the most probable model in the model space. Sampling using sorted priors could be useful for anytime algorithms that grab the most probable models first as these models are most likely to be true given the original prior.

Another advantage is that maintaining the prior index reduces the data that has to be stored to know the location in the prior space during sampling. All that is needed to maintain the prior index is the index itself. After some number of models M have been drawn, the last prior index is stored to continue sampling from that point if more models are needed.

In the worst case scenario, however, the prior index will touch on every possible model, cyclical and acyclical, in the model space. To reduce the potential of having to enumerate through the entire model space a technique is used that can simplify the prior. As independence/dependence constraints are learned, they can be used to modify the prior on a per-edge basis. If an independence is true, then two of the three possible edge states are not allowed and the prior index can eliminate these edge states by setting them to zero. Additionally, if it should be discovered that a dependence between X and Y or even better an edge orientation such as $X \longleftarrow Y$, that too can change the prior by eliminating the edge state $(X \qquad Y)$ from consideration. To facilitate this convenience, if a given edge state has a zero probability then it is removed from the index and never contributes to a sampled model.

7.3.2 Updating the Prior into a Posterior

The prior is quite informative as a guide for any policy but the true usefulness comes in being able to update this into a posterior distribution based on the learned constraints. The sample M is used to allow an approximation of the posterior distribution by updating the individual model probabilities. When each sample model is placed into M its probability is computed based on the initial prior. After the end of sampling the probabilities of all models in M can be normalized by computing the total probability of M and dividing each model probability by the total. As models are eliminated from the model space by what is learned, they are effectively being removed from the total probability. This means that over time, M comes closer to representing a model space with higher probability of matching the distribution of data and thus a more accurate posterior can be computed.

The posterior is maintained by normalizing the probabilities within M rather than updating the initial prior directly. Updating an edge state probability, i.e., $p(X \longrightarrow Y)$, based on the learned constraints does not lead to a true posterior because this update is independent of all other edge states. For example simply setting one of the three possible edge states to 0.0 does not immediately imply the other two edge states are each 50% more probable. One reason is that causal models with cycles are not allowed and their existence impacts the edge state probabilities.

A representational trick is used to aid in sampling: whenever it is learned that an edge state is disallowed due to a constraint, the probability of that edge state is set to 0.0 without updating the other edge states. These kinds of updates are maintained in a version of the initial prior to speed up the identification of valid models during sampling. If it is known ahead of time that a model is going to have zero probability as identified by the edge state having zero probability, this model can be skipped altogether.

7.4 Details of The Posterior Policy

The approach taken by the Posterior policy can be described as a best-first search that greedily chooses the inference rule that yields the best informative gain. Higher probable rules indicate actions that lead closer to discovering the underlying causal model. Simultaneously, the inference rules that do evaluate to True eliminate large portions of the alternative models. Conditional independence inference rules can eliminate up to 2/3rds of the model space as the constrain the causal skeleton. Edge-orientation rules can also eliminate up to 1/2 of the model space as they eliminate potential causal directions.

The high-level overview of how the Posterior policy follows.

1. Using the prior and learned constraints, identify a set M of models.

2. From this sampled model set M, identify the set CI of inference rules valid for these models. Note that this set, CI, of rules may not contain all inference rules that would be valid against the entire model space. CI is essentially a subset of all inference rules that might be encountered.

3. Rank the inference rules by the probability of the models in M.

4. Choose one inference rule, r_1.

5. Based on the result of applying r_1, update the probability of the sampled models M.

6. Repeat steps 3-5 until there are no more inference rules in CI. At this point all other potential causal models have been eliminated or the set of rules across the sample has been exhausted.

7. Re-sample from the model space using the prior and the constraints learned thus far to obtain a new set of sampled models M.

8. Repeat steps 2-7 until there are no more inference rules left.

CHAPTER 8
EVALUATION OF THE POSTERIOR POLICY

The primary metric used in this work has been focused on optimality as a function of the number of inference rules taken to discover a causal model. This evaluation criteria is used to investigate the efficiency of the Posterior policy. Since the Posterior policy is being compared against the PC policy, the set of rules are limited to those used by PC. The inference rule sequence can still be described in terms of cost but conditional independence rules are assumed to have a unitless cost of one while the conditional dependence and edge-orientation rules have zero cost. Other cost assignments can be used as can additional rules but this simple assignment and small set of rules makes it clear what efficiency is within the rule set for PC.

8.1 Comparing Policies on Expected Cost

In the previous comparison of the PC policy against the Optimal policy, efficiency was calculated as a total cost of the inference rule sequence. The average cost of a policy is calculated by taking the average total cost across all models used in the evaluation. This average cost is then used to compare the PC policy against Augmented PC and the Optimal policy.

However when a probability is used across the model space, then this calculation is modified by taking into account the probability of each model. This changes the description of the cost of an inference rule sequence to one of an expected cost. When all causal models are equally likely, expected cost is the same as the average cost incurred by the policy across all models. If a cost is assigned to every inference rule within the sequence, the entire sequence can be summed to derive a total cost for executing every rule. The probability of the causal model for which the inference rule sequence is identified can be determined by using the prior over the model space. With the total cost of the inference rule sequence and the probability of the causal model, the expected cost of the inference rule sequence can be computed.

$$E_{cost} = \sum_{m=1}^{|models|} p_m * cost_m \tag{8.1}$$

To obtain an accurate assessment of the performance of a policy, policies are applied across multiple models across domains of varying size. For a domain of a particular size, some number of causal models are drawn that could exist in that domain. For each causal model, an inference rule sequence is obtained for each policy under evaluation. The expected cost of the policy is then computed for each sized domain. This gives an assessment of how the different policies would behave within a domain of a certain size. The expected costs give a metric that allows evaluation between the different policies. When randomness is a component of the planning policy, the policy can be used multiple times on the same model to obtain an average of the cost of the policy's inference rule sequence for that particular model.

As done with the comparison of the PC policy against the Optimal policy, the usage of the oracle means we assume that all conditional independence tests can be applied without error and that the assumptions of causal faithfulness, causal sufficiency, and the causal Markov condition hold.

8.2 Comparing Search Policies on Correctness

As done when comparing the performance of the PC policy against the Optimal policy, a causal model oracle is used to both randomly sample models from the model space and to correctly answer rules applied to those models. To determine policy correctness policies are compared across causal skeleton precision, causal skeleton recall and edge-orientations.

Precision is the proportion of edges included in the learned model that are correct, while recall is the proportion of edges in the sampled model that are included in the learned model [33]. With the oracle and under the assumption that there are no latent variables, search policies in this evaluation are able to identify the entire causal skeleton. While precision and recall are normally used to help understand a learning algorithm's Type I and Type II errors, for this evaluation they are only used to ensure that precision and recall match exactly across the search policies. If policies did not match it would indicate that at least one policy was unable to fully learn the causal model making further evaluation futile.

It is possible that search policies arrive at the same causal skeleton but do so with different inference rules leading to different edge-orientations. To determine that causal search policies are in agreement on conditional independence rules, edge-orientations can be compared between them and against the true causal model. This detects if a policy misidentifies a minimal separating set when finding a conditional independence. Because the edge-orientation rules have been found to be correct [34] if one policy had an incorrect separating set it may not be able to have the same orientation as found in the sampled model.

Note that search policies may not be able to fully orient the causal models they learn. When the best that a policy can do is identify the Markov-equivalence class, then full orientation is not possible. In this case comparisons are done across those edges that are consistently oriented in the set.

All policies in the evaluation produced models with the same precision, recall, and edge-orientation and for this reason those criteria are not reported in the statistics.

8.3 Performance of the Posterior Policy

To evaluate the performance impact from utilizing an informed search policy, several tests were conducted to compare the performance of the Posterior policy against the PC policy and the Optimal policy. The augmented PC policy was not chosen in place of the PC policy because PC is a more recognized policy that is being used by the causality community. This would allow highlight of any differences found between informed search and standard, uninformed search.

8.3.1 Performance with an Uninformative Prior

The first evaluation criteria is to determine the performance of an informed search policy, the Posterior policy, in an environment where most learning is applied: uninformative priors. As explained in section 6.3, an uninformative prior is a prior where every model in the model space has equal probability initially giving a policy no indication where to begin. Evaluation was performed across 3- to 10-variable domains.

8.3.1.1 Performance Evaluation

Figure 8.1 shows the results of the PC policy, the Posterior policy, and the Optimal policy. The expected costs are also shown in Table 8.1.

It is very apparent that the expected costs for both the PC and the Posterior policies grow exponentially with the size of the domain but that the Posterior policy grows much slower than does PC. For the 10-variable domain size, the Posterior policy finds the Markov-equivalence class using about 28% of the rules that the PC policy uses. The Optimal policy appears to be unchanging, but it is in fact slowly increasing across domain size.

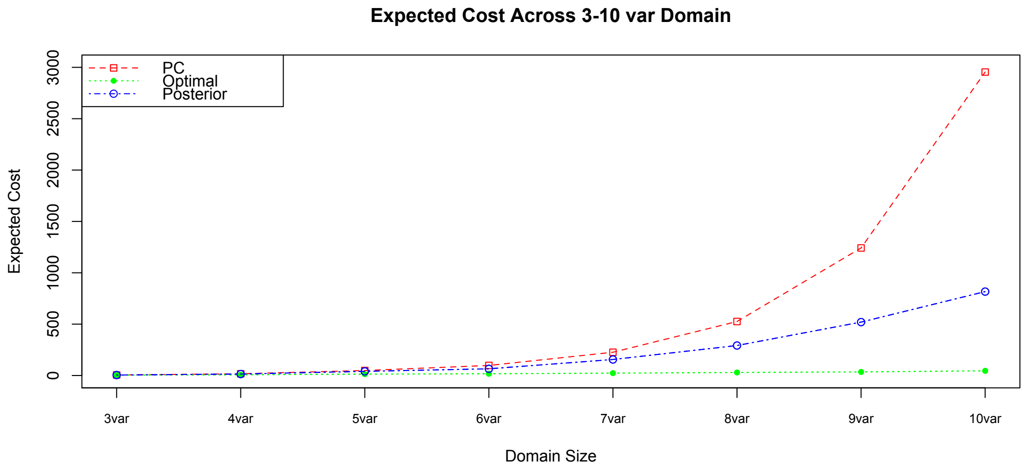


Figure 8.1. Comparison of PC, Posterior, and Optimal Causal Policies on an Uninformative Prior.

To demonstrate finer detail, Figure 8.2 shows the results limited to the 3-, 4-, and 5-variable domains.

The PC policy is shown as doing well for very small domains, as compared against an informed policy with an uninformed prior. Even at 5-variables, the PC policy is still doing fairly well. But the performance fades beyond 5-variables. This is where the Posterior, even without a lot of initial information on the model space, consistently outperforms the PC policy. The distribution of percent improvements are shown in Figures 8.3-8.5. For the 3-variable domain, improvement is modest. But for the 4-var and 5-var domains, the Posterior is identifying around 20% shorter inference rule sequences than does the PC policy.

PC is more efficient than Posterior for those models which can be found through the fixed approach of an uninformed policy. In the 3-variable domain, these models are the three possible *colliders*, those models with two edges where two variables are directly causal for the third variable. An example is shown in Figure 8.6(a).

Table 8.1. The Expected Costs for the PC, Posterior, and Optimal Policies on an Uninformative Prior.

Domain	Optimal	Posterior	PC
3	3.36	4.60	5.04
4	7.44	14.28	17.01
5	13.46	40.55	48.55
6	17.22	65.20	97.92
7	23.03	156.01	226.39
8	29.60	292.98	525.79
9	35.11	520.48	1241.80
10	45.78	817.62	2954.07

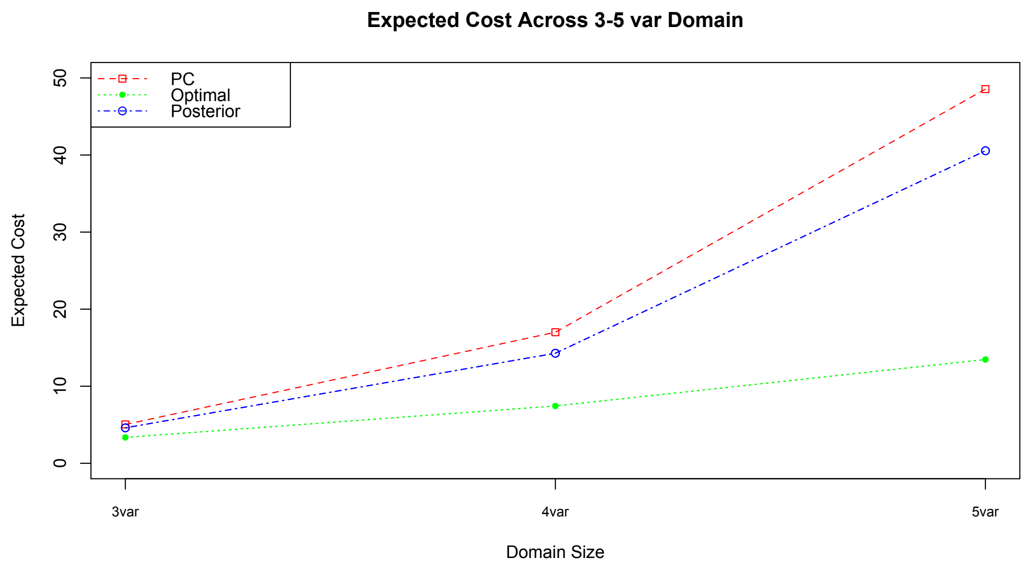


Figure 8.2. Comparison of PC, Posterior, and Optimal Causal Policies on an Uninformative Prior on Domains of Size 3-5 Variables.

Both the PC and Posterior policies perform similarly for the *empty* model that has no edges (Figure 8.7(a)), models with a *single edge* (Figure 8.7(b)), and *3-variable clique* models with three edges (Figure 8.7(c)). For the empty model, both PC and Posterior can quickly find the lack of dependence between all of the variables. The 3-variable clique in a 3-variable

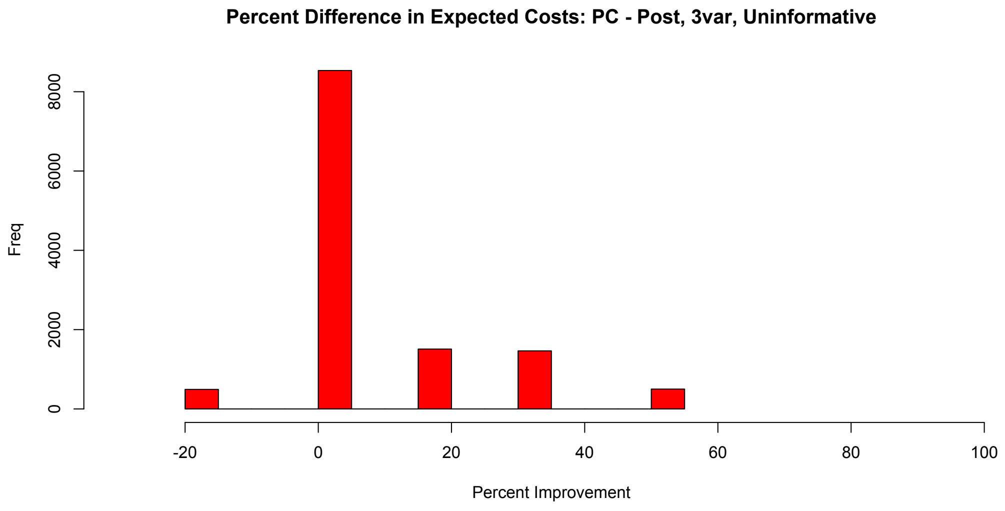


Figure 8.3. Percent Improvement of the Posterior Policy Over the PC Policy in the 3-variable Domain.

