Mind Change Optimal Learning of Bayes Net Structure from Dependency and Independency Data

Oliver Schulte[a,*], Wei Luo[b], Russell Greiner[c]

[a] *Simon Fraser University, Burnaby, BC V5A 1S6, Canada*
[b] *The University of Queensland, Brisbane, Queensland 4072, Australia*
[c] *University of Alberta, Edmonton, Alberta T6G 2E8, Canada*

Abstract

This paper analyzes the problem of learning the structure of a Bayes net in the theoretical framework of Gold's learning paradigm. Bayes nets are one of the most prominent formalisms for knowledge representation and probabilistic and causal reasoning. We follow constraint-based approaches to learning Bayes net structure, where learning is based on observed conditional dependencies and independencies between variables of interest (e.g., the data are of the form "X is dependent on Y given any assignment to variables $\mathbf{S}$" or of the form "X is independent of Y given any assignment to variables $\mathbf{S}$"). Applying learning criteria in this model leads to the following results. (1) The mind change complexity of identifying a Bayes net graph over variables $\mathbf{V}$ from either dependency data or from independency data is $\binom{|\mathbf{V}|}{2}$, the maximum number of edges. (2) There is a unique fastest mind-change optimal Bayes net learner for either data type; convergence speed is evaluated using Gold's dominance notion of "uniformly faster convergence". For dependency data, the optimal learner conjectures a graph if it is the unique Bayes net pattern that satisfies the observed dependencies with a minimum number of edges, and outputs "no guess" otherwise. For independency data, the optimal learner conjectures a graph if it is the unique Bayes net pattern that satisfies the observed dependencies with a maximum number of edges, and outputs "no guess" otherwise. We investigate the complexity of computing the output of the fastest mind-change optimal learner for either data type, and show that each of these two problems is NP-hard (assuming P = RP). To our knowledge these are the first NP-hardness results concerning the existence of a uniquely optimal Bayes net structure.

1. Introduction

One of the goals of computational learning theory is to analyze the complexity of practically important learning problems, and to design optimal learning algorithms for them that meet performance guarantees. In this paper, we model

*Corresponding author

learning the structure of a Bayes net (BN) as a language learning problem in the Gold paradigm. We apply identification criteria such as mind change bounds [11, Ch. 12.2][26], mind-change optimality [16, 17], and text-efficiency (minimizing time or number of data points before convergence) [22, 10]. Bayes nets, one of the most prominent knowledge representation formalisms [24, 25, 13, 7], are widely used to define probabilistic models in a graphical manner, with a directed acyclic graph (DAG) whose edges link the variables of interest.

We base our model of BN structure learning on an approach known as "constraint-based" learning [6]. Constraint-based learning views a BN structure as a specification of conditional dependencies of the form $X \not\perp\!\!\!\perp Y|\mathbf{S}$, where X and Y are variables of interest and $\mathbf{S}$ is a set of variables disjoint from $\{X, Y\}$. (Read $X \not\perp\!\!\!\perp Y|\mathbf{S}$ as "variable X is dependent on variable Y given values for the variables in the set $\mathbf{S}$".) An example of a conditional dependence statement represented by a Bayes net would be written as $F \not\perp\!\!\!\perp M|C$ where F (resp., M, C) is eye colour of father (resp., mother, child). In this view, a BN structure is a syntactic representation of a dependency relation [24, Sec.3.3]. It is possible for distinct BN structures to represent the same dependency relation; in that case the equivalent BN structures share a partially directed graph known as a *pattern* (defined below), so a BN pattern is a unique syntactic representation of a dependency relation. A dependency relation meets the mathematical definition of a language in the sense of Gold's paradigm, where the "strings" in the language are dependence statements of the form "$X \not\perp\!\!\!\perp Y|\mathbf{S}$". Gold's paradigm considers language learning with both positive and negative data instances [10, 11]. In constraint-based BN learning, each positive instance corresponds to a dependency statement (e.g., "father's eye color is dependent on mother's eye color given child's eye color") and each negative instance to an independency statement (e.g., "father's eye color is independent of mother's eye color").

We show that with data of either type alone, the mind change complexity of learning a Bayes net graph for a given set of variables $\mathbf{V}$ is $\binom{|\mathbf{V}|}{2}$—the maximum number of edges in a graph with node set $\mathbf{V}$. Our analysis leads to a characterization of BN learning algorithms that are mind-change optimal. A learner is mind-change optimal if and only if it minimizes the number of mind changes not only globally in the entire learning problem, but also locally in subproblems encountered after receiving some evidence [16, 17]. For dependency data, mind-change optimal BN learners are exactly those that conjecture a BN pattern G only if the pattern is the unique one that satisfies the observed dependencies *with a minimum number of edges*. For independency data, mind-change optimal BN learners are exactly those that conjecture a BN pattern G only if the pattern is the unique one that satisfies the observed independencies *with a maximum number of edges*.

Applying Gold's notion of dominance in convergence time [10, p.462], we show that there is a fastest mind-change optimal learner, for either data type, whose convergence time dominates that of all other mind-change optimal learners. The fastest learners are defined as follows: If there is more than one BN pattern G that satisfies the observed dependencies (independencies) with a minimum (maximum) number of edges, output "?" (for "no guess"). If there is a

unique pattern G that satisfies the observed dependencies with an optimal number of edges, output G. Thus standard identification criteria in Gold's paradigm lead to a natural and novel algorithm for learning BN structure. The technically most complex result of the paper examines the computational complexity of the fastest mind-change optimal BN learners: we show that computing the conjectures for each data type is NP-hard (assuming that P = RP).

Paper Organization. We introduce concepts and results from both learning theory and Bayes net theory in the next section. Section 3 presents and discusses our model of BN structure learning as a language learning problem. Section 4 analyzes the mind change complexity of BN structure learning. Section 5 characterizes the mind-change optimal learning algorithms for this problems and describes the fastest mind-change optimal learner. The final two sections define the problem of computing the output of the fastest mind-change optimal learner and show that the problem is NP-hard. We close this section by describing some related work.

Related Work. Many BN learning systems follow the "search and score" paradigm, seeking a structure that optimizes some numeric scoring function [6]. Our work is in the alternative constraint-based (CB) paradigm; see [6], [21, Ch.10]. The Tetrad system [29] includes a number of CB methods for different classes of Bayes nets. A fundamental difference between existing CB approaches and our model is that the existing methods assume access to an oracle that returns an answer for every query of the form "does $X \not\perp\!\!\!\perp Y|\mathbf{S}$ hold?". In contrast, our model corresponds to the situation of a learner whose evidence (in the form of (in)dependency assertions) grows incrementally over time in an on-line setting. Another difference is that existing CB methods assume that their oracle indicates whether two variables are conditionally independent (learning from negative data) [31, Ch.5.4], or even that the oracle indicates *both* whether two variables are conditionally dependent and whether they are conditionally independent (learning from positive and negative data) [6]. Kelly outlines a model of learning causal graphs from dependency data only [15]. To our knowledge, our work is the first application of Gold's language learning paradigm to Bayes net learning, and the first learning-theoretic analysis of constraint-based BN learning from dependency data only.