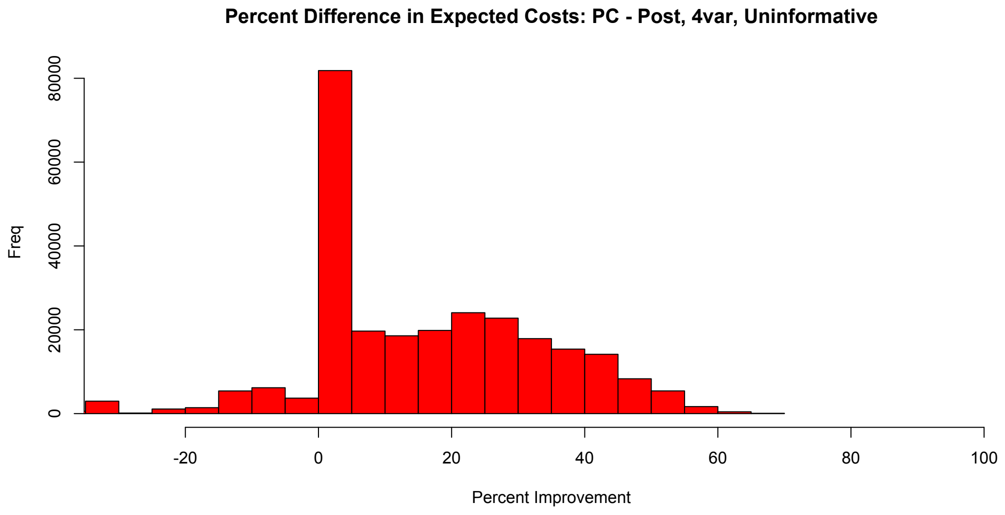


Figure 8.4. Percent Improvement of the Posterior Policy Over the PC Policy in the 4-variable Domain.

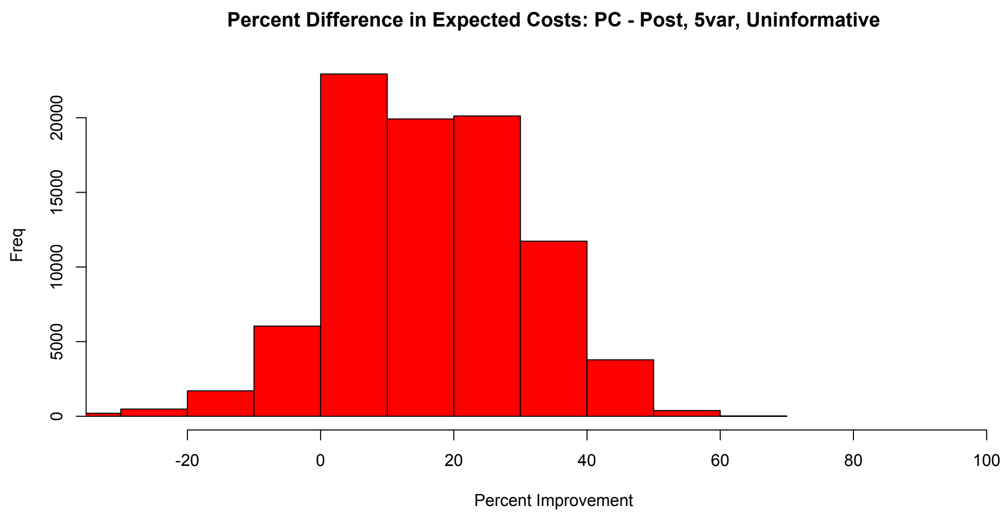


Figure 8.5. Percent Improvement of the Posterior Policy Over the PC Policy in the 5-variable Domain.

domain is a fully-connected model. Every conditional independence test must be exhausted by both policies to find the edges so the policies perform equally here as well. The single edge models are easily identified by PC as three marginal independence tests are sufficient. The Posterior, having no information to utilize, also begins with the marginal independence tests which is enough to help it identify these models as well.

Where the Posterior policy outperforms PC is on the models with two edges that do not contain a collider. These are models composed of *chains* (Figure 8.8(a)) such as $Y \longrightarrow Z$, $Z \longrightarrow X$ or models composed of *forks* (Figure 8.8(b)) such as $Y \longrightarrow Z$, $Y \longrightarrow X$.

In the 4-variable domain, the PC policy outperforms the Posterior policy on 72 of the 543 possible models. The best class of models for PC are the *two disconnected edges* models (Figure 8.9(a)), the *two colliders plus one edge* models (Figure 8.9(b)) and the *three colliders* models (Figure 8.9(c)) as well as the *colliders* demonstrated in the 3-variable domain.

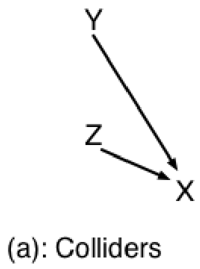


Figure 8.6. Model Structure for the 3-variable Domain in which PC outperforms the Posterior.

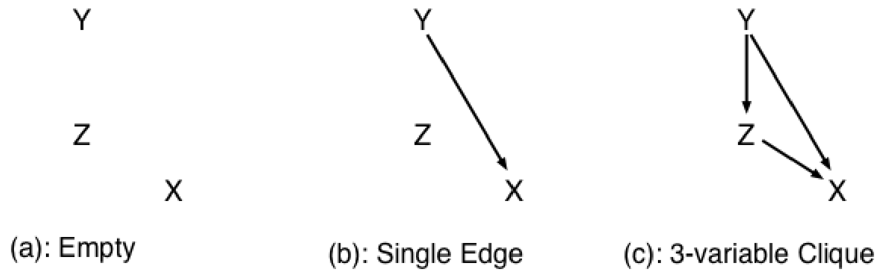


Figure 8.7. Model Structures for the 3-variable Domain in which PC and Posterior Perform the Same.

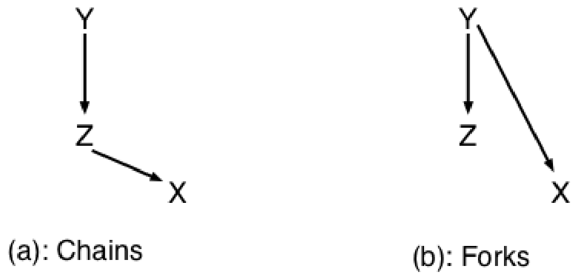


Figure 8.8. Model Structures for the 3-variable Domain in which the Posterior outperforms PC.

There are 53 models where PC and the Posterior perform the same. These models are the *empty* model with no edges, the *fully-connected hierarchies* and the *3-variable cliques.*

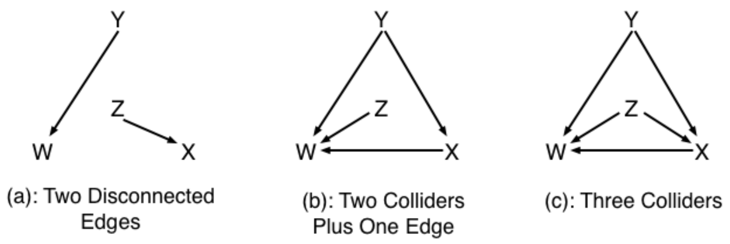


Figure 8.9. Model Structures for the 4-variable Domain in which PC outperforms the Posterior.

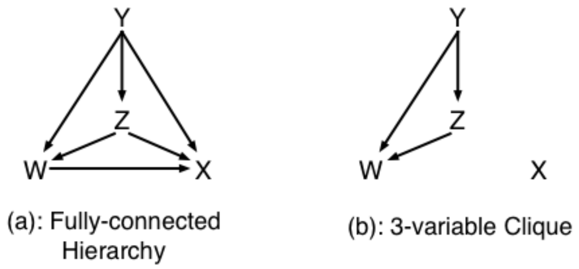


Figure 8.10. Model Structures for the 4-variable Domain in which PC and Posterior Perform the Same.

The *fully-connected hierarchies* are models with six edges where one variable is causal for the other three, one variable is causal for two, and one variable is causal for one. This is described as a hierarchy because the causality runs outward from one variable and one variable is directly caused by every other variable. An example is shown in Figure 8.10(a). The *3-variable cliques* (see Figure 8.10(b)) are the same as found in the 3-variable domain (Figure 8.7(c)), but they are not fully-connected models in the 4-variable domain.

Of the remaining 418 models, the Posterior can outperform PC, even without an initial prior. These are models with two to five edges. A representative sample is shown in Figure 8.11(a)-(l). As in the 3-variable domain, the models easier for the Posterior to identify are

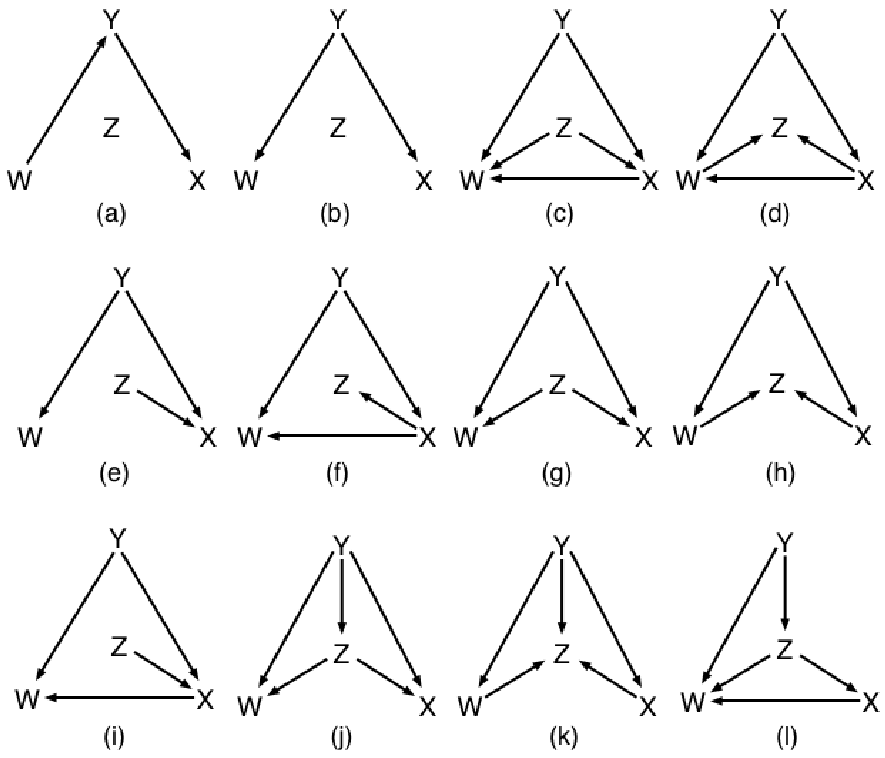


Figure 8.11. Model Structures for the 4-variable Domain in which the Posterior outperforms PC.

ones with *single chains* and the *single forks*. The Posterior also performs well when there is a combination of colliders and forks.

With over twenty-nine thousand models in the 5-variable domain, the specific performances are not broken down by model. But cursory inspection shows that PC continues to outperform the Posterior for models with only a few edges while the Posterior outperforms PC for models between two and nine edges.

These results imply that an informed search with an uninformative prior will have difficulty identifying very sparse causal models. There will also be no performance gain for fully-connected models. However the real advantage for an informed search is on models between the two extremes that are expected to have some causal structure. As the domain size increases, the size of the set of models which fit between these two extremes increases which explains why an informed policy would be preferred beyond small domains.

Another observation on using an uninformative prior is that when the domain size is small, the amount of information gained from learning the result of an inference rule is minimal in regards to the number of models eliminated. This explains why an uninformed policy such as PC can still look competitive.

But as the domain size increases, learning the result of an inference rule can eliminate a larger set of alternative models. This elimination narrows down the inference necessary to discern among the remaining models of which an uninformed policy is unaware. As the domain size increases, the uninformed policy begins to apply longer inference rule sequences that end up being wasted efforts while the informed policy uses fewer rules because it is updating its knowledge based on what it learns.

8.3.1.2 Example of the Posterior Inference Rule Sequence Under an Uninformative Prior

To demonstrate the different inference rule sequences between the PC policy and the Posterior policy, consider the model shown in Figure 8.12 where Y is causal for both X and Z.

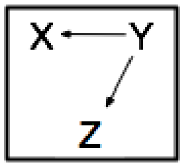


Figure 8.12. Example Model to Demonstrate the Inference Rule Sequence Found by the Posterior Policy When Given an Uninformative Prior.

Under a uniform prior, this model has a probability of 0.037037, including cyclical models in the model space. Without cyclical models, this probability is adjusted to 0.04.

The PC policy will initially consider marginal independence tests followed by conditional independence tests. If conditional dependence rules and edge-orientation rules are ignored, the PC policy will produce the following inference rule sequence:

$$CIRule(X,[],Z):F;\ CIRule(X,[],Y):F;\ CIRule(Y,[],Z):F;\ CIRule(X,[Y],Z):T;$$
$$CIRule(Y,[X],Z):F;\ CIRule(X,[Z],Y):F;$$

Here we've identified the output of the rule as being either True (T) or False (F). Note that this inference rule sequence can only identify a Markov-equivalence class in Figure 8.13.

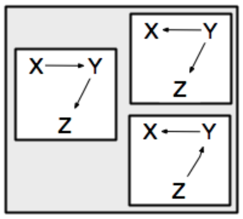


Figure 8.13. The Markov-Equivalence Class Containing a Set of Models Consistent With the Data That Cannot Be Distinguished Further by PC.

Since the PC algorithm does not specify an exact order across variables, a random order across variables is allowed during each application of the PC policy, so distinct iterations may produce slightly different orderings but the composition of the inference rule sequence remains consistent. Even though the rule $CIRule(X,[Y],Z)$ that results in True can completely explain the structure of the model, the PC policy will continue trying conditional independence rules that could not possibly be true. This is one example of why it is inefficient.

The Posterior policy identifies an inference rule sequence that hones in on more probable model and eliminates the unnecessary rules. One inference rule sequence it can identify is:

$$CIRule(Y, [], Z) : F; CIRule(X, [], Z) : F; CIRule(X, [Y], Z) : T;$$

The Initial Probability for each model in the domain is shown in Table 8.2 [1].

Table 8.2. Probabilities of Models in a 3-variable Domain With an Uninformative Prior After Posterior Selects Inference Rules.

Model	Initial Probability	After 1st	After 2nd	After 3rd
(Z, Y)	0.037	0.04762	0.0	0.0
(Y, X)	0.037	0.0	0.0	0.0
(X, Y), (X, Z), (Z, Y)	0.037	0.04762	0.0556	0.0
(X, Y), (Z, X)	0.037	0.04762	0.0556	0.0
(Y, X), (Y, Z)	0.037	0.04762	0.0556	0.2
(X, Y), (Y, Z)	0.037	0.04762	0.0556	0.2
(X, Y), (X, Z)	0.037	0.04762	0.0556	0.0
(X, Y)	0.037	0.0	0.0	0.0
(Z, X)	0.037	0.0	0.0	0.0
(Y, X), (Z, X), (Z, Y)	0.037	0.04762	0.0556	0.0
(X, Z), (Z, Y)	0.037	0.04762	0.0556	0.0
(X, Y), (Z, Y)	0.037	0.04762	0.0	0.0
(Y, X), (Z, X)	0.037	0.04762	0.0	0.0
(Y, X), (Z, Y)	0.037	0.04762	0.0556	0.2
(X, Y), (X, Z), (Y, Z)	0.037	0.04762	0.0556	0.0
(X, Z), (Y, X)	0.037	0.04762	0.0556	0.0
(X, Z), (Y, X), (Y, Z)	0.037	0.04762	0.0556	0.0
(X, Z), (Y, Z)	0.037	0.04762	0.0556	0.0
(X, Y), (Z, X), (Z, Y)	0.037	0.04762	0.0556	0.0
(Y, Z)	0.037	0.04762	0.0	0.0
()	0.037	0.0	0.0	0.0
(Z, X), (Z, Y)	0.037	0.04762	0.0556	0.0
(Y, X), (Y, Z), (Z, X)	0.037	0.04762	0.0556	0.0
(X, Z)	0.037	0.0	0.0	0.0
(Y, Z), (Z, X)	0.037	0.04762	0.0556	0.0

[1]Probabilities are not adjusted to account for cycles.

Based on the Initial Probability, the rank of rules by the Posterior is shown under First Ranking in Table 8.3. When all models are equally probable, the ranking of rules by their probability of being True results in a higher probability of the marginal independence being True.

Table 8.3. The Ranking of Available Inference Rules by the Posterior Policy With an Uninformative Prior.

Inference Rule	First Ranking	Second Ranking	Third Ranking
CIRule(X, [], Y)	0.2222	0.1429	0.0556
CIRule(X, [], Z)	0.2222	0.1429	–
CIRule(Y, [], Z)	0.2222	–	–
CIRule(X, [Z], Y)	0.1111	0.1429	0.1667
CIRule(X, [Y], Z)	0.1111	0.1429	0.1667
CIRule(Y, [X], Z)	0.1111	0.1429	0.1667

Similar to the PC policy, the Posterior policy will choose from among the marginal independence rules at random. In the example inference rule sequence, the Posterior chooses $CIRule(Y, [], Z)$, applies it, identifies that it is False, and updates the probabilities on the model space. The updated model probabilities are in the After 1st column of Table 8.2. Since the marginal independence test results in False, six models are eliminated from the space.

After this update, there remain five marginal/conditional independence rules that could be True. The updated rule rankings are in the Second Ranking column of Table 8.3. With all five available rules having the same probability, the Posterior can choose among them at random and in the example sequence it chose rule $CIRule(X, [], Z)$. This rule evaluates to False on the model being inferred, eliminates another five models from the model space and the Posterior updates the probabilities on the model space (the After 2nd column in Table 8.2).

There are now four rules that could be True with the conditional independence tests ranked higher than the marginal independence test. The Posterior will then randomly choose from among the conditional independence rules because the marginal test has a lower probability of being True. In this example, the Posterior has chosen rule $CIRule(X, [Y], Z)$. This rule evaluates to True, eliminating all but three models from the model space, those models found in the Markov-equivalence class.

At this point the Posterior can stop because there is no other model, save for the three in the Markov-equivalence class, and among these three, the Posterior identifies that there are no further rules it can apply to distinguish them. This is one example of an inference rule sequence found by the Posterior. Randomization within the Posterior will generate different sequences on subsequent runs. Examples include:

$$\begin{gathered} CIRule(X, [], Y) : F; \\ CIRule(Y, [], Z) : F; \\ CIRule(Y, [X], Z) : F; \\ CIRule(X, [Z], Y) : F; \\ CIRule(X, [Y], Z) : T; \end{gathered}$$

$$\begin{gathered} CIRule(Y, [], Z) : F; \\ CIRule(X, [], Z) : F; \\ CIRule(X, [Y], Z) : T; \end{gathered}$$

$$\begin{gathered} CIRule(X, [], Y) : F; \\ CIRule(X, [], Z) : F; \\ CIRule(Y, [], Z) : F; \\ CIRule(X, [Z], Y) : F; \\ CIRule(X, [Y], Z) : T; \end{gathered}$$

$$CIRule(X, [], Y) : F;$$

$$CIRule(X, [], Z) : F;$$

$$CIRule(Y, [], Z) : F;$$

$$CIRule(X, [Y], Z) : T;$$

8.3.1.3 Computational Trade-Offs

Note that the decrease in expected cost that results from applying reasoning over the model space incurs a computational cost. While the Posterior policy finds shorter inference rule sequences than does the PC policy, it does so by taking a much longer time to compute. The implementation was in the Python programming language to facilitate ease in scripting and to facilitate integration into other programs and it is expected that there would be an improvement in speed if another language was used, for example Java.

Table 8.4. Expected Times in Seconds to Identify an Inference Rule Sequence Across 3- to 10-variables With an Uninformative Prior.

Domain	PC	Posterior
3-var	0.0019	0.0195
4-var	0.0090	0.3220
5-var	0.0701	12.9969
6-var	0.2265	2628.84
7-var	2.10	6852.2
8-var	13.35	14813
9-var	73.89	32144
10-var	367.08	71905

Table 8.4 shows the expected time in seconds to identify the inference rule sequence for a model in each domain. The Posterior, while identifying a much smaller inference rule sequence across the 10-var domain, does require nearly 19 hours per model to do so while the PC policy only requires 6 minutes.

There are certainly things that can be done to improve the computational complexity of an informed policy but one of the central points of this research is that informational policies lead to shorter inference rule sequences to uncover the Markov-equivalence class. Rules carry real-world costs that are both financial and time-consuming. The advantage of spending longer times deciding on the next action tend to outweigh the *apply everything* approach of the PC policy. When a policy is able to reason over more complex inference rules such as randomized experimental designs, the time and costs it saves from applying just a few will out perform a policy that applies every single experiment.

8.3.2 Performance with an Informative Prior

Results indicate that under an uninformative prior an informed search policy identifies a shorter inference rule sequence than an uninformed policy so the impact on performance of an informed search policy when given an advantage, an informative prior, was evaluated next. An informative prior is a prior that specifies a probability distribution across the model space in which models do not have a uniform distribution. Some models will have higher probabilities than others and under some informative priors, some models may have zero probability. A prior is defined as described in section 7.3.1. Each edge (X, Y) has one of three possible states with a probability on each state, $p(X \longrightarrow Y)$, $p(X \longleftarrow Y)$, $p(X \quad\quad Y)$. The total sum of the probability on an edge must add up to 1.0 and if one state is not possible it is assigned an edge probability of 0.0 effectively eliminating models from the model space for which the edge is present. An informative prior is further defined as any prior where a single edge has a non-uniform probability across the possible edge states.

8.3.2.1 Specification of the Informative Prior

Specification of an informative prior is not unreasonable. Researchers often have some intuition for potential causes and can describe these assessments in terms of altering the priors on the edges across the model space. Even if there is limited evidence that one edge state is more probable than another, this informative prior could be useful to guide a causal search policy. Because a probability of 0.0 on an edge state removes the state from consideration by an informed search policy, for most cases specifying an incredibly small probability such as $p = 1e^{-7}$ should suffice. The small probability of the edge state allows those models that have that edge state to still be found but they need not initially be a part of the set M used to guide rule ranking. Those models with different states on the same edge have a higher probability and applying a rule that would be True on that edge state might eliminate those unlikely models anyway. If however what was thought as impossible was actually possible, then the posterior would never be adjusted to reflect that and the informed search policy will never find those unexpected models.

For evaluation, the informative prior chosen during an experiment was used both for sampling and as input into the Posterior policy. This does not mean that the Posterior has knowledge of which model in the model space was chosen. It only has the prior over the model space in which the chosen model is generated from and can use this prior to rank rules and update the model space probabilities based on what it learns. Having one prior from which to draw models and another, which includes a different probability distribution across the model space including the possibility of a different set of models, is an evaluation of policy performance under a misinformed prior that is evaluated in section 8.3.3.

The experiment is limited to a maximum of ten edge probabilities and to the 3-, 4-, and 5-variable domain sizes. There are a large number of combinations that can be used for an informative prior. Because the focus was on simply determining if an informed policy could out perform an uninformed search policy, one and only one informative prior was chosen and used to determine performance. For each edge (X, Y) in a domain, a probability was picked at random for each edge state.

For the 3-variable domain, performance was tested under zero edges (the uninformative prior), 1-, 2-, and 3-edges. For the 1-edge prior a prior was tested consisting of probabilities on a single edge (X, Y). When evaluating a 2-edge prior the probabilities on the edge (X, Y) were kept and probabilities on a second edge (X, Z) were added. Under a 3-edge prior, edge probabilities for (Y, Z) were added. In this manner the prior becomes more and more specific so evaluation of performance as a function of specificity can also be done.

Table 8.5 shows the informative prior of 10-edge states where the order of the edge states are: $p(X \longrightarrow Y)$, $p(X \longleftarrow Y)$, $p(X \quad Y)$.

Table 8.5. The Informative Prior Used for Evaluation.

Edge Index	Edge	$p(X \longrightarrow Y)$	$p(X \longleftarrow Y)$	$p(X \quad Y)$
1	(X,Y)	0.8	0.2	0.0
2	(X,Z)	0.6	0.2	0.2
3	(Y,Z)	0.2	0.0	0.8
4	(X,W)	0.1	0.1	0.8
5	(Y,W)	0.7	0.3	0.0
6	(Z,W)	0.9	0.0	0.1
7	(X,A)	0.3	0.6	0.1
8	(Y,A)	0.7	0.1	0.2
9	(Z,A)	0.5	0.3	0.2
10	(W,A)	0.5	0.5	0.0

As an example of reading the table, the prior specifies that the probability of X being causal for Y is 0.8, the probability that Y is causal for X is 0.2 and that it must be that X and Y are involved in a causal dependence. Zero probabilities states were allowed to evaluate performance when models are eliminated by the prior under the assumption that the prior is informative and not misleading.

8.3.2.2 Performance Evaluation

Figure 8.14 plots the expected cost for the PC, Posterior, and Optimal policies under an informative prior for 3- to 5-variable domains.

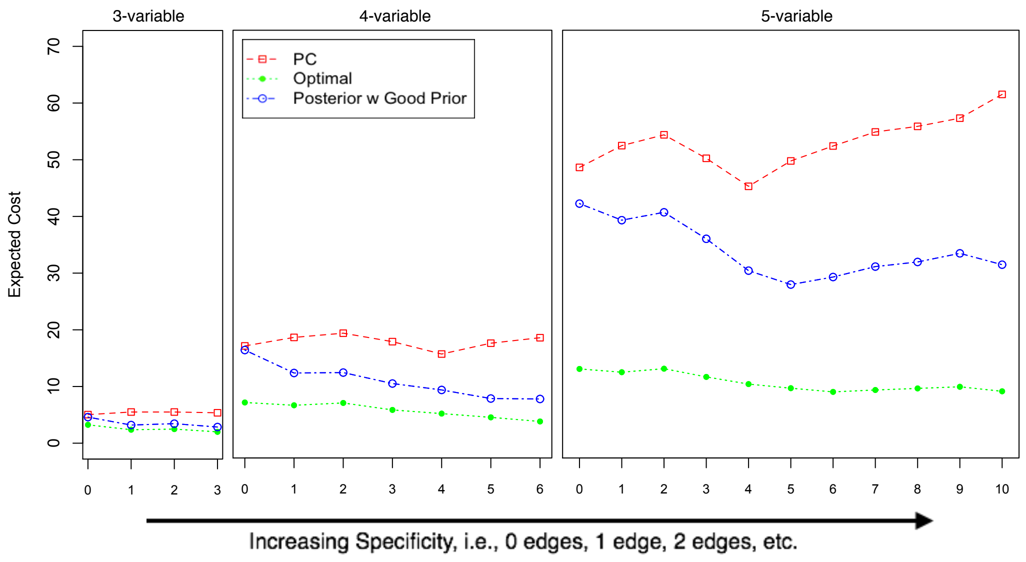


Figure 8.14. Comparison of PC, Posterior, and Optimal Causal Policies on an Informative Prior.

As with the uninformative prior, the informed Posterior policy outperforms the uninformed PC policy. Note that the Optimal policy still knows the model for each iteration but given the prior over the model space it has fewer models it must eliminate to prove it has this model.

The performance characteristic for the Posterior is such that as the prior is more specific over the model space, the expected cost of the inference rule sequence decreases. The sample M used by the Posterior for ranking is drawn from this prior leading to rules being ranked higher that are more likely to hold true over the model space defined by the informative prior. As these rules are applied, the Posterior updates the prior and allowing it to sample higher probable models from the model space. This in turn leads to uncovering the Markov-equivalence class faster and in fewer inferences.

The prior we've chosen itself leads to some interesting behavior in the expected costs across the PC policy in that certain edge probabilities actually make it easier for the PC policy while others make it harder. This is because the third edge probability:

$$p(Y, Z) = [0.2,\ 0.0,\ 0.8]$$

and the fourth edge probability:

$$p(X, W) = [0.1,\ 0.1,\ 0.8]$$

are examples where its more likely that no causal dependence exists between two variables. For the PC policy, it will attempt marginal independence rules first and when 80% of the time there is no dependence between them, there is a greater chance that marginal independence will identify the situation leading to smaller expected costs. But the PC policy still suffers from an increasing expected cost as the prior includes probabilities on edge states where causal dependencies do hold. When edge probabilities specify that dependencies between edges are more likely to exist then it requires more than marginal independence rules and potentially conditional independence rules of large separating set sizes before PC can identify the Markov-equivalence class. In those situations PC will require longer and longer inference rule sequences.

The distribution of percent improvements are shown in Figures 8.15-8.19. Each possible model in the domain was iterated a number of times. For the 3- and 4-variable domains there are 500 iterations per model. For the 5-variable domain, there are 30 iterations per model. For each model-iteration, the percent improvement is computed as:

$$100.0 * (\text{Posterior cost - PC cost})/\text{PC cost}$$

The distributions are then the density across the percent improvements for each domain and within each domain, for the number of edges with a probability. Note that the distributions become smoother as the number of models in the domain increases.

In all cases the Posterior has a noticeable percent improvement over PC with the percent growing as the prior becomes more specific. The percentage improvement of the Posterior over the PC grows as the domain grows as expected because as the prior becomes more specific, the Posterior makes a more informed decision on which rule to apply next. At a prior across all 10-edges in the 5-variable domain, a percent improvement of 43.8% is seen. But this is the percent improvement for one particular prior. A more thorough investigation across a large number of potential priors would be needed in order to ascertain the true percentage improvement one could hope to get over PC with a fully-specified prior.