A Bayes net that satisfies a set of given dependencies $\mathcal{D}$ is said to be an I-map for $\mathcal{D}$. We show the NP-hardness of the following problem: for a given set of dependencies $\mathcal{D}$ represented by an oracle O (Section 6), decide whether there is a unique edge minimal I-map G for $\mathcal{D}$, and if so, output G. Bouckaert proved that the problem is NP-hard without the uniqueness condition [2, Lemma 4.5]. However, Bouckaert's proof cannot be adapted for our uniqueness problem, which requires a much more complex reduction. To our knowledge, ours are the first NP-hardness results for deciding the existence of a *uniquely* optimal Bayes net structure for any optimality criterion.

2. Preliminaries: Language Identification and Bayes Nets

We first introduce general concepts from learning theory, followed by basic definitions from Bayes net theory and computational complexity theory.

2.1. Language Identification with Bounded Mind Changes

We employ notation and terminology from [12], [19, Ch.1], [22], and [10]. We write $\mathbb{N}$ for the set of natural numbers $\{0, 1, 2, \ldots\}$. The symbols $\subseteq, \supseteq, \subset, \supset$, and $\emptyset$ respectively stand for subset, superset, proper subset, proper superset, and the empty set. We assume that there is an at most countable set $\mathbb{E}$ of potential evidence items (strings in language learning). A **language** is a subset of $\mathbb{E}$; we write L for a generic language [10, p.449]. A **language learning problem** is defined by a collection of languages; we write $\mathcal{L}$ for a generic collection of languages. A **text** T is a mapping of $\mathbb{N}$ into $\mathbb{E} \cup \{\#\}$, where $\#$ is a symbol not in $\mathbb{E}$. (The symbol $\#$ models pauses in data presentation.) We write content(T) for the intersection of $\mathbb{E}$ and the range of T. A text T is **for** a language L iff $L = \text{content}(T)$. The initial sequence of text T of length n is denoted by $T[n]$. The set of all finite initial sequences over $\mathbb{E} \cup \{\#\}$ is denoted by SEQ. We also use SEQ$(\mathcal{L})$ to denote finite initial sequences consistent with languages in $\mathcal{L}$. Greek letters σ and τ range over SEQ. The notation $|\sigma|$ denotes the length of sequence σ. We write content(σ) for the intersection of $\mathbb{E}$ and the range of σ. We write $\sigma \subset T$ to denote that text T extends initial sequence σ; similarly for $\sigma \subset \tau$. A **learner** Ψ **for** a collection of languages $\mathcal{L}$ is a mapping of SEQ$(\mathcal{L})$ into $\mathcal{L} \cup \{?\}$, where ? corresponds to the vacuous conjecture "no guess". Our term "learner" corresponds to the term "scientist" in [19, Ch.2.1.2]. We say that a learner Ψ **identifies** a language L on a text T for L, if $\Psi(T[n]) = L$ for all but finitely many n. Next we define identification of a language collection relative to some evidence.

Definition 1. *A learner Ψ for $\mathcal{L}$ **identifies** $\mathcal{L}$ given σ $\iff$ for every language $L \in \mathcal{L}$, and for every text $T \supset \sigma$ for L, the learner Ψ identifies L on T.*

Thus a learner Ψ identifies a language collection $\mathcal{L}$ if Ψ identifies $\mathcal{L}$ given the empty sequence Λ. A learner Ψ **changes its mind** at some nonempty finite sequence $\sigma \in$ SEQ if $\Psi(\sigma) \neq \Psi(\sigma^-)$ and $\Psi(\sigma^-) \neq ?$, where σ^- is the initial segment of σ with σ's last element removed [11, Ch.12.2]. (No mind changes occur at the empty sequence Λ.)

Definition 2. *Let* $\text{MC}(\Psi, T, \sigma)$ *denote the total number of mind changes of Ψ on text T after sequence σ (i.e.,* $\text{MC}(\Psi, T, \sigma) = |\{\tau : \sigma \subset \tau \subset T : \Psi$ *changes its mind at* $\tau\}|$*).*

1. *Ψ identifies $\mathcal{L}$ **with mind-change bound** k given $\sigma \iff \Psi$ identifies $\mathcal{L}$ given σ and Ψ changes its mind at most k times on any text $T \supset \sigma$ for a language in $\mathcal{L}$ after σ (i.e., if $T \supset \sigma$ extends data sequence σ and T is a text for any language $L \in \mathcal{L}$, then* $\text{MC}(\Psi, T, \sigma) \leq k$*).*

2. *A language collection $\mathcal{L}$ is **identifiable with mind change bound** k given σ $\iff$ there is a learner Ψ such that Ψ identifies $\mathcal{L}$ with mind change bound k given σ.*

2.2. Bayes Nets: Basic Concepts and Definitions

We employ notation and terminology from [25], [24] and [31]. A **Bayes net structure** is a directed acyclic graph $G = (\mathbf{V}, E)$. Two nodes X, Y are **adjacent** in a BN if G contains an edge $X \to Y$ or $Y \to X$. The **pattern** $\pi(G)$ of DAG G is the partially directed graph over $\mathbf{V}$ that has the same adjacencies as G, and contains an arrowhead $X \to Y$ if and only if G contains a triple $X \to Y \leftarrow Z$ where X and Z are not adjacent. Since a pattern is also a graph, we use G for patterns as well unless the distinction between graphs and patterns is important in context. An (undirected) **path** in G is a sequence of nodes such that every two consecutive nodes in the sequence are adjacent in G and no node occurs more than once in the sequence. A node Y is a **collider on undirected path** p in DAG G if p contains a triple $X \to Y \leftarrow Z$. If X and Z are adjacent in G, the collider Y is **shielded**, otherwise **unshielded**. Every BN structure defines a separability relation between a pair of nodes X, Y relative to a set of nodes $\mathbf{S}$, called **d-separation**: if X, Y are two variables and $\mathbf{S}$ is a set of variables disjoint from $\{X, Y\}$, then $\mathbf{S}$ d-separates X and Y if along every (undirected) path between X and Y there is a node W satisfying one of the following conditions:

1. W is a collider on the path and neither W nor any of its descendants is in $\mathbf{S}$, or
2. W is not a collider on the path and W is in $\mathbf{S}$.

We write $(X \perp\!\!\!\perp Y|\mathbf{S})_G$ if X and Y are d-separated by $\mathbf{S}$ in graph G. If two nodes X and Y are not d-separated by $\mathbf{S}$ in graph G, then X and Y are **d-connected** by $\mathbf{S}$ in G, written $(X \not\perp\!\!\!\perp Y|\mathbf{S})_G$. The d-connection relation, or **dependency relation**, for a graph is denoted by $\mathcal{D}_G$, that is, $\langle X, Y, \mathbf{S}\rangle \in \mathcal{D}_G$ iff $(X \not\perp\!\!\!\perp Y|\mathbf{S})_G$. The d-separation relation, or **independency relation**, for a graph is denoted by $\mathcal{I}_G$, that is, $\langle X, Y, \mathbf{S}\rangle \in \mathcal{I}_G$ if and only if $(X \perp\!\!\!\perp Y|\mathbf{S})_G$. Verma and Pearl proved that two Bayes nets G_1 and G_2 represent the same dependency relation iff they have the same pattern (i.e., $\mathcal{D}_{G_1} = \mathcal{D}_{G_2}$ iff $\pi(G_1) = \pi(G_2)$ [35, Thm. 1]). Thus we use a pattern as a syntactic representation for a Bayes net (in)dependency relation. The **dependency space** over a set of variables $\mathbf{V}$, denoted by $\mathcal{D}_{\mathbf{V}}$, contains all conditional dependency statements of the form $(X \not\perp\!\!\!\perp Y|\mathbf{S})$, where X, Y are distinct variables in $\mathbf{V}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y\}$. The **independency space** over a set of variables $\mathbf{V}$, denoted by $\mathcal{I}_{\mathbf{V}}$, contains all conditional independency statements of the form $(X \perp\!\!\!\perp Y|\mathbf{S})$, where X, Y are distinct variables in $\mathbf{V}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y\}$. We shall make use of the following basic fact about d-separation; for the proof see [24, Cor. 3.4].

Lemma 3. *If two nodes A, B in a DAG G are not adjacent, then there is a separating set $\mathbf{S}$ such that A, B are d-separated by $\mathbf{S}$ in G, that is, $(A \perp\!\!\!\perp B|\mathbf{S})_G$.*

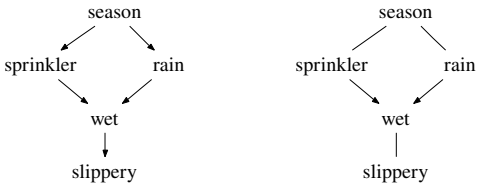

Figure 1: Sprinkler network and its pattern

Example. Figure 1 shows a Bayes net from [25, p.15]. In this network, node `wet` is an unshielded collider on the path `sprinkler` − `wet` − `rain`; node `wet` is not a collider on the path `sprinkler` − `wet` − `slippery`. The pattern of the network has the same skeleton, but contains only two edges that induce the collider `wet`. The variables `sprinkler` and `rain` are d-separated given the set {`season`}, written (`sprinkler` ⫫ `rain`|`season`)$_G$, which can be seen as follows. There are two undirected paths from `sprinkler` to `rain`, namely `sprinkler` − `wet` − `rain` and `sprinkler` − `season` − `rain`. For the first path, clause (1) of the definition of d-separation applies, since `wet` is a collider on the path `sprinkler` − `wet` − `rain` and neither `wet` nor its descendant `slippery` is contained in the conditioning set {`season`}. For the second path, clause (2) applies, since `season` is not a collider on the path `sprinkler` − `season` − `rain` and `season` is a member of the conditioning set {`season`}. The variables `sprinkler` and `rain` are *not* d-separated given the set {`season`, `wet`}, written (`sprinkler` ⫫̸ `rain`|`season`)$_G$, because `wet` is a collider on the path `sprinkler` − `wet` − `rain` contained in the conditioning set, which violates clause (1) of the definition of d-separation.

2.3. Computational Complexity

This section presents some of the necessary background from complexity theory. We outline standard complexity theory concepts only briefly to introduce our notation; more details may be found in textbooks such as [23], [28]. A *decision problem* $\mathbb{P}$ asks whether a given input string s satisfies a certain property, in which case we write $s \in \mathbb{P}$. Let $\mathbb{P}, \mathbb{Q}$ be two decision problems and let f be a function that maps an input string s for $\mathbb{P}$ to an input string t for $\mathbb{Q}$. We say that f **many-one reduces** $\mathbb{P}$ to $\mathbb{Q}$ if $s \in \mathbb{P}$ if and only if $f(s) \in \mathbb{Q}$, for all inputs s. If the function f is computable in polynomial time, the problem $\mathbb{P}$ is **many-one reducible** to $\mathbb{Q}$ in polynomial time, which we denote by $\mathbb{P} \leq_P \mathbb{Q}$. If problem $\mathbb{P}$ can be decided with a finite number of calls to an oracle for problem $\mathbb{Q}$, the problem $\mathbb{P}$ is Turing-reducible to $\mathbb{Q}$. If problem $\mathbb{P}$ can be decided by a randomized algorithm with an oracle for problem $\mathbb{Q}$ that runs in polynomial time, we have a randomized polynomial-time Turing reduction of $\mathbb{P}$ to $\mathbb{Q}$, which we denote by $\mathbb{P} \leq_{\mathrm{RP}} \mathbb{Q}$. The class RP comprises the decision problems that can be decided in polynomial time with a randomized algorithm [23, Def.11.1]. The class NP is the class of problems that can be decided in nondeterministic polynomial time, and $\mathrm{NP}^{\mathrm{NP}}$ is the class of problems that can be decided in nondeterministic polynomial time given an oracle for an NP-complete problem (e.g. the satisfiability problem SAT). The class $\mathrm{FP}^{(\mathrm{NP}^{\mathrm{NP}})}$ comprises

the functions whose values can be computed in polynomial time given an oracle for problems in the class $\mathrm{NP}^{\mathrm{NP}}$ (cf. [23, Ch.17.1]). Our target problem for reduction will be the problem of deciding the existence of a unique exact cover by 3-sets.

UEC3SET

Instance A finite set U with $|U| = 3q$ and a collection $\mathcal{C}$ of 3-element subsets of U.

Question Does $\mathcal{C}$ contain a unique *exact cover* for U, that is, a unique subcollection $\mathcal{C}' \subseteq \mathcal{C}$ such that every element of U occurs in exactly one member of $\mathcal{C}'$?

We apply the following well-known result concerning the complexity of the unique exact cover problem.

Proposition 4. *Let* SAT *denote the satisfiability problem. There is a randomized polynomial-time Turing reduction of* SAT *to* UEC3SET*, so* $\mathrm{SAT} \leq_{\mathrm{RP}}$ UEC3SET*. Thus a polynomial time algorithm for* UEC3SET *yields a polynomial time algorithm for provided that* P = RP*. So* UEC3SET *is* NP*-hard under that assumption.*

The proposition follows from the famous theorem of Valiant and Vazirani that gives a probabilistic Turing reduction of SAT to Unique SAT [34]. Standard polynomial-time many-one reductions show that Unique SAT reduces to UEC3SET. Next we introduce our model of BN structure learning, which associates a language collection with a given set of variables $\mathbf{V}$; the language collection comprises all (in)dependency relations defined by Bayes net structures.