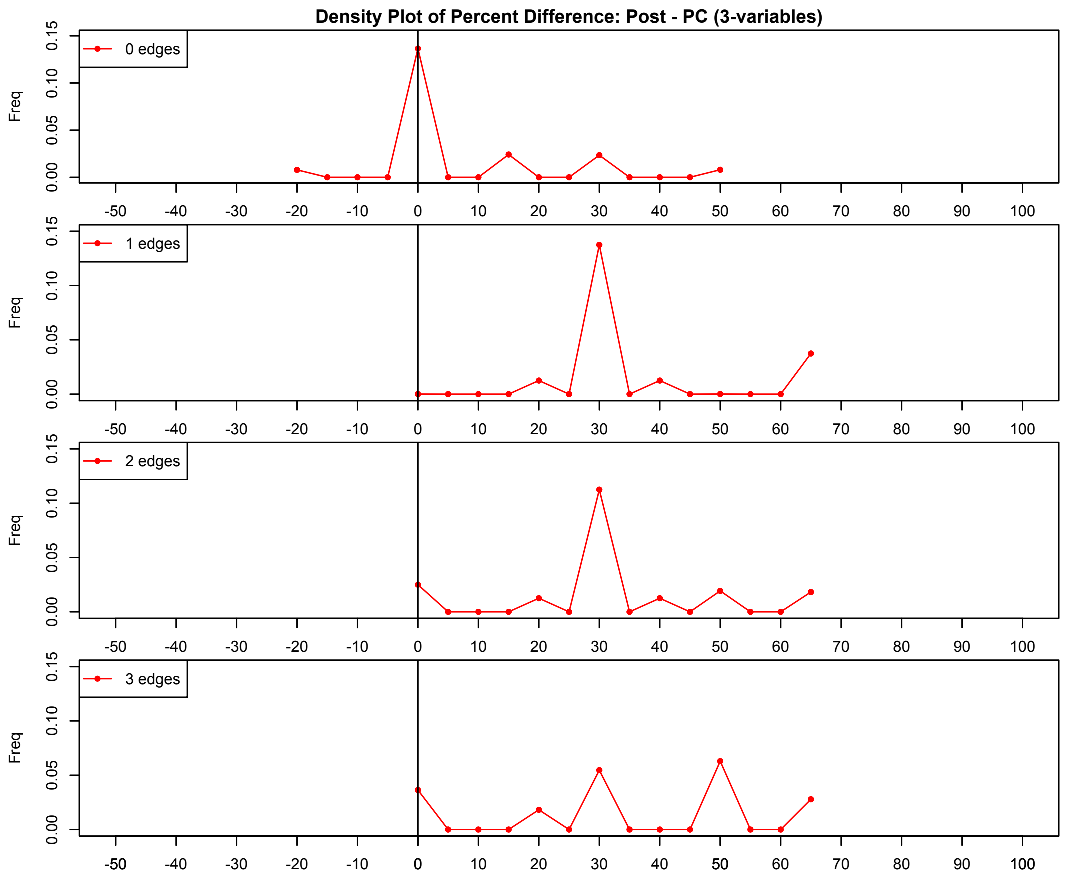


Figure 8.15. Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 3-variable Domain.

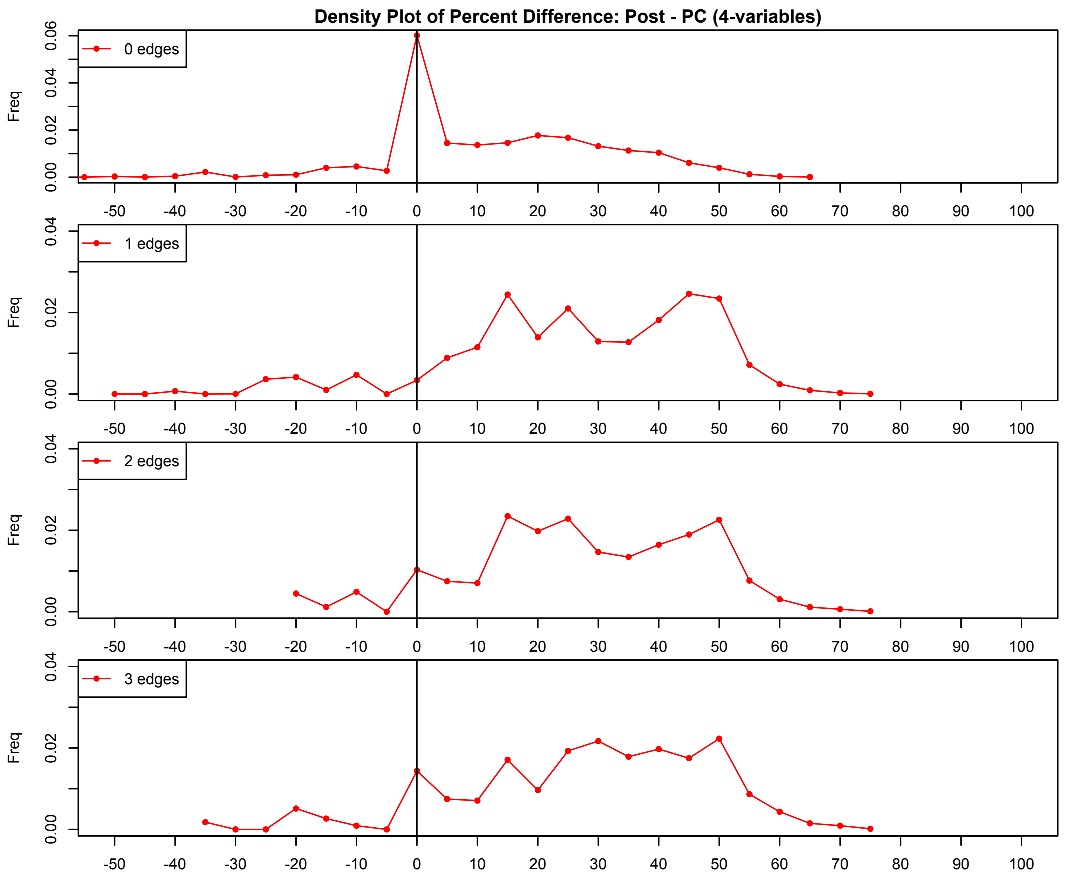


Figure 8.16. Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 4-variable Domain, 0- to 3-edges.

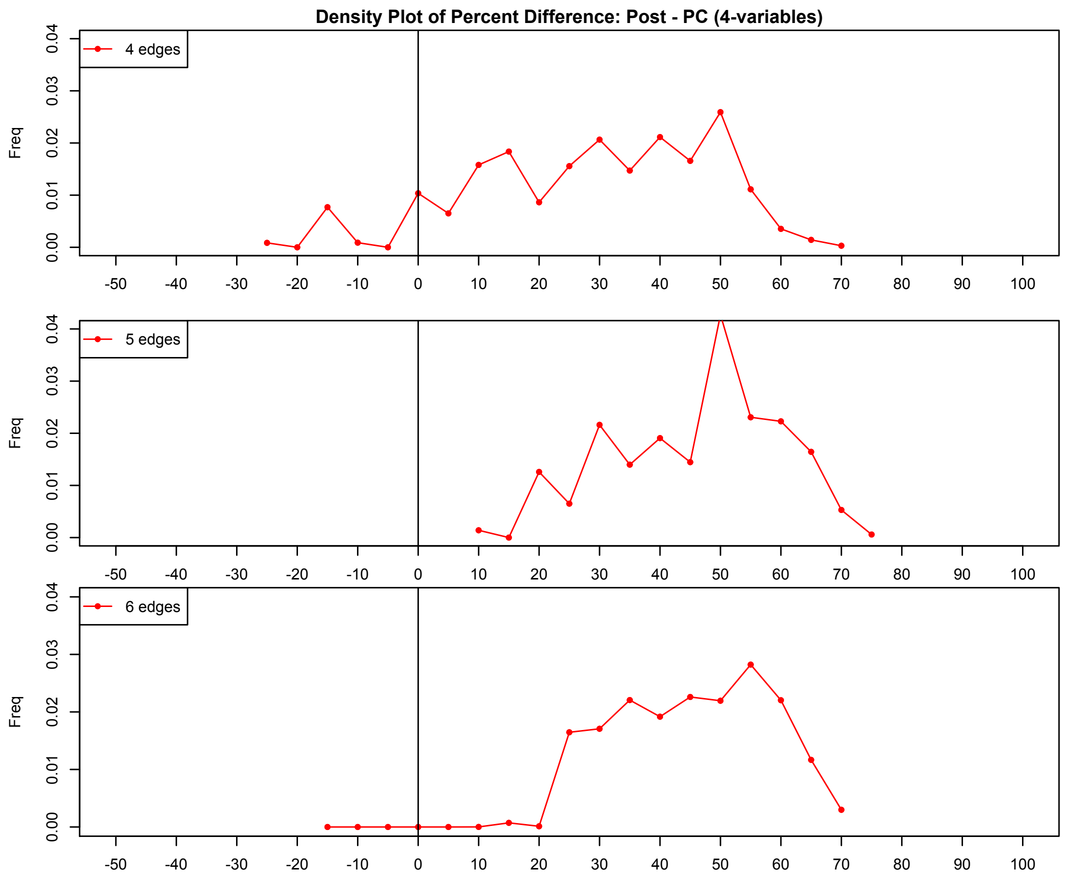


Figure 8.17. Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 4-variable Domain, 4- to 6-edges.

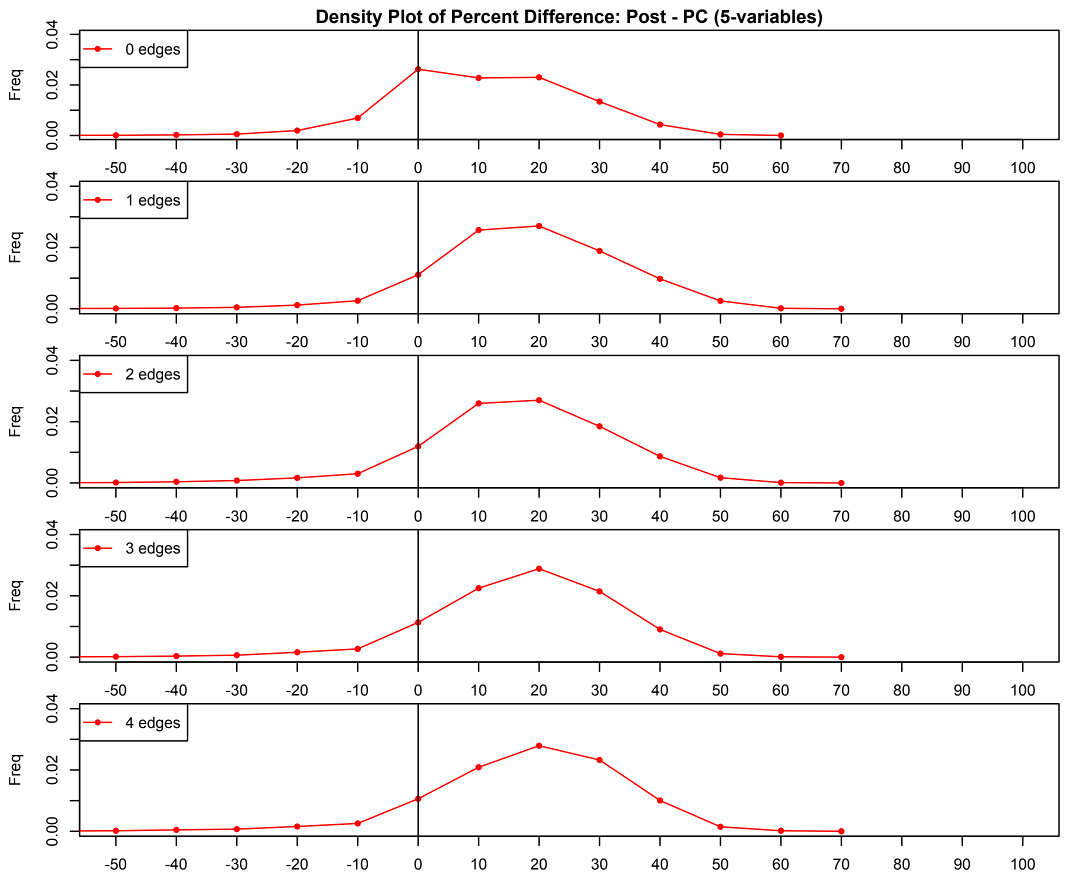


Figure 8.18. Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 5-variable Domain, 0- to 4-edges.

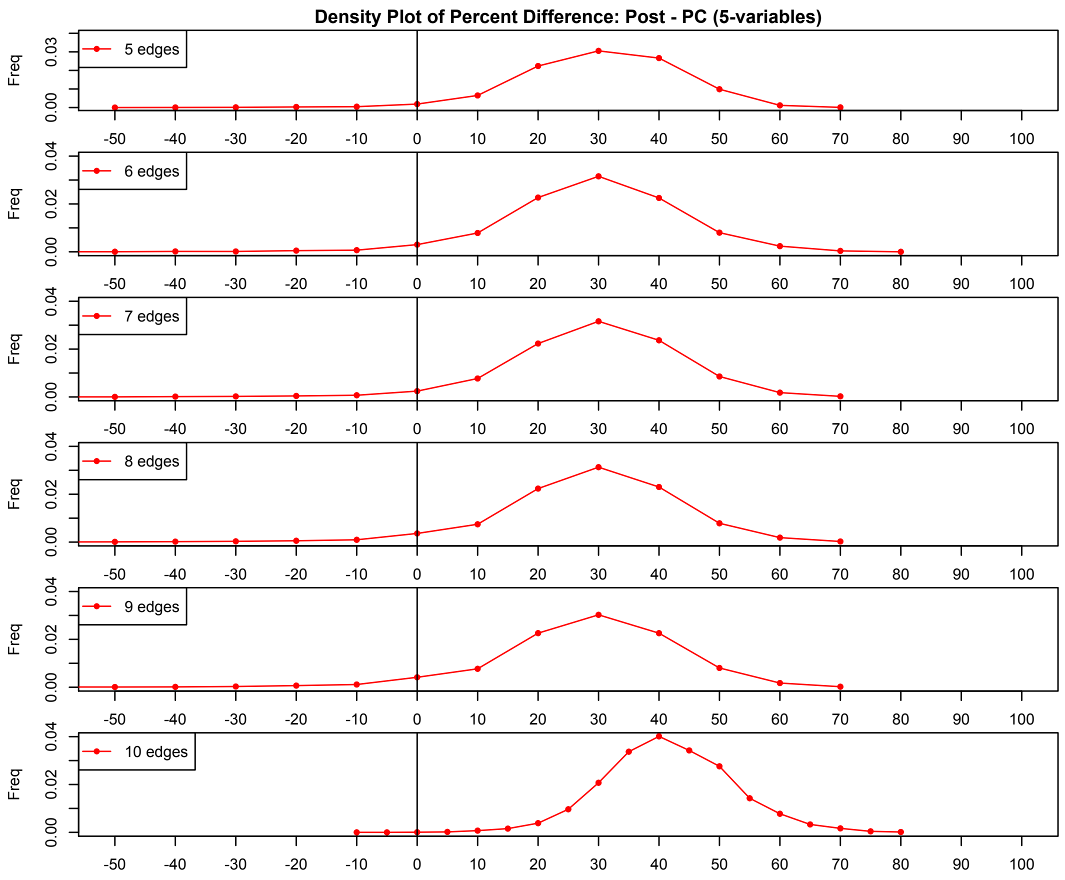


Figure 8.19. Percent Improvement of the Posterior Over the PC Policy With Informative Prior on a 5-variable Domain, 5- to 10-edges.

8.3.2.3 Example of the Posterior Inference Rule Sequence Under an Informative Prior

Continuing the example from the Uninformed Prior section, performance is again demonstrated on the model shown in Figure 8.20 where Y is causal for both X and Z.

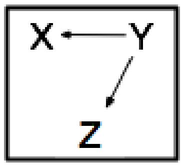


Figure 8.20. Example Model to Demonstrate the Inference Rule Sequence Found by the Posterior Policy When Given an Informative Prior.

Under the informative prior on 3-edges in a 3-variable domain, the prior is as follows:

$$p(X,Y) = [0.8,\ 0.2,\ 0.0]\ p(X,Z) = [0.6,\ 0.2,\ 0.2]\ p(Y,Z) = [0.2,\ 0.0,\ 0.8]$$

Which gives a probability for this model as 0.008.

$$p(X \longleftarrow Y) * p(Y \longrightarrow Z) * P(X \qquad Z) = 0.2 * 0.2 * 0.2 = 0.008$$

This probability does not take into account cycles and normalizing the probability of the model by removing cycles from the model space. The computation of the probability on the model space could, but model and inference rule comparisons would remain the same as normalization will never differentiate between two probabilities that were equal prior to normalization. The Initial Probability column in Table 8.12 identifies the probabilities of the models in the model space.

The PC policy continues as it did under an uninformative prior as the extra information on the model space is unused:

Table 8.6. Probabilities of Models in a 3-variable Domain With an Informative Prior After Posterior Selects Inference Rules.

Model	Initial Probability	After 1st	After 2nd	After 3rd
(Z, Y)	0.0	0.0	0.0	0.0
(Y, X)	0.032	0.0816	0.0	0.0
(X, Y), (X, Z), (Z, Y)	0.0	0.0	0.0	0.0
(X, Y), (Z, X)	0.128	0.0	0.0	0.0
(Y, X), (Y, Z)	0.008	0.0204	0.04	0.1111
(X, Y), (Y, Z)	0.032	0.0816	0.16	0.4444
(X, Y), (X, Z)	0.384	0.0	0.0	0.0
(X, Y)	0.128	0.3265	0.0	0.0
(Z, X)	0.0	0.0	0.0	0.0
(Y, X), (Z, X), (Z, Y)	0.0	0.0	0.0	0.0
(X, Z), (Z, Y)	0.0	0.0	0.0	0.0
(X, Y), (Z, Y)	0.0	0.0	0.0	0.0
(Y, X), (Z, X)	0.032	0.0816	0.0816	0.0
(Y, X), (Z, Y)	0.0	0.0	0.0	0.0
(X, Y), (X, Z), (Y, Z)	0.096	0.2449	0.48	0.0
(X, Z), (Y, X)	0.096	0.0	0.0	0.0
(X, Z), (Y, X), (Y, Z)	0.024	0.0612	0.12	0.0
(X, Z), (Y, Z)	0.0	0.0	0.0	0.0
(X, Y), (Z, X), (Z, Y)	0.0	0.0	0.0	0.0
(Y, Z)	0.0	0.0	0.0	0.0
()	0.0	0.0	0.0	0.0
(Z, X), (Z, Y)	0.0	0.0	0.0	0.0
(Y, X), (Y, Z), (Z, X)	0.008	0.02041	0.04	0.0
(X, Z)	0.0	0.0	0.0	0.0
(Y, Z), (Z, X)	0.0	0.0	0.0	0.0

$$CIRule(X,[],Z):F;\ CIRule(X,[],Y):F;\ CIRule(Y,[],Z):F;\ CIRule(X,[Y],Z):T;$$
$$CIRule(Y,[X],Z):F;\ CIRule(X,[Z],Y):F;$$

and the PC policy identifies the Markov-equivalence class once again.

However the Posterior policy relies on the prior as a guide using the probabilities computed in Table 8.12 as a guide. Within this set of models given the probabilities, the initial ranking of the marginal/conditional independence rules are in the First Ranking column in Table 8.7.

Table 8.7. The Ranking of Available Inference Rules by the Posterior Policy With an Informative Prior.

Inference Rule	First Ranking	Second Ranking	Third Ranking
CIRule(X, [], Y)	0.0	0.0	0.0
CIRule(X, [], Z)	0.16	0.4082	0.0
CIRule(Y, [], Z)	0.192	0.4898	–
CIRule(X, [Z], Y)	0.0	0.0	0.0
CIRule(X, [Y], Z)	0.04	0.1020	0.2
CIRule(Y, [X], Z)	0.608	–	–

Note that rules $CIRule(X,[],Y)$ and $CIRule(X,[Z],Y)$ have a zero rank because the prior on the model space specifies that X and Y are part of a causal dependence. For this reason, they are effectively removed from consideration by the Posterior policy.

To identify this model, the most beneficial inference rule to the Posterior is: $CIRule(Y,[X],Z)$ with a rank of 0.608. This rule returns False given the model. The Posterior then updates the probabilities on the model space, the After 1st column in Table 8.12, and re-ranks the available inference rules, the Second Ranking column in Table 8.7.

The next highest probable rule is: $CIRule(Y,[],Z)$ with a rank of 0.4898 that evaluates to False. After updating the model space with this information, the model probabilities are shown in the After 2nd column in Table 8.12 and across these models only one rule remains that could possibly be True: $CIRule(X, [Y], Z)$. Evaluation of this rule results in True and narrows down the model space to what is shown in the After 3rd column in Table 8.12.

As there are no more inference rules that can be applied and the two remaining models fit within the Markov-equivalence class, the Posterior is finished. Its final inference rule sequence is:

CIRule(Y, [X], Z):F; CIRule(Y, [], Z):F; CIRule(X, [Y], Z):F;

The Posterior can choose rules at random that have the same ranking. However with this prior there were no two rules with the same ranking after applying any rule in the final sequence, so the Posterior always identifies this same inference rule sequence for this model given this prior.

8.3.2.4 Computational Trade-Offs

Since an informative prior changes the probability over the model space, evaluation of the impact on timing for the Posterior policy is included. Table 8.8 shows the time in seconds for both the PC and the Posterior policies across the informative prior.

As the prior becomes more specific the computational timing requirements of an informed policy decreases. Part of the improvement in timing is due to the informed policy being able to identify shorter inference rule sequences. The informative prior is guiding it such that it is able to identify the rules more likely to be True after than it does under an uninformative prior. Additionally, because the informative prior is used as a guide, the models that an informed policy considers for the set M becomes smaller when an edge state has a zero probability because that eliminates up to a third of the potential models that must be considered. For example on 3-edges, the informative prior eliminates the probability that Z is causal for Y and a subsequent decrease in expected time occurs. Likewise on 6-edges the informative prior specifies that W is not causal for Z and another decrease in timing occurs.

Table 8.8. Expected Times in Seconds to Identify an Inference Rule Sequence Across 3- to 5-variables With an Informative Prior.

Domain	Edges	PC	Posterior
3	0-edges	0.0013	0.0195
3	1-edges	0.0014	0.0198
3	2-edges	0.0015	0.0214
3	3-edges	0.0016	0.0130
4	0-edges	0.0091	0.3220
4	1-edges	0.0099	0.3044
4	2-edges	0.0098	0.3259
4	3-edges	0.0096	0.2587
4	4-edges	0.0095	0.2763
4	5-edges	0.0094	0.2715
4	6-edges	0.0095	0.4865
5	0-edges	0.0701	12.7453
5	1-edges	0.0725	4.1229
5	2-edges	0.0626	4.5821
5	3-edges	0.0615	6.3945
5	4-edges	0.0628	6.2981
5	5-edges	0.0672	5.0806
5	6-edges	0.0649	3.7696
5	7-edges	0.0658	3.8425
5	8-edges	0.0643	3.7981
5	9-edges	0.0645	3.8931
5	10-edges	0.0671	3.6866

8.3.3 Performance with a Misinformative Prior

Researchers can sometimes be wrong about their intuition of possible causal relationships so a very small evaluation was conducted on performance when the informative prior misspecifies the probabilities of the causal models. For example the researcher may believe that it is more probable that X is directly causal for Y when in fact, X and Y are marginally independent. We refer to this as a misinformative prior.

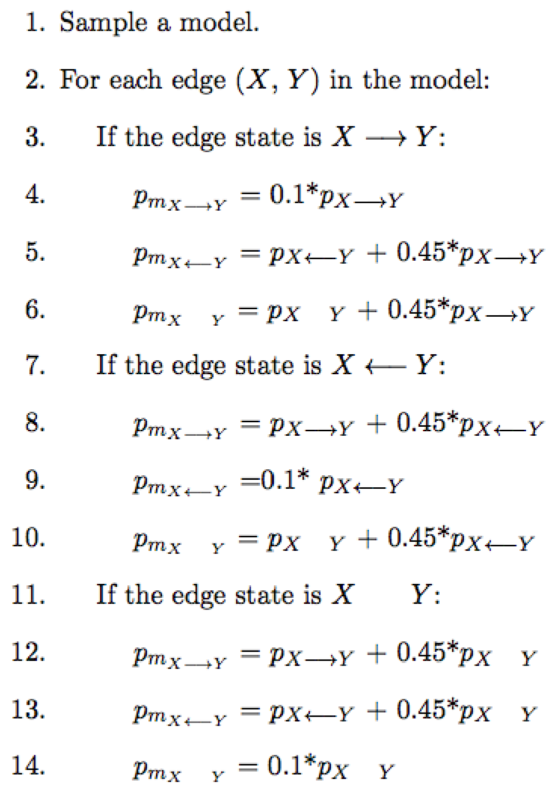
Because an informed policy updates the probability on the model space after it applies an inference rule, then the policy should be able to correct itself after some number of rules are applied. While wrong initially, the policy, through process of inference and update, should be able to overcome the initial misinformative prior.

We do not consider misinformative priors that incorrectly specify 0.0 or 1.0 probabilities on an edge state as an informed policy would incorrectly eliminate or include models. After exhausting all inference with this kind of misinformative prior, the policy would be left with an empty set indicating that there was no valid model. Edge state probabilities can be specified to be very small or very large, but should never be 0.0 or 1.0. With such a prior, an informed policy would still be able to identify the Markov-equivalence class.

8.3.3.1 Specification of the Misinformative Prior

The complexity of the definition of a misinformed prior makes it very difficult to determine complete performance under all poorly specified priors, so the evaluation was limited to what happens when the prior specifies that the correct generative model has a much lower probability than many alternative models. If the generative model is assigned a low probability by the prior, then this leads the informed policy to select a pattern of inferences that may be wasteful by applying inference rules that are meant to identify other models.

The sampling procedure is thus to use the original informative prior, p, from which to sample models. For each model that is sampled, the edges in the sampled model are used to construct the misinformative prior. If the edge $X \longrightarrow Y$ is in the model, then the probability of $X \longrightarrow Y$ in the original informative prior, $p_{X \longrightarrow Y}$ is reduced by 90%. The new edge state probability then is $0.1 * p_{X \longrightarrow Y}$. The other edge states then split the difference. Thus, the misinformative prior, p_m is defined in Figure 8.21.



```
1.  Sample a model.
2.  For each edge $(X, Y)$ in the model:
3.      If the edge state is $X \longrightarrow Y$:
4.          $p_{m_{X \longrightarrow Y}} = 0.1 * p_{X \longrightarrow Y}$
5.          $p_{m_{X \longleftarrow Y}} = p_{X \longleftarrow Y} + 0.45 * p_{X \longrightarrow Y}$
6.          $p_{m_{X \quad Y}} = p_{X \quad Y} + 0.45 * p_{X \longrightarrow Y}$
7.      If the edge state is $X \longleftarrow Y$:
8.          $p_{m_{X \longrightarrow Y}} = p_{X \longrightarrow Y} + 0.45 * p_{X \longleftarrow Y}$
9.          $p_{m_{X \longleftarrow Y}} = 0.1 * p_{X \longleftarrow Y}$
10.         $p_{m_{X \quad Y}} = p_{X \quad Y} + 0.45 * p_{X \longleftarrow Y}$
11.     If the edge state is $X \quad Y$:
12.         $p_{m_{X \longrightarrow Y}} = p_{X \longrightarrow Y} + 0.45 * p_{X \quad Y}$
13.         $p_{m_{X \longleftarrow Y}} = p_{X \longleftarrow Y} + 0.45 * p_{X \quad Y}$
14.         $p_{m_{X \quad Y}} = 0.1 * p_{X \quad Y}$
```

Figure 8.21. Specification of the Misinformative Prior.

8.3.3.2 Performance Evaluation

Figure 8.22 shows the performance of the Posterior under the misinformative prior. For comparison the performance of the PC, Optimal, and Posterior policies are plotted under an informative prior.

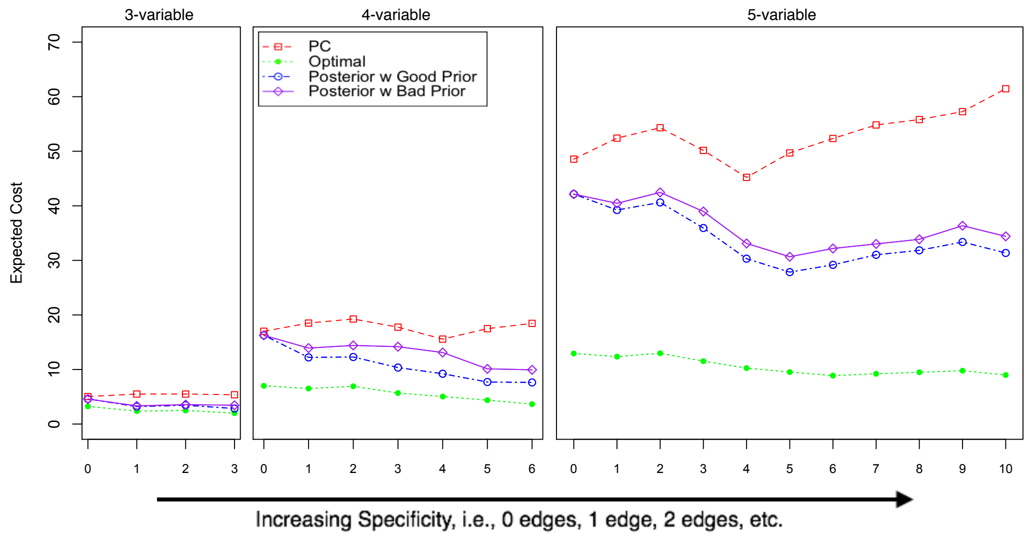


Figure 8.22. Comparison of PC, Posterior, and Optimal Policies Against the Posterior on a Misinformative Prior.

The results indicate that in general the expected cost from the inference rule sequence found by the Posterior is a little worse than it would be under an informative prior but it is not a large difference. In fact under some conditions the misinformative prior guides the Posterior to identify a shorter inference rule sequence than it does under the good prior. This improvement, which occurs in the 5-variable domain for a prior across 6-edges to 10-edges, is most likely a fortuitous situation that results from the way this particular misinformative prior has been specified. It is unlikely that these results imply that poorly specified priors will in general yield decreased expected costs.

The distribution of percent improvements of the Posterior over the PC policy are shown in Figures 8.23-8.27.