3. Bayes Net Learning With Bounded Mind Changes

This section defines our model of BN structure learning. We discuss the assumptions in the model and compare them to assumptions made in other constraint-based BN learning approaches.

3.1. Definition of the Learning Model

Fix a set of variables $\mathbf{V}$. Let $\mathcal{L}^D_{\mathbf{V}}$ be the **set of BN-dependency relations over variables V** (i.e., $\mathcal{L}^D_{\mathbf{V}} = \{\mathcal{D}_G : G \text{ is a pattern over } \mathbf{V}\}$). A **complete dependency sequence** T is a mapping of $\mathbb{N}$ into $\mathcal{D}_{\mathbf{V}} \cup \{\#\}$. A dependency sequence T is **for** a dependency relation $\mathcal{D}$ iff $\mathcal{D} = \mathrm{content}(T)$. A Bayes net learning algorithm Ψ for dependency data maps a finite data sequence σ over $\mathcal{D}_{\mathbf{V}} \cup \{\#\}$ to a pattern G or the output ? (for "no guess"). As Table 1 illustrates, this defines a language learning model, with some changes in terminology that reflect the Bayes net context.

General Language Learning	Bayes Net Structure Learning
string	conditional dependency statement $X \not\perp\!\!\!\perp Y \mid \mathbf{S}$
language	conditional dependency relation
index	pattern
text	complete dependency sequence

Table 1: The correspondence between constraint-based learning of Bayes Nets from conditional dependency data and Gold's language learning model.

Example. Let G be the DAG in Figure 1. The dependency relation for the graph $\mathcal{D}_G$ contains $\{ \langle \texttt{season}, \texttt{sprinkler}, \emptyset \rangle, \langle \texttt{season}, \texttt{sprinkler}, \{\texttt{rain}\} \rangle, \ldots, \langle \texttt{sprinkler}, \texttt{rain}, \{\texttt{season}, \texttt{wet}\} \rangle, \langle \texttt{sprinkler}, \texttt{rain}, \{\texttt{season}, \texttt{slippery}\} \rangle \}$. Any text enumerating $\mathcal{D}_G$ is a dependency sequence for $\mathcal{D}_G$.

For learning from independency data, let $\mathcal{L}^I_{\mathbf{V}}$ be the **set of BN-independency relations over variables V** (i.e., $\mathcal{L}^I_{\mathbf{V}} = \{\mathcal{I}_G : G \text{ is a pattern over } \mathbf{V}\}$). A **complete independency sequence** T is a mapping of $\mathbb{N}$ into $\mathcal{L}^I_{\mathbf{V}} \cup \{\#\}$. An independency sequence T is **for** an independency relation $\mathcal{I}$ iff $\mathcal{I} = \text{content}(T)$. A Bayes net learning algorithm Ψ for independency data maps a finite data sequence σ over $\mathcal{L}^I_{\mathbf{V}} \cup \{\#\}$ to a pattern G or the output ? (for "no guess").

3.2. Discussion of the Learning Model

We discuss and motivate the three key components of our learning model: the hypothesis space (language collection), the data generation model, and what a learner should output given data.

The Hypothesis Space. A Bayes net defines a dependency relation via the d-separation criterion. The motivation for this criterion stems from how a Bayes net represents a probability distribution P. Let P be a joint distribution over variables $\mathbf{V}$. If $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ are three disjoint sets of variables, then $\mathbf{X}$ and $\mathbf{Y}$ are **stochastically independent given S**, denoted by $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{S})_P$, if $P(\mathbf{X}, \mathbf{Y} \mid \mathbf{S}) = P(\mathbf{X} \mid \mathbf{S})\, P(\mathbf{Y} \mid \mathbf{S})$ whenever $P(\mathbf{S}) > 0$. If $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{S}$ are disjoint sets of nodes in G and $\mathbf{X}$ and $\mathbf{Y}$ are not empty, then $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{S}$ if and only if every pair $\langle X, Y \rangle$ in $\mathbf{X} \times \mathbf{Y}$ is d-separated by $\mathbf{S}$. In constraint-based BN learning, it is common to assume that the probability distribution generating the data of interest has a faithful BN representation [31, Thm.3.2], [25, Ch.2.4] (for further discussion, see [36], [32, Ch.8.1]).

Definition 5. *Let* $\mathbf{V}$ *be a set of variables,* G *a Bayes net over* $\mathbf{V}$*, and* P *a joint distribution over* $\mathbf{V}$*. Then* G *is* ***faithful to*** P *if* $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{S})_P$ *in* $P \iff (\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{S})_G$ *in* G*.*

Faithfulness implies that the dependencies in the data can be exactly represented in a Bayes net or a pattern; our language learning model therefore incorporates this assumption. It is easy to see that a graph G is faithful to a distribution P if and only if G is faithful with respect to variable pairs, that is, if $(X \not\perp\!\!\!\perp Y \mid \mathbf{S})_P$ in $P \iff (X \not\perp\!\!\!\perp Y \mid \mathbf{S})_G$ in G for all variables X, Y, over all sets $\mathbf{S}$. Therefore CB methods focus on pairwise conditional (in)dependencies of the form $X \not\perp\!\!\!\perp Y \mid \mathbf{S}$. We also follow this approach throughout the paper.

The Data Model. Our model follows Gold's paradigm which does not specify how linguistic data—for us, observed (in)dependencies—are generated. In practice, a BN learner obtains a random sample $\mathbf{d}$ drawn from the data generating joint distribution over the variables $\mathbf{V}$, and applies a suitable statistical criterion to decide if a dependency $X \not\perp\!\!\!\perp Y|\mathbf{S}$ holds [31], [33, Sec.4]. Many CB systems use a statistical test to answer queries to a dependency oracle: given a query "Does $X \not\perp\!\!\!\perp Y|\mathbf{S}$ hold?", the system answers "yes" if the test rejects the hypothesis $X \perp\!\!\!\perp Y|\mathbf{S}$, and "no" otherwise. The assumption that this procedure yields correct results is called the assumption of valid statistical testing [6, Sect.6.2]. Compared to this assumption, our model of learning from conditional dependencies (positive data) is more realistic in two respects. First, the model assumes only that *dependency information* is available, but does not rely on *independence* data. In fact, many statisticians hold that no independence conclusion should be drawn when a statistical significance test fails to reject an independence hypothesis [9]. Second, the dependency learning model does not assume that the dependency information is supplied by an oracle all at once, but explicitly considers learning in a setting where more information becomes available as the sample size increases. Our model still assumes that a statistically significant correlation does not disappear as the sample size increases. The extent to which this assumption is plausible depends on the testing strategy that extracts correlations from the given samples. The most common approach in constraint-based methods is to employ a fixed conservative significance level (e.g., $\alpha = 0.1\%$ [31, Ch.5], [8], [33]) for any sample size; with this kind of testing strategy, our assumption that the store of observed correlations grows monotonically is quite plausible. In fact, the results in this paper generalize to a model in which correlations may be taken back at later stages of learning, as long as there is a bound on the number of retractions: it can be shown that the mind change optimal learner in the generalized model conjectures that no dependencies will be retracted in the future, and then follows the output of the mind change optimal learner studied in this paper.