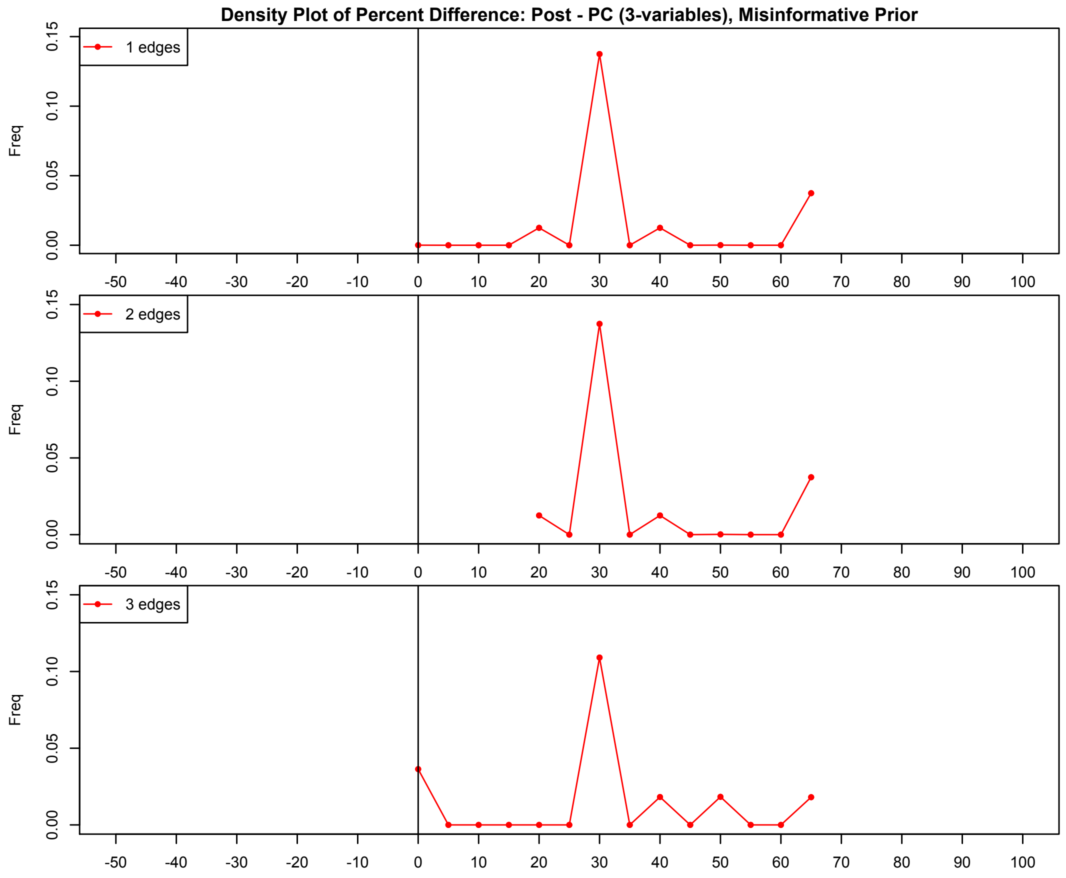


Figure 8.23. Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 3-variable Domain.

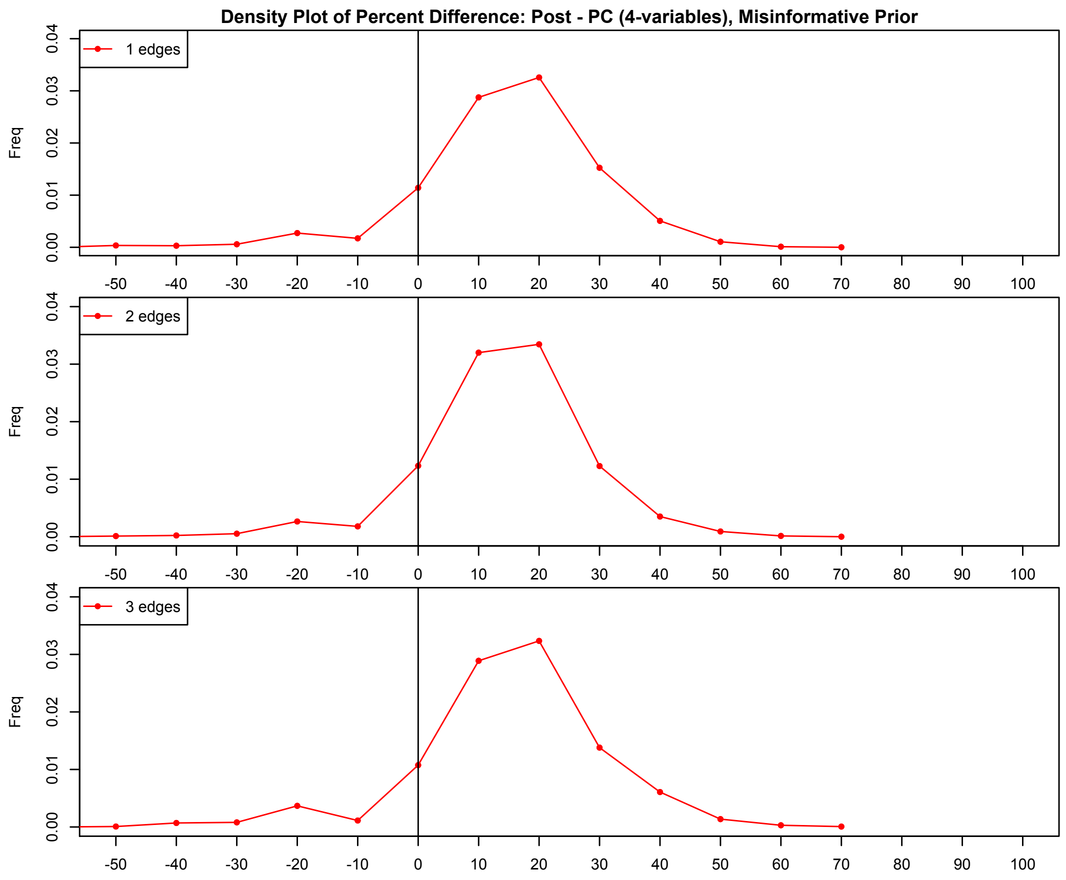


Figure 8.24. Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 4-variable Domain, 1- to 3-edges.

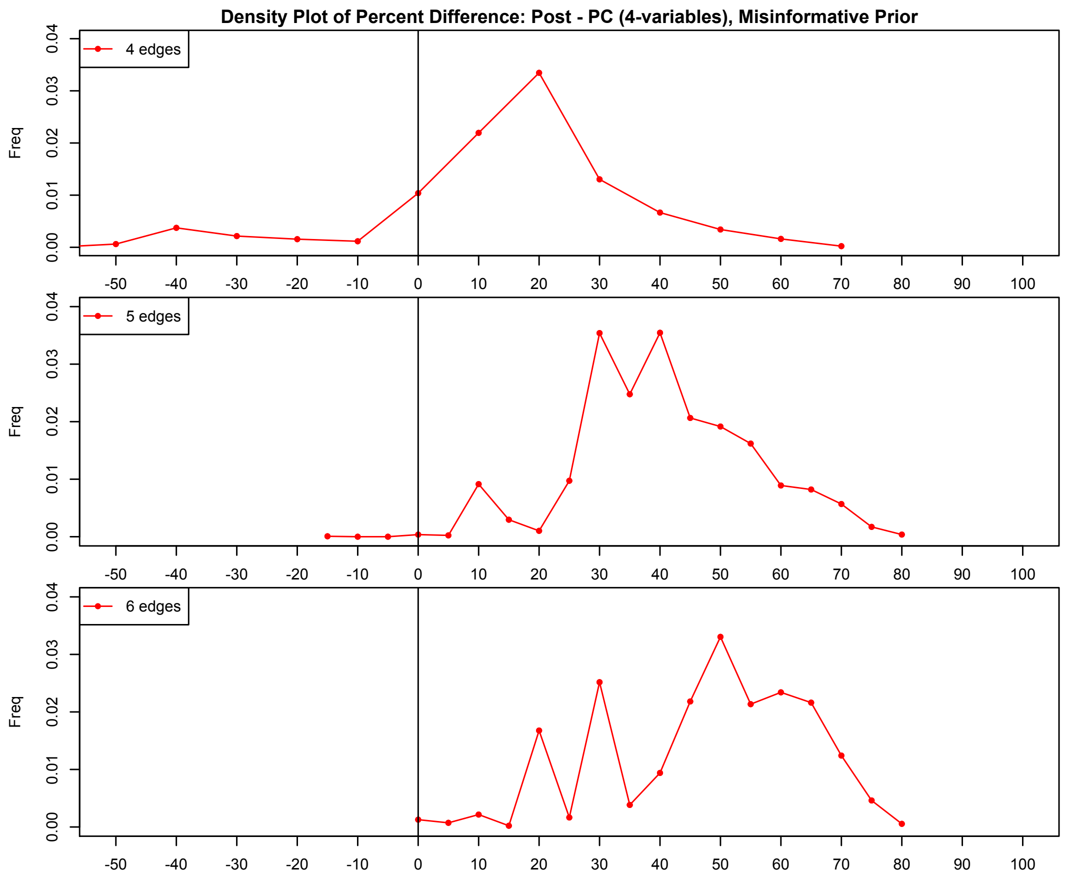


Figure 8.25. Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 4-variable Domain, 4- to 6-edges.

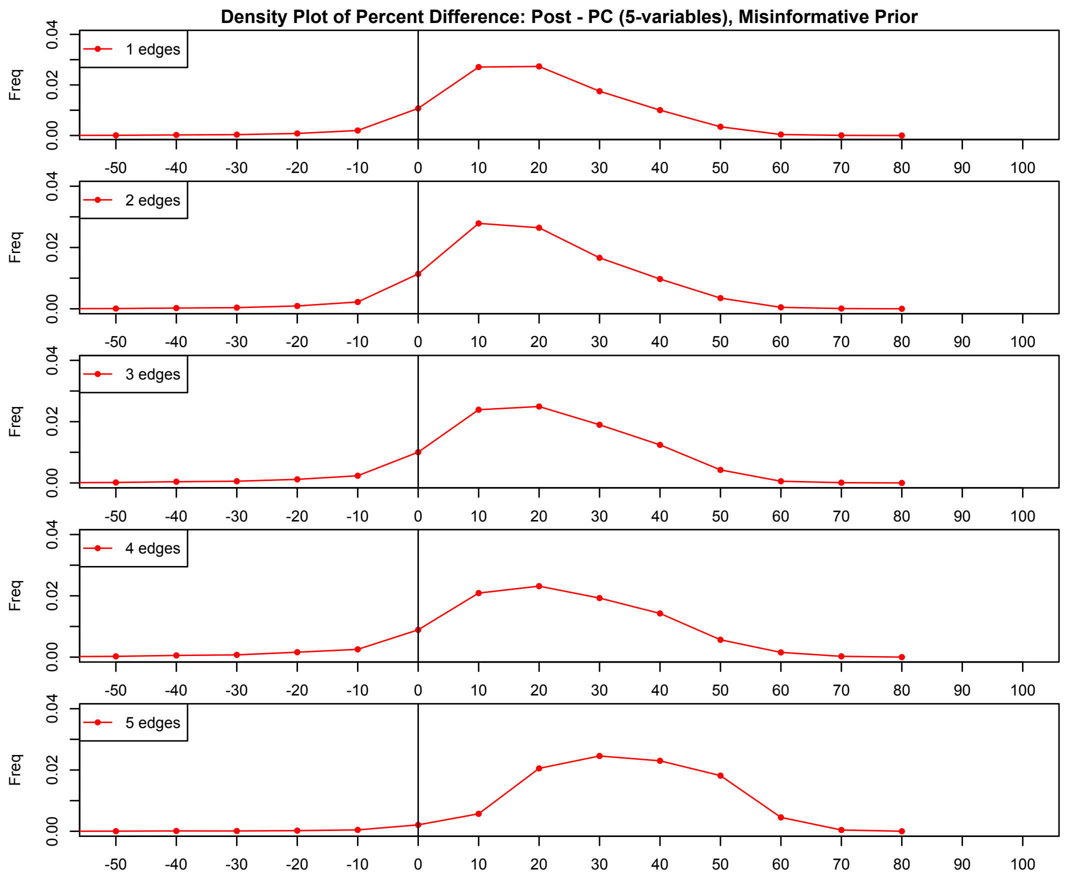


Figure 8.26. Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 5-variable Domain, 1- to 5-edges.

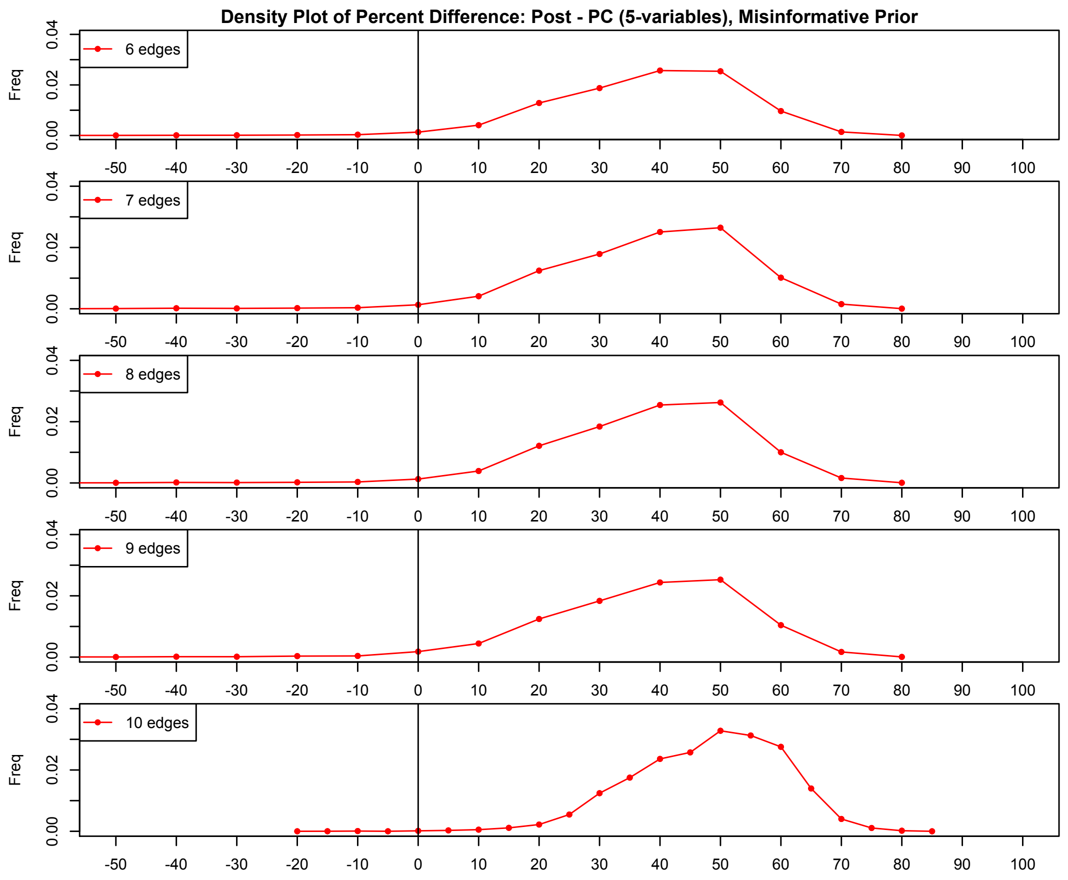


Figure 8.27. Percent Improvement of the Posterior Over the PC Policy With Misinformative Prior on a 5-variable Domain, 6- to 10-edges.

The results indicate that a wide range of priors, even ones which get some part of the model space probabilities incorrect, can lead to an informed search policy finding a shorter inference rule sequence than an uninformed policy. And as shown with the performance under an informative prior, the percent improvement of the Posterior over the PC policy grows as a function of domain size and the size of the specified prior.

8.3.3.3 Example of the Posterior Inference Rule Sequence Under an Misinformative Prior

An example helps to illustrate how a misinformative prior can still aid learning. For example, consider the model $((X, Y), (X, Z))$ as an example of the way a misinformative prior initially guides the Posterior and how the Posterior can recover from the poorly specified prior.

Assume the informative prior is used to sample models but that the informed policy relies on the misinformative prior specified in Table 8.9. The corresponding probabilities across the model space for models with non-zero probabilities are shown in Table 8.10.

Table 8.9. The Misinformative Prior Used for Discussion.

Edge Index	Edge	Informative Prior			Mis-informative Prior		
		$p(X \to Y)$	$p(X \leftarrow Y)$	$p(X \quad Y)$	$p(X \to Y)$	$p(X \leftarrow Y)$	$p(X \quad Y)$
1	(X,Y)	0.8	0.2	0.0	0.2	0.8	0.0
2	(X,Z)	0.6	0.2	0.2	0.2	0.6	0.2
3	(Y,Z)	0.2	0.0	0.8	0.8	0.0	0.2

The original informative prior is included as a comparison between the probabilities defined across the model space. For the example model $((X, Y), (X, Z))$, the misinformative prior gives a probability of 0.008 whereas the informative prior gives a probability of 0.384. The consequence of this is that the Posterior will initially presume the rules that lead to

Table 8.10. Probabilities of Models in a 3-variable Domain With a Misinformative Prior.

Model	Probability (Informative Prior)	Probability (Misinformative Prior)
(X, Y)	0.128	0.008
((X, Y), (X, Z))	0.384	0.008
((X, Y), (Y, Z))	0.032	0.032
((X, Y), (Z, X))	0.128	0.024
((X, Y), (X, Z), (Y, Z))	0.096	0.032
((X, Z), (Y, X))	0.096	0.032
((X, Z), (Y, X), (Y, Z))	0.024	0.128
(Y, X)	0.032	0.032
((Y, X), (Y, Z))	0.008	0.128
((Y, X), (Z, X))	0.032	0.096
((Y, X), (Y, Z), (Z, X))	0.008	0.384

discovering this model should rank lower than other rules and subsequently will apply a poorer choice first.

Given the probabilities on the model space from the misinformative prior, the non-zero ranking of the marginal/conditional independence rules are shown under column First Ranking in Table 8.11.

Table 8.11. The Ranking of Available Inference Rules by the Posterior Policy With a Misinformative Prior.

Inference Rule	First Ranking	Second Ranking	Third Ranking
CIRule(X, [], Y)	0.0	0.0	0.0
CIRule(X, [], Z)	0.04	0.0476	0.0
CIRule(Y, [], Z)	0.136	0.1619	–
CIRule(X, [Z], Y)	0.0	0.0	0.0
CIRule(X, [Y], Z)	0.16	–	–
CIRule(Y, [X], Z)	0.064	0.0762	0.0909

The Posterior selects the highest ranking rule which is:

$$CIRule(X, [Y], Z) \text{ - rank: } 0.16$$

That returns False given the model. The Posterior then updates the probabilities on the model space to those found in column After 1st in Table 8.12. Model $((X, Y), (X, Z))$ has increased in probability but is still not the most likely.

Table 8.12. Probabilities of Models in a 3-variable Domain With a Misinformative Prior After Posterior Selects Inference Rules.

Model	Initial Probability	After 1st	After 2nd	After 3rd
(Z, Y)	0.0	0.0	0.0	0.0
(Y, X)	0.0	0.0381	0.0	0.0
(X, Y), (X, Z), (Z, Y)	0.0	0.0	0.0	0.0
(X, Y), (Z, X)	0.0	0.0286	0.0341	0.15
(Y, X), (Y, Z)	0.0	0.128	0.0	0.0
(X, Y), (Y, Z)	0.0	0.0	0.0	0.0
(X, Y), (X, Z)	0.0	0.0095	0.0114	0.05
(X, Y)	0.0	0.0095	0.0	0.0
(Z, X)	0.0	0.0	0.0	0.0
(Y, X), (Z, X), (Z, Y)	0.0	0.0	0.0	0.0
(X, Z), (Z, Y)	0.0	0.0	0.0	0.0
(X, Y), (Z, Y)	0.0	0.0	0.0	0.0
(Y, X), (Z, X)	0.0	0.1143	0.0	0.0
(Y, X), (Z, Y)	0.0	0.0	0.0	0.0
(X, Y), (X, Z), (Y, Z)	0.0	0.0381	0.0455	0.0
(X, Z), (Y, X)	0.0	0.0381	0.0455	0.2
(X, Z), (Y, X), (Y, Z)	0.0	0.1524	0.1818	0.0
(X, Z), (Y, Z)	0.0	0.0	0.0	0.0
(X, Y), (Z, X), (Z, Y)	0.0	0.0	0.0	0.0
(Y, Z)	0.0	0.0	0.0	0.0
()	0.0	0.0	0.0	0.0
(Z, X), (Z, Y)	0.0	0.0	0.0	0.0
(Y, X), (Y, Z), (Z, X)	0.0	0.4571	0.5455	0.0
(X, Z)	0.0	0.0	0.0	0.0
(Y, Z), (Z, X)	0.0	0.0	0.0	0.0

With the updated probabilities the Posterior policy re-ranks the available rules to those shown in column Second Ranking in Table 8.11. The next highest probable rule is $CIRule(Y, [], Z)$ testing if Y and Z are marginally independent. They are in fact not marginally independent and the new model space probabilities are updated to those in column After 2nd in Table 8.12.

Across these models there is only one rule remaining that could possibly be True (column Third Ranking, Table 8.11.) Since this rule is True in the model ((X, Y), (X, Z)), the model space PMF is shown in column After 3rd in Table 8.12. The non-zero probable models identifies the Markov-equivalence class in which the model is a member.

The final inference rule sequence for the Posterior is:

$$CIRule(X, Z, [Y]) : F;\ CIRule(Y, Z, []) : F;\ CIRule(Y, Z, [X]) : T;$$

Compare this sequence with the two sequences that could be identified by the Posterior under the informative prior:

$$CIRule(Y, Z, [X]) : T;\ CIRule(Y, Z, []) : F;$$

$$CIRule(Y, Z, [X]) : T;\ CIRule(X, Z, []) : F;$$

In both cases, the Posterior identifies the model with an inference rule sequence of two rules because under the informative prior model $((X, Y), (X, Z))$ has the highest probability.

8.3.3.4 Computational Trade-Offs

The timing profile under a misinformative prior is interesting because one might suspect that the Posterior spends at least a little time wasted looking at the wrong rules. Table 8.13 compares the expected time in seconds for the Posterior across the 3- to 5-variable domains as the misinformative prior is built up.

Under the misinformative prior for the 3-variable domain, the Posterior is slower under 1-, 2-, and 3-edges in the prior. When there are zero edges no meaningful difference is seen as expected since in both cases the prior is actually uninformative. However for the 6-edge prior in the 4-variable domain and for 5- to 10-edges in the 5-variable domain, a noticeable

Table 8.13. Expected Times in Seconds to Identify an Inference Rule Sequence Across 3- to 5-variables With a Misinformative Prior.

Domain	Edges	PC	Posterior (Informative)	Posterior (Misinformative)
3	0-edges	0.0013	0.0195	0.0196
3	1-edges	0.0014	0.0198	0.0232
3	2-edges	0.0015	0.0214	0.0233
3	3-edges	0.0016	0.0130	0.0175
4	0-edges	0.0091	0.3220	0.3221
4	1-edges	0.0099	0.3044	0.3317
4	2-edges	0.0098	0.3259	0.3145
4	3-edges	0.0096	0.2587	0.2708
4	4-edges	0.0095	0.2763	0.2818
4	5-edges	0.0094	0.2715	0.2928
4	6-edges	0.0095	0.4865	0.4334
5	0-edges	0.0701	12.7453	12.9969
5	1-edges	0.0725	4.1229	9.2943
5	2-edges	0.0626	4.5821	8.9088
5	3-edges	0.0615	6.3945	6.2957
5	4-edges	0.0628	6.2981	5.9124
5	5-edges	0.0672	5.0806	5.0070
5	6-edges	0.0649	3.7696	2.4784
5	7-edges	0.0658	3.8425	3.9751
5	8-edges	0.0643	3.7981	4.0447
5	9-edges	0.0645	3.8931	4.1303
5	10-edges	0.0671	3.6866	3.4690

difference is seen in that the Posterior under the misinformative prior actually requires less time than it does under the informative prior. This coincides with the improvement in the expected costs of the inference rule sequence seen in the performance evaluation section and as such this improvement is linked to the specifics of the prior and should not be expected in general when priors are misspecified.

8.3.4 Performance as a Function of the Size of M

For all of the results presented thus far, a set M of size 100 has been used. During preliminary investigations into using M to represent the probability distribution across the model space an exact PMF was initially considered where the $|M|$ was set to the number of acyclic models in the domain. As one might expect, this quickly became infeasible for anything other than the 3-, 4-, and 5-variable domains with the 5-variable domain holding 29281 models. To determine a value of M that reduced what had to be held in memory and yet was large enough to give good results, performance of the Posterior as a function of the size of M was evaluated.

Three sizes for M were used in the evaluation: M = 10, 100, and 1000. For the 3- and 4-variable domains, this size with M=1000 is large enough to sample the entire model space in memory at once. For M=100, it can sample all models in the 3-variable model space but only under the full 6-edge prior can it sample an entire model space for the 4-variable domain. In all three set sizes, the best any one of them can do is to approximate the model space at any time.

Figure 8.28 shows the performance of the Posterior across M=10, 100, and 1000. We've also included the PC policy as reference. Table 8.14 provides the expected costs across all four policies.

Note that for M=10 it initially follows the performance characteristic of the M=100 and M=1000 executions but at 5-variable it follows, almost exactly, the performance character-

Table 8.14. Expected Costs for the Posterior Across Different Sizes of M.

Domain	Edges	Models	PC	Posterior M=10	Posterior M=100	Posterior M=1000
3-var	0-edges	25	5.040	5.013	4.601	4.604
3-var	1-edges	16	5.500	3.452	3.201	3.313
3-var	2-edges	16	5.500	3.570	3.446	3.534
3-var	3-edges	11	5.364	3.105	2.851	3.225
4-var	0-edges	543	17.002	16.613	16.271	14.278
4-var	1-edges	336	18.508	14.456	12.216	12.213
4-var	2-edges	336	19.254	15.035	12.292	12.295
4-var	3-edges	234	17.757	13.370	10.354	10.363
4-var	4-edges	234	15.569	11.409	9.220	9.223
4-var	5-edges	143	17.486	9.303	7.703	7.710
4-var	6-edges	93	18.451	8.314	7.625	7.624
5-var	0-edges	29281	48.555	48.556	42.156	40.549
5-var	1-edges	17632	52.404	52.417	39.228	39.170
5-var	2-edges	17632	54.314	54.294	40.619	40.735
5-var	3-edges	12404	50.162	50.163	35.934	37.197
5-var	4-edges	12404	45.212	45.203	30.308	32.404
5-var	5-edges	7394	49.697	49.683	27.846	29.654
5-var	6-edges	4882	52.340	52.344	29.180	29.030
5-var	7-edges	4882	54.822	54.835	31.024	31.375
5-var	8-edges	4882	55.809	55.832	31.841	32.036
5-var	9-edges	4882	57.259	57.275	33.365	33.115
5-var	10-edges	2916	61.458	61.511	31.360	30.251

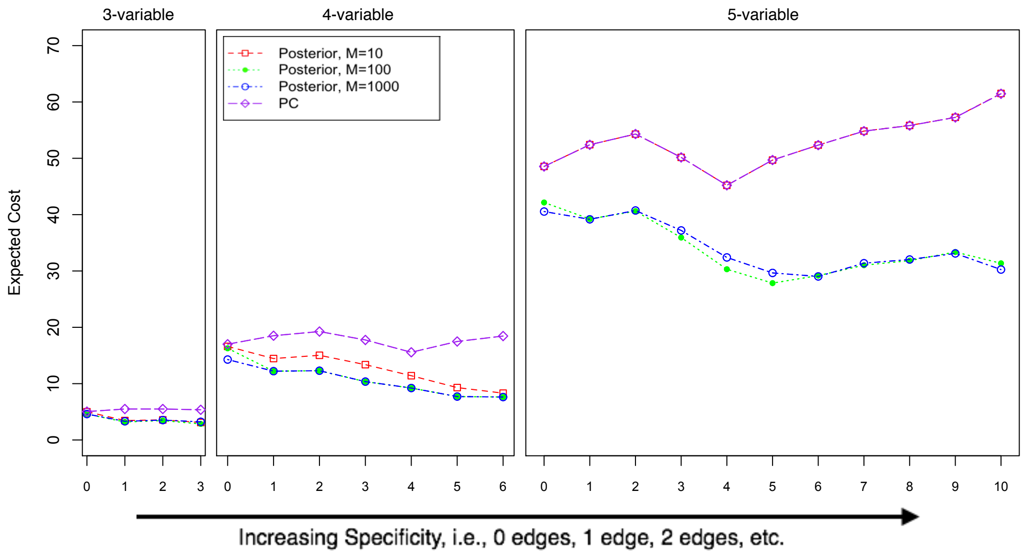


Figure 8.28. Comparison of the Posterior Under Different Sizes of M.

istic of the PC policy. For small domains, M=10 is enough to sample a representative set of models that the informed search policy can still identify a good inference rule sequence. But as the number of potential models in the domain grows, M=10 is too small and as with the PC algorithm, the M=10 version of the Posterior is unable to make use of the probabilities across the model space to guide selection of rules. In fact M=10 is slightly worse than the PC algorithm indicating it is pursuing rules that do not help as much as it would to iterate through the marginal and conditional independence inference rules.

Another interesting observation is that M=100 does as well as the M=1000 Posterior and in some cases better. This implies that 100 models, sampled randomly from the model space, provides enough to rank the rules such that the Posterior can identify inference rule sequences as short as those found by sampling 1000 models. The reason most likely is that for the 5-variable domain 1000 sampled models isn't able to rank the available rules any better than it can with 100 models. This is one part of the trade-off with approximating the model space with a smaller memory requirement.

CHAPTER 9
CONCLUSION

9.1 Summary of Contributions

In this dissertation a new method of inferring a causal model by considering causal learning as a search across the space of inference rules has been presented. This search identifies an inference rule sequence that when applied constrains the space of potential causal models for a domain.