Although we have argued that our new model of learning Bayes nets from conditional dependencies is more realistic than existing models based on independency information, in this paper we also study learning from independencies, for two reasons. (1) From a mathematical point of view, it is natural to consider learning from both positive and negative data and examine the similarities and differences between them in the domain of Bayes net learning. (2) Machine learning researchers have extensively researched and widely applied methods based on independency data [31, 32, 3, 33].

Output of the Learner. In our model, a learner outputs one of the possible hypotheses (languages) or the vacuous conjecture ?. While the use of a ? for "no guess" is standard in learning theory, it supplies a user with no information about what can be inferred from the data. One approach to this issue in BN constraint-based learning is to indicate as part of the output that certain aspects of the BN model are uncertain. For instance, the CPC algorithm marks an edge $A - B$ if the results of statistical testing are ambiguous about the orientation of

the edge [27]. A general formulation of this approach is to allow the learner to output a *set* of hypotheses, rather than either a single hypothesis or ?. In the limit the learner has to converge to a singleton comprising the correct pattern. In the set learning model, a learner Ψ changes its mind at some nonempty finite sequence $\sigma \in \text{SEQ}$ if $\Psi(\sigma) \not\subseteq \Psi(\sigma^-)$. [18] defines the set learning model and shows that the results in this paper carry over to it.

Since the set of dependency relations $\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}$ constitutes a language collection in the sense of the Gold paradigm, we can employ standard identification criteria to analyze this learning problem; similarly for the set $\mathcal{L}^{I}_{\mathbf{V}}$ of independency relations. General results from learning theory that hold for any language collection are thus applicable to learning Bayes net structure from either dependency or independency information. We begin by applying a fundamental result in Bayes net theory to determine the mind change complexity of the problem.

4. The Mind Change Complexity of Learning Bayes Net Structure

Following Angluin [1, Condition 3] and Shinohara [30], we say that a class of languages $\mathcal{L}$ has **finite thickness** if the set $\{L \in \mathcal{L} : s \in L\}$ is finite for every string or evidence item $s \in \bigcup \mathcal{L}$. For language collections with finite thickness, their mind change complexity is determined by a structural feature called the inclusion depth [17, Def.6.1].

Definition 6. *Let $\mathcal{L}$ be a language collection and L be a language in $\mathcal{L}$. The **inclusion depth** of L in $\mathcal{L}$ is the size n of the largest index set $\{L_i\}_{1 \leq i \leq n}$ of distinct languages in $\mathcal{L}$, such that*

$$L \subset L_1 \subset \cdots \subset L_i \subset \cdots \subset L_n.$$

*The **inclusion depth** of $\mathcal{L}$ is the maximum of the inclusion depths of languages in $\mathcal{L}$.*

The next proposition establishes the connection between inclusion depth and mind change complexity. It follows immediately from the general result for ordinal mind change bounds established in [17, Prop. 6.1].

Proposition 7. *Let $\mathcal{L}$ be a language collection with finite thickness. Then there is a learner Ψ that identifies $\mathcal{L}$ with mind change bound k $\iff$ the inclusion depth of $\mathcal{L}$ is at most k.*

Since we are considering Bayes nets with finitely many variables, the dependency space $\mathcal{D}_{\mathbf{V}}$ is finite, so the language collection $\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}$ containing all BN-dependency relations is finite and therefore $\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}$ has finite thickness; similarly the independency space $\mathcal{I}_{\mathbf{V}}$ has finite thickness. Hence we have the following corollary.

Corollary 8. *Let $\mathbf{V}$ be a set of variables. There exists a learner Ψ that identifies a BN dependency relation from $\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}$, respectively a BN independency relation from $\mathcal{L}^{\mathcal{I}}_{\mathbf{V}}$, with mind change bound k $\iff$ the inclusion depth of $\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}$, respectively $\mathcal{L}^{\mathcal{I}}_{\mathbf{V}}$, is at most k.*

A fundamental result in Bayes net theory allows us to determine the inclusion depth of a dependency relation in $\mathcal{L}^D_{\mathbf{V}}$ or an independency relation in $\mathcal{L}^I_{\mathbf{V}}$. An edge $A \to B$ is **covered** in a DAG G if the parents of B are exactly the parents of A plus A itself (see Figure 2). The operation that reverses the direction of the arrow between A and B is a **covered edge reversal**. The following theorem was conjectured by Meek [20] and proven by Chickering [4, Thm.4].

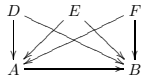

Figure 2: Edge $A \to B$ is covered, whereas $D \to A$ is not covered.

Theorem 9 (Chickering-Meek)**.** *Let G and H be two DAGs over the same set of variables $\mathbf{V}$. Then $\mathcal{D}_G \subseteq \mathcal{D}_H \iff$ the DAG H can be transformed into the DAG G by repeating the following two operations: (1) covered edge reversal, and (2) single edge deletion.*

The next corollary characterizes the inclusion depth of a BN independence relation $\mathcal{I}_G$ for a graph G in terms of a simple syntactic feature of G, namely the number of adjacencies in G. For a BN dependence relation $\mathcal{D}_G$, the corresponding feature is the number of adjacencies *not* in G.

Proposition 10. *Let $G = (\mathbf{V}, E)$ be a Bayes net structure.*

1. *The inclusion depth of the BN-dependence relation $\mathcal{D}_G$ equals $\binom{|\mathbf{V}|}{2} - |E|$, the number of adjacencies* not *in G. So the dependency relation defined by the totally disconnected network has inclusion depth $\binom{|\mathbf{V}|}{2}$, and the dependency relation defined by the complete network has inclusion depth* 0.