The idea of the causal search policy was introduced as a method that allows an algorithm to search across the set of inference rules. And given a set of inference rules and a goal it was demonstrated that different policies generate different inference rule sequences that can produce the same causal knowledge but with different performance characteristics.

An ideal causal search policy was developed called the Optimal policy to evaluate the best performance in terms of sequence length that a policy could achieve. For example the PC policy based on the most prominent constraint-based causal learning algorithm, is far from optimal. It grows super-exponentially in the length of its inference rule sequence as a function of the size of the domain to which it is applied.

Finally the Posterior policy was introduced that used a prior over the model space to guide selection of rules and it was demonstrated how the Posterior policy is able to achieve shorter inference rule sequences over the PC policy, even when the prior is misspecified. This is because the Posterior is a member of a class of informed policies that select their next steps based on what they learn from applying previous rules.

9.2 Implications

The different inference rules can be unified into a single formal specification. This breaks the unnecessary restrictions algorithms have been limited to where they only consider one class of inference rule, such as conditional independence rules. With a formal specification, rules are selectable by a learning algorithm as they are applicable and with clearly explained actions and outcomes.

As researchers studying causality, concern about the quality of the inferred causal model can be greater than how well the algorithm itself is designed. This decouples the specification and strength of the inference rule from the algorithm itself. There is no longer a need to extend any one algorithm to account for weakness in the assumptions embedded in that algorithm's inference rules.

Learning over larger domains can be achieved within system resource limits such as time or cost. The fixed-approaches of current learning algorithms are prone to super-exponential inference rule sequences that are impractical for common domains. A causal search policy can decide on an inference rule sequence based on the cost and benefit of the rule.

9.3 Areas of Future Work

Our work here on the development of informed search policies is only the beginning of what is possible with combining reasoning with inference rules. There are several research directions that could not be covered by this dissertation but can be investigated in future work. These include expanding the set of inference rules considered by the policies, incorporating probabilistic constraints across the model space rather than deterministic constraints, and development of new search policies and new heuristics.

9.3.1 Faster Search By Sampling Inference Rules

The major drawback to the Posterior policy is the time it takes to reason about its next actions. One method to overcome the time to select an action is to sample from the set of inference rules and to rank those as opposed to identifying every possible inference rule in the set of sampled models. The algorithm would first sample M from the model space and then it would sample I from the rule space, and it would use M to rank members of I.

This solution presents an interesting trade-off: a potentially longer inference rule sequences for shorter time to decide the next action. The length of the final sequence may grow as the chances of missing a highly discerning rule increases as a function of the sample size of I resulting in needing to apply more rules than a more optimal approach. Research into methods of sampling, good sizes for I, and measuring the trade-off under different choices would benefit causal search.

9.3.2 Additional Inference Rules

The research in this dissertation limited itself to a small set of inference rules including the conditional independence inference rule, conditional dependence inference rule, and edge-orientation rules.

There is still much that can be done to describe the predicates that make up the inference rule specification including the development of a base set of predicates capable of representing the full breadth of inference rules. There are many QEDs that can be formally specified and research could investigate their use along side conditional independence rules and edge-orientation rules. It is not yet known if there is a better way to describe the utility of inference rules and improved understanding of inference rule utilities would provide a more realistic context for evaluating the utility of causal search. This work also assumed complete information in that each rule could be given a correct answer from an oracle. The impact of probabilities across the causal assumptions was not investigated nor how those probabilities influence rule utility.

9.3.3 Causal Search with Probabilistic Constraints

One area this research will focus on in the future is to investigate the impact on causal learning when probabilistic constraints are used rather than deterministic constraints. When using probabilistic constraints a probability is assigned to the outcomes of a rule. The probability on a inference rule could be a combination of the probability that an assumption holds combined with the probability that the inference rule holds via statistical significance.

Consider a marginal independence inference rule. If X and Y are marginally independent, this can be assigned as a strong probability at 1.0. But if instead it is found that marginal independence of X and Y is false, then it could be said that there is a $1/|i|$ probability of that holding where $|i| = [2 ** (|V| - 2)] - 1$ since X and Y can be ignored from V and the empty set from the power set because it was already run with the marginal inference rule. As CIRules are applied using all other sets of variables, X and Y become more likely to be dependent. The probabilities could be weighted as the set size increases meaning it is more unlikely that larger sets will make X and Y independent but violations of causal sufficiency and selection bias must be taken into account.

With probabilistic constraints, the causal search problem is no longer specified as an MDP but rather as a partially observable MDP (POMDP). The model space is no longer considered as fully observable because the exact state become unknown. The probability of the belief across the model space closely aligns with the probability that can be assigned over the belief state within the POMDP. The non-deterministic environment could then be represented as a combination of AND/OR search trees with belief states.

9.3.4 Development of New Policies

The Posterior policy is one of many informed search policies. Consider the Contentious policy that applies inference rules whose evaluation, if True or False, would be the most surprising. Another interesting policy would be one based on the Lowest Entropy. The Lowest Entropy policy could rank inference rule based on the resulting entropy that occurs as a result of splitting the model space. Since the outcome of a rule essentially splits the model space into two components, one set where the rule is True and another where it is False, using entropy as a guide would have the policy choose rules that can eliminate models in a manner similar to the process where a decision tree identifies splits.

Search policy heuristics could be extended to allow user interaction where a user guides the policy after every inference rule by re-specifying the model space manually. Policies can be designed so that they are guided by different goals rather than trying to uncover a single model. Researchers could assign weights to constraints to specify priorities for learning. This would allow a policy to make inference rule decisions based on user preference rather than driving towards the highest probable models.

BIBLIOGRAPHY

[1] Alan Abramowitz and Jeffrey A. Segal. *Senate Elections.* University of Michigan Press, 1992.

[2] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to Act Using Real-Time Dynamic Programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[3] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees.* CRC press, 1984.

[4] Donald T. Campbell and Julian C. Stanley. *Experimental and Quasi-Experimental Designs for Research.* Rand McNally, Chicago, IL, 1966.

[5] David M. Chickering. Learning Bayesian Networks is NP-Complete. In Douglas H. Fisher and Hans-Joachim Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, chapter 12, pages 121–130. Springer, 1996.

[6] David M. Chickering. Optimal Structure Identification with Greedy Search. *JMLR*, 3:507–554, November 2002.

[7] William G. Cochran and Gertrude M. Cox. *Experimental Designs.* Wiley, New York, 1950.

[8] Diego Colombo, Marloes H. Maathuis, Markus Kalisch, Thomas S. Richardson, and et. al. Learning High-Dimensional Directed Acyclic Graphs with Latent and Selection Variables. *The Annals of Statistics*, 40(1):294–321, 2012.

[9] Gregory Cooper and Changwon Yoo. Causal Discovery from a Mixture of Experimental and Observational Data. In *Proc. Fifthteenth Conference on Uncertainty in Artificial Intelligence (UAI99)*, pages 116–125. Citeseer, 1999.

[10] Alexander Philip Dawid. Conditional Independence in Statistical Theory. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–31, 1979.

[11] Frederick Eberhardt. *Causation and Intervention.* PhD thesis, CMU, 2007.

[12] Frederick Eberhardt. Causal Discovery as a Game. In Dominik Janzing Isabelle Guyon and Bernhard Schölkopf, editors, *Causality: Objectives and Assessment(NIPS 2008 Workshop)*, volume 6 of *JMLR Workshop and Conference Proceedings*, pages 87–96, Whistler, BC, Canada, December 12, 2010.

[13] Frederick Eberhardt, Clark Glymour, and Richard Scheines. On the Number of Experiments Sufficient and in the Worst Case Necessary to Identify All Causal Relations among N Variables. In *In UAI*, pages 178–184. AUAI Press, 2005.

[14] Frederick Eberhardt, Clark Glymour, and Richard Scheines. N-1 Experiments Suffice to Determine the Causal Relations Among N Variables. *Innovations in Machine Learning*, pages 97–112, 2006.

[15] Nir Friedman and Daphne Koller. Being Bayesian about Network Structure. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 201–210. Morgan Kaufmann Publishers Inc., 2000.

[16] Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. Kernel Measures of Conditional Dependence. In *NIPS*, volume 20, pages 489–496, 2007.

[17] Arthur Getis. A Spatial Causal Model of Economic Interdependency Among Neighboring Communities. *Environment and Planning A*, 21(1):115–120, 1989.

[18] Priscilla E. Greenwood and Michael S. Nikulin. *A Guide to Chi-Squared Testing*, volume 280. John Wiley & Sons, 1996.

[19] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J. Smola. A Kernel Method for the Two-Sample-Problem. In *Advances in Neural Information Processing Systems*, pages 513–520, 2006.

[20] Bernhard Haeupler, Telikepalli Kavitha, Rogers Mathew, Siddhartha Sen, and Robert E. Tarjan. Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance. *ACM Transactions on Algorithms (TALG)*, 8(1):3, 2012.

[21] Eric A. Hansen and Shlomo Zilberstein. Lao*: A Heuristic Search Algorithm that Finds Solutions with Loops. *Artificial Intelligence*, 129(1):35–62, 2001.

[22] Daniel M. Hausman and James Woodward. Independence, Invariance and the Causal Markov Condition. *The British Journal for the Philosophy of Science*, 50(4):521–583, 1999.

[23] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian Networks: the Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, 1995.

[24] David Heckerman, Christopher Meek, and Gregory Cooper. A Bayesian Approach to Causal Discovery. *Computation, Causation, and Discovery*, 19:141–166, 1999.

[25] Samuel S. Hwang, Jae S. Chang, Kyu Y. Lee, Yong M. Ahn, and Yong S. Kim. The Causal Model of Insight in Schizophrenia Based on the Positive and Negative Syndrome Scale Factors and the Structural Equation Modeling. *The Journal of Nervous and Mental Disease*, 197(2):79, 2009.

[26] David D. Jensen, Andrew S. Fast, Brian J. Taylor, and Marc E. Maier. Automatic Identification of Quasi-Experimental Designs for Discovering Causal Knowledge. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 372–380. ACM, 2008.

[27] Ron Kohavi, Randal M. Henne, and Dan Sommerfield. Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the Hippo. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 959–967. ACM, 2007.

[28] Ronny Kohavi, Thomas Crook, Roger Longbotham, Brian Frasca, Randy Henne, Juan Lavista Ferres, and Tamir Melamed. Online Experimentation at Microsoft. In *Proceedings of the Third International Workshop on Data Mining Case Studies*, pages 11–23. ACM, 2009.

[29] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[30] Stefan Kramer, Nada Lavra, and Peter Flach. *Propositionalization Approaches to Relational Data Mining*. Springer-Verlag, New York, NY, 2001.

[31] Aurelie C. Lozano, Hongfei Li, Alexandru Niculescu-Mizil, Yan Liu, Claudia Perlich, Jonathan Hosking, and Naoki Abe. Spatial-Temporal Causal Modeling for Climate Change Attribution. In *Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 587–596. ACM, 2009.

[32] Marc E. Maier, Matthew J. Rattigan, and David D. Jensen. Discovering Causal Knowledge by Design. Technical report, Tech Report 09-47, University of Massachusetts Amherst, Computer Science Department, 2009.

[33] Marc E. Maier, Brian J. Taylor, Huseyin Oktay, and David D. Jensen. Learning Causal Models of Relational Domains. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 531–538. AAAI, 2010.

[34] Christopher Meek. Causal Inference and Causal Explanation with Background Knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 403–410. Morgan Kaufmann Publishers Inc., 1995.

[35] Stijn Meganck, Philippe Leray, and Bernard Manderick. Learning Causal Bayesian Networks from Observations and Experiments: A Decision Theoretic Approach. *Modeling Decisions for Artificial Intelligence*, pages 58–69, 2006.

[36] Stijn Meganck, Philippe Leray, and Bernard Manderick. Uncado: Unsure Causal Discovery. In *Les 4mes Journes Francophones sur les Rseaux Baysiens (JFRB 2008)*, volume 8, pages 94–104, 2008.

[37] Kevin P. Murphy. Active Learning of Causal Bayes Net Structure. Technical report, Department of Computer Science, University of California, Berkeley, 2001.

[38] Hüseyin Oktay, Brian J. Taylor, and David D. Jensen. Causal Discovery in Social Media Using Quasi-Experimental Designs. In *Proceedings of the First Workshop on Social Media Analytics*, pages 1–9. ACM, 2010.

[39] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Fransico, CA, 1988.

[40] Judea Pearl. Causal Diagrams for Empirical Research. *Biometrika*, 82(4):669–688, 1995.

[41] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, 2000.

[42] Judea Pearl, Thomas Verma, and et. al. *A Theory of Inferred Causation.* Morgan Kaufmann San Mateo, CA, 1991.

[43] Joseph Ramsey, Jiji Zhang, and Peter L. Spirtes. Adjacency-Faithfulness and Conservative Causal Inference. *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence*, pages 401–408, 2006.

[44] Matthew J. Rattigan, Marc E. Maier, and David D. Jensen. Relational Blocking for Causal Discovery. In *Proceedings of the Twenty-Fifth National Conference on Artificial Intelligence*, pages 145–151, 2011.

[45] Robert W. Robinson. Counting Unlabeled Acyclic Digraphs. *Combinatorial Mathematics V*, pages 28–43, 1977.

[46] Stuart J. Russell, Peter Norvig, John F. Canny, Jitendra M. Malik, and Douglas D. Edwards. *Artificial Intelligence: a Modern Approach*, volume 2. Prentice hall Englewood Cliffs, 1995.

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

[48] Ilya Shpitser and Judea Pearl. Dormant Independence. In *Proceedings of the Twenty Third Conference on Artificial Intelligence (AAAI-08)*, pages 1081–1087. AAAI Press, 2008.

[49] Peter Spirtes. Introduction to Causal Inference. *The Journal of Machine Learning Research*, 11:1643–1662, 2010.

[50] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search.* MIT Press, Cambridge, MA, 2nd edition, 2001.

[51] Peter Spirtes, Christopher Meek, and Thomas Richardson. Causal Inference in the Presence of Latent Variables and Selection Bias. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 499–506. Morgan Kaufmann Publishers Inc., 1995.

[52] Jin Tian, Azaria Paz, and Judea Pearl. Finding Minimal d-separators. Technical report, UCLA Computer Science Department, 1998.

[53] Simon Tong and Daphne Koller. Active Learning for Structure in Bayesian Networks. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 863–869. Citeseer, 2001.

[54] Thomas Verma and Judea Pearl. Equivalence and Synthesis of Causal Models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, pages 255–270, New York, NY, USA, 1991.

[55] Janet Wittes and Sylvan Wallenstein. The Power of the MantelHaenszel Test. *Journal of the American Statistical Association*, 82(400):1104–1109, 1987.

[56] Changwon Yoo and Gregory F. Cooper. A Computer-Based Microarray Experiment Design-System for Gene-Regulation Pathway Discovery. In *AMIA Annual Symposium Proceedings 2003*, pages 733–737. American Medical Informatics Association, 2003.

[57] Jiji Zhang and Peter Spirtes. Strong Faithfulness and Uniform Consistency in Causal Inference. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 632–639. Morgan Kaufmann Publishers Inc., 2002.

[58] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-Based Conditional Independence Test and Application in Causal Discovery. *ArXiv e-prints, 1202.3775*, 2012.