2. *The inclusion depth of the BN-independency relation $\mathcal{I}_G$ equals $|E|$, the number of adjacencies in G. So the independency relation defined by the totally disconnected network has inclusion depth 0, and the independency relation defined by the complete network has inclusion depth $\binom{|\mathbf{V}|}{2}$.*

Proof. Part 2 follows from Part 1, since an inclusion chain $\mathcal{D}_{G_1} \subset \cdots \subset \mathcal{D}_{G_k}$ of BN-dependency relations corresponds to an inclusion chain $\mathcal{I}_{G_k} \subset \cdots \subset \mathcal{I}_{G_1}$ of the complementary independency relations. To establish Part 1, we use downward induction on the number of edges n in graph G. Let $N = \binom{|\mathbf{V}|}{2}$. Base case: $n = N$. Then G is a complete graph, so $\mathcal{D}_G$ contains all dependency statements in the statement space $\mathcal{D}_{\mathbf{V}}$, and therefore has 0 inclusion depth. Inductive step: Assume the hypothesis for $n + 1$ and consider a graph G with n edges. Add an edge to G to obtain a BN G' with $n + 1$ edges that is a supergraph of G'. Adding an edge can only increase the number of d-connecting paths between two given variables, so $\mathcal{D}_G \subseteq \mathcal{D}_{G'}$. Suppose that the edge added is a new link between nodes A and B that are not adjacent in G. By the d-separation Lemma 3, there is a separating set $\mathbf{S}$ such that A, B are d-separated by $\mathbf{S}$ in G, that is, $(A \perp\!\!\!\perp B|\mathbf{S})_G$. However in the graph G', the nodes A and B

are adjacent, so they are not d-separated by any set, and we have $(A \not\perp\!\!\!\perp B|\mathbf{S})_{G'}$. So G' entails strictly more dependencies than G, that is,

$$\mathcal{D}_G \subset \mathcal{D}_{G'}.$$

By inductive hypothesis, there is an inclusion chain

$$\mathcal{D}_{G'} \subset \mathcal{D}_{G_1} \cdots \subset \mathcal{D}_{G_{N-(n+1)}}$$

consisting of BN dependency relations. Hence the inclusion depth of G is at least $N-(n+1)+1 = N-n$.

To show that the inclusion depth of G is exactly $N-n$, consider any inclusion chain

$$\mathcal{D}_G \subset \mathcal{D}_H \subset \cdots \subset \mathcal{D}_{\mathbf{V}}. \tag{1}$$

Theorem 9 implies that graph G can be obtained from H with covered arc reversals and/or edge deletions. If G can be obtained from H with covered arc reversals only, then H can be likewise obtained from G, since covered edge reversals are symmetric, and so by Theorem 9 we would have $\mathcal{D}_G = \mathcal{D}_H$, contradicting the choice of H. So H has at least one more edge than G and thus H has at least $n+1$ edges. Applying the inductive hypothesis to H, it follows that the inclusion chain

$$\mathcal{D}_H \subset \cdots \subset \mathcal{D}_{\mathbf{V}}$$

has length at most $N-(n+1)$, so the inclusion chain (1) has at most $N-(n+1)+1 = N-n$ members. ■ Propositions 7 and 10 imply that the mind change complexity of identifying a Bayes Net structure over variables $\mathbf{V}$ is given by the maximum number of edges over $\mathbf{V}$.

Corollary 11. *For any set of variables* $\mathbf{V}$*, the inclusion depth of* $\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}$ *and the inclusion depth of* $\mathcal{L}^{\mathcal{I}}_{\mathbf{V}}$ *is* $\binom{|\mathbf{V}|}{2}$*. So the mind change complexity of identifying the correct Bayes Net structure from dependency data or from independency data is* $\binom{|\mathbf{V}|}{2}$*.*

The next section characterizes the BN learning algorithms that achieve optimal mind change performance.

5. Mind-Change Optimal Learners for Bayes Net Structure

We analyze mind-change optimal algorithms for identifying Bayes net structure. The intuition underlying mind-change optimality is that a learner that is efficient with respect to mind changes minimizes mind changes not only globally in the entire learning problem, but also locally in subproblems after receiving some evidence [17, 16, 14]. We formalize this idea as in [17, Def.2.3]. If a mind change bound exists for $\mathcal{L}$ given σ, let $\mathrm{MC}_{\mathcal{L}}(\sigma)$ be the least k such that $\mathcal{L}$ is identifiable with k mind changes given σ. For example, given a sequence σ of

dependencies, let $G = (\mathbf{V}, E)$ be a BN that satisfies the dependencies in σ with a minimum number of edges. Then the mind change complexity $\mathrm{MC}_{\mathcal{L}^{\mathcal{D}}_{\mathbf{V}}}(\sigma)$ is $\binom{|\mathbf{V}|}{2} - |E|$. Mind change optimality requires that a learner should succeed with $\mathrm{MC}_{\mathcal{L}}(\sigma)$ mind changes after each data sequence σ.

Definition 12 (based on Def.2.3 of [17])**.** *A learner Ψ is **strongly mind-change optimal** (SMC-optimal) for $\mathcal{L}$ if for all data sequences σ the learner Ψ identifies $\mathcal{L}$ given σ with at most $\mathrm{MC}_{\mathcal{L}}(\sigma)$ mind changes.*

The next proposition characterizes SMC-optimal learners for language collections with finite inclusion depth. It follows from the general characterization of SMC-optimal learners for all language collections established in [17, Prop.4.1].

Proposition 13. *Let Ψ be a learner that identifies a language collection $\mathcal{L}$ with finite inclusion depth. Then Ψ is SMC-optimal for $\mathcal{L}$ if and only if for all data sequences σ: if $\Psi(\sigma) \neq ?$, then $\Psi(\sigma)$ is the unique language consistent with σ that maximizes inclusion depth.*

Applying the proposition to Bayes net learners yields the following corollary.

Corollary 14. *Let Ψ be a Bayes net learner that identifies the correct Bayes net pattern for a set of variables $\mathbf{V}$ from dependency data. The learner Ψ is SMC-optimal for $\mathcal{L}^{D}_{\mathbf{V}}$ $\iff$ for all dependency sequences σ, if the output of Ψ is not ?, then Ψ outputs a uniquely edge-minimal pattern for the dependencies $\mathcal{D} = \mathrm{content}(\sigma)$.*

It is easy to implement a slow SMC-optimal BN learner. For example, for a given set of dependencies $\mathcal{D}$ it is straightforward to check if there is a pattern G that covers exactly those dependencies (i.e., $\mathcal{D}_G = \mathcal{D}$). So an SMC-optimal learner could output a pattern G if there is one that matches the observed dependencies exactly, and output ? otherwise. But such a slow learner requires exponentially many dependency statements as input before it conjectures a graph. There are SMC-optimal learners that produce a guess faster; in fact, using Gold's notion of "uniformly faster", we can show that there is a unique fastest SMC-optimal learner. Gold proposed the following way to compare the convergence speed of two learners [10, p. 462].

Definition 15. *Let $\mathcal{L}$ be a language collection.*

1. *The convergence time of a learner Ψ on text T is defined as* $\mathrm{CP}(\Psi, T) \equiv$ *the least time m such that $\Psi(T[m]) = \Psi(T[m'])$ for all $m' \geq m$.*
2. *A learner Ψ identifies $\mathcal{L}$ uniformly faster than learner Φ $\iff$*
   (a) *for all languages $L \in \mathcal{L}$ and all texts T for L, we have* $\mathrm{CP}(\Psi, T) \leq \mathrm{CP}(\Phi, T)$*, and*
   (b) *for some language $L \in \mathcal{L}$ and some text T for L, we have* $\mathrm{CP}(\Psi, T) < \mathrm{CP}(\Phi, T)$.

For a language collection $\mathcal{L}$ with finite inclusion depth, Proposition 13 implies that if there is no language L that uniquely maximizes inclusion depth given

σ, then a learner that is SMC-optimal outputs ? on σ. Intuitively, the fastest SMC-optimal learner delays making a conjecture no longer than is necessary to meet this condition. Formally, this learner is defined as follows for all sequences $\sigma \in \mathrm{SEQ}(\mathcal{L})$:

$$\Psi^{\mathcal{L}}_{\mathrm{fast}}(\sigma) = \begin{cases} ? & \text{if no language uniquely maximizes inclusion depth given } \sigma \\ L & \text{if } L \in \mathcal{L} \text{ uniquely maximizes inclusion depth given } \sigma. \end{cases}$$

The next observation asserts that $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ is the fastest SMC-optimal method for a given language collection $\mathcal{L}$.

Observation 16. *Let $\mathcal{L}$ be a language collection with finite inclusion depth. Then $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ is SMC-optimal and identifies $\mathcal{L}$ uniformly faster than any other SMC-optimal learner for $\mathcal{L}$.*

Proof. It is easy to see that $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ identifies $\mathcal{L}$, so Proposition 13 implies that $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ is SMC-optimal. Let $\Psi \neq \Psi^{\mathcal{L}}_{\mathrm{fast}}$ be any other SMC-optimal learner that identifies $\mathcal{L}$. By Proposition 13, if $\Psi(\sigma) \neq ?$, then $\Psi(\sigma) = \Psi^{\mathcal{L}}_{\mathrm{fast}}(\sigma)$. So Ψ does not converge faster than $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ on any text. Therefore for any language $L \in \mathcal{L}$ and text T for L, we have

$$\mathrm{CP}(\Psi^{\mathcal{L}}_{\mathrm{fast}}, T) \leq \mathrm{CP}(\Psi, T).$$

Let $\sigma \in \mathrm{SEQ}(\mathcal{L})$ be a data sequence such that $\Psi(\sigma) \neq \Psi^{\mathcal{L}}_{\mathrm{fast}}(\sigma) = L$. Since both Ψ and $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ are SMC-optimal, Proposition 13 implies that L uniquely maximizes inclusion depth given σ. So $\Psi^{\mathcal{L}}_{\mathrm{fast}}(\sigma) = L$ and $\Psi(\sigma) = ?$. Now let $T_L \supset \sigma$ be any text for L extending σ. It is easy to see that the language L uniquely maximizes inclusion depth on any data sequence σ' with $\sigma \subseteq \sigma' \subset T_L$. So

$$\mathrm{CP}(\Psi^{\mathcal{L}}_{\mathrm{fast}}, T_L) \leq |\sigma|,$$

and clearly

$$\mathrm{CP}(\Psi, T_L) > |\sigma|$$

since $\Psi(\sigma) = ?$. Thus $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ identifies $\mathcal{L}$ uniformly faster than Ψ, and in general faster than any other SMC-optimal learner. ■

Observation 16 leads to the following algorithm for identifying a BN pattern from dependency data.

Corollary 17. *Let $\mathbf{V}$ be a set of variables. For a given sequence of dependencies σ, the learner $\Psi^{\mathcal{D}}_{\mathrm{fast}}$ outputs ? if there is more than one edge-minimal pattern that covers the dependencies in σ, and otherwise outputs a uniquely edge-minimal pattern for the dependencies $\mathcal{D} = \mathrm{content}(\sigma)$. The learner $\Psi^{\mathcal{D}}_{\mathrm{fast}}$ is SMC-optimal and identifies the correct pattern uniformly faster given dependency data than any other SMC-optimal BN structure learner.*

The corollary shows that the criteria of mind-change optimality and convergence speed determine a unique, natural and novel method for learning Bayes net structure. The next example illustrates this method.

Example 18. *Let* $\mathbf{V} = \{A, B, C, D\}$. *If*

$$\begin{aligned}\sigma = (&B \not\perp\!\!\!\perp D, B \not\perp\!\!\!\perp D|\{A\}, B \not\perp\!\!\!\perp D|\{C\},\\ &C \not\perp\!\!\!\perp D, C \not\perp\!\!\!\perp D|\{A\}, C \not\perp\!\!\!\perp D|\{B\},\\ &B \not\perp\!\!\!\perp C|\{D\}, B \not\perp\!\!\!\perp C|\{A, D\}),\end{aligned}$$

then $\Psi^{\mathbf{V}}_{\text{fast}}$ *outputs the graph shown in Figure 3(a), for the following reasons. According to Corollary 17, maximizing inclusion depth given dependency data is equivalent to minimizing the number of edges in the output graph. In any edge-minimal graph* G *consistent with the data* σ*, nodes* B *and* D *are adjacent: if they are not adjacent in some graph* G*, then by Lemma 3 there is a d-separating set* $\mathbf{S}$ *such that* $(B \perp\!\!\!\perp D|\mathbf{S})_G$*. According to the data,* B *and* D *are dependent given any proper subset of* $\{A, C\}$*, so we have* $(B \not\perp\!\!\!\perp D|\{A, C\})_G$*. The skeleton of any such graph* G' *must contain undirected paths* $B - A - D$ *and* $B - C - D$*, that is, all paths between* B *and* D *are blocked by* A *and* C*, but there is an unblocked path given just* A *and an unblocked path given just* C*. But any such graph* G *contains 4 edges, whereas the graph shown in Figure 3(a) entails the dependencies in* σ *with just two edges. So nodes* B *and* D *are adjacent in any edge-minimal graph consistent with* σ*. The same argument holds for nodes* C *and* D*, so any edge-minimal graph consistent with* σ *contains a triple* $B - D - C$*. To entail the observed dependence* $B \not\perp\!\!\!\perp C|\{D\}$*, that triple must be oriented as* $B \rightarrow D \leftarrow C$*; the only alternative is to add an edge between* B *and* C*, but this fails to minimize the number of edges. So the only pattern with only two edges that is consistent with the observed dependencies* σ *is the one shown in Figure 3(a).*

Now suppose after some time, we have accumulated more data, and obtain the following data sequence

$$\begin{aligned}\sigma' = \sigma \cup &\\ &A \not\perp\!\!\!\perp B, A \not\perp\!\!\!\perp B|\{C\}, A \not\perp\!\!\!\perp B|\{D\},\\ &A \not\perp\!\!\!\perp C, A \not\perp\!\!\!\perp C|\{B\}, A \not\perp\!\!\!\perp C|\{D\}, B \not\perp\!\!\!\perp C),\end{aligned}$$

then the optimal learner $\Psi^{\mathbf{V}}_{\text{fast}}$ *outputs the graph shown in Figure 3(b), for the following reasons. Similar to the analysis of the previous data, any edge-minimal graph must have* A *adjacent to* B*: For if* A *is not adjacent to* B *in a graph* G *consistent with* σ'*, then the skeleton of* G *must contain two paths* $A - C - B$ *and* $A - D - B$*, and if* G *has the minimum number of 4 necessary edges, these are all the links in* G*. In any DAG with such a skeleton, there is at least one collider (else a cycle would occur). Suppose without loss of generality that* B *is a collider, so the graph contains a triple* $C \rightarrow B \leftarrow D$*. But then either the dependence* $C \not\perp\!\!\!\perp D|A$ *or the dependence* $C \not\perp\!\!\!\perp D$ *is not entailed by* G*. So any graph with only 4 edges that is consistent with* σ' *has the same skeleton as that shown in Figure 3(b). The only remaining question is the placement of colliders. If both* A *and* D *are colliders, the dependence* $B \not\perp\!\!\!\perp C$ *fails. If neither is a collider, the dependence* $B \not\perp\!\!\!\perp C|\{A, D\}$ *fails. If* A *is a collider and* D *is not, the dependence* $B \not\perp\!\!\!\perp C|\{D\}$ *fails. So the only possibility is to have* D *as a*

collider and A not a collider, which results in the pattern shown in Figure 3(b).

Figure 3: To illustrate SMC-optimal learning from dependency data. (a) The output of the optimal learner $\Psi^{\mathbf{V}}_{\text{fast}}$ on example data σ. (b)The output of the optimal learner $\Psi^{\mathbf{V}}_{\text{fast}}$ on example data σ'.

The next corollary characterizes the fastest mind-change optimal learner for independency data.

Corollary 19. *Let* $\mathbf{V}$ *be a set of variables. For a given sequence of independencies* σ*, the learner* $\Psi^{\mathcal{I}}_{\text{fast}}$ *outputs ? if there is more than one edge-maximal pattern that covers the independencies in* σ*, and otherwise outputs a uniquely edge-maximal pattern for the independencies* $\mathcal{I} = \text{content}(\sigma)$*. The learner* $\Psi^{\mathcal{I}}_{\text{fast}}$ *is SMC-optimal and identifies the correct pattern uniformly faster given independency data than any other SMC-optimal BN structure learner.*

Intuitively, the learner $\Psi^{\mathcal{I}}_{\text{fast}}$ removes as many potential adjacencies as are necessary to entail the independencies given in the data. One of the most prominent constraint-based algorithms, the PC algorithm [31, Ch.5.4.2], can be seen as a heuristic implementation of this idea: Basically, the PC algorithm starts with the completely connected network over n variables, and then successively considers independencies of the form $X \perp\!\!\!\perp Y | \mathbf{S}$, for $|\mathbf{S}| = 0, 1, \ldots, n$, removing links between the conditionally independent variables X and Y. Examples of the kind of Bayes net structure that result from this algorithm are given in [31]. The remainder of the paper analyzes the run-time complexity of the optimal $\Psi^{\mathcal{D}}_{\text{fast}}$ and $\Psi^{\mathcal{I}}_{\text{fast}}$ methods; we show that computing the output of these learners is NP-hard (assuming that P = RP). The proof gives two many-one reductions of the problem of deciding the existence of a unique exact cover by 3-sets, to the problems of computing the output of the fastest mind change optimal learner from independency respectively dependency data.

6. Computational Complexity of Fast Mind-Change Optimal Identification of Bayes Net Structure from Independency Data

Because the NP-hardness proofs for the optimal learners are quite different for dependency and independency data, we divide the analysis into different

sections. We begin with the setting of independency data, whose NP-hardness proof is much simpler. General complexity theory concepts and notation were reviewed in Section 2.3.

We describe the standard approach of analyzing the complexity of constraint-based learners in the Bayes net literature. As with any run-time analysis, an important issue is the representation of the input to the algorithm. The most straightforward approach for our learning model would be to take the input as a list of (in)dependencies, and the input size to be the size of that list. However, in practice CB learners do not receive an explicitly enumerated list of (in)dependencies, but rather they have access to a statistical oracle (cf. Section 3.2). Enumerating relevant (in)dependencies through repeated queries to the oracle is part of the computational task of a CB learner. Accordingly, the standard complexity analysis takes an (in)dependency oracle and a set of variables as the input to the learning algorithm (e.g., [5, Def.12],[2]). The oracle is assumed to be represented syntactically in a reasonably concise way. For example, it may be a Turing Machine that computes the characteristic function of a given dependency relation $\mathcal{D}$.

Definition 20. *An independency oracle O for a variable set $\mathbf{V}$ is a function that takes as input independency queries from the independency space $\mathcal{I}_{\mathbf{V}}$ and returns, in constant time, either "yes" or "?".*

The independency relation associated with oracle O is given by $\mathcal{I}_O = \{X \not\perp\!\!\!\perp Y | \mathbf{S} \in \mathcal{I}_{\mathbf{V}} : O \text{ returns "yes" on input } X \perp\!\!\!\perp Y | \mathbf{S}\}$. The oracle represents the independencies observed on a finite data sample, which may be incomplete; that is, there need not be any graph G such that $\mathcal{I}_G = \mathcal{I}_O$.

Remark. Our model of learning Bayes net structure can be reformulated in terms of a sequence of oracles: Instead of a complete sequence of independence statements for an independence relation $\mathcal{I}_G$, the learner could be presented with a sequence of independency oracles $O_1, O_2, \ldots, O_n, \ldots$ such that $\mathcal{I}_{O_i} \subseteq \mathcal{I}_{O_{i+1}}$ and $\bigcup_{i=1}^{\infty} \mathcal{I}_{O_i} = \mathcal{I}_G$. The mind change and convergence time results remain the same in the independency oracle model.

We say that a pattern G is an **I-cover** of a set of independencies $\mathcal{I}$ if G entails all the independencies in $\mathcal{I}$ (i.e., $\mathcal{I} \subseteq \mathcal{I}_G$). Computing the conjectures of the learner $\Psi^{\mathcal{I}}_{\text{fast}}$ poses the following computational problem.

Unique Maximal I-cover

Input A set of variables $\mathbf{V}$ and an independency oracle O for $\mathbf{V}$.

Output If there is a *unique* DAG pattern G that entails the independencies in O with a maximal number of edges, output G. Otherwise output ?.

This is a function maximization problem; the corresponding decision problem is the following.

Unique I-cover

Instance A set of variables $\mathbf{V}$, an independency oracle O for $\mathbf{V}$, and a bound k.

Question Is there a DAG pattern G such that: G entails the independencies in O, every other DAG pattern G' entailing the independencies in O has fewer edges than G, and G has at least k edges?

Clearly an efficient algorithm for the function maximization problem yields an efficient algorithm for UNIQUE I-COVER. We will show that UEC3SET reduces to UNIQUE I-COVER. We also give an upper bound on the problem complexity in terms of oracle computations. Recall that SAT is the NP-complete satisfiability problem and $\mathrm{FP}^{(\mathrm{NP}^{\mathrm{NP}})}$ is the class of functions computable in polynomial time given an oracle for decision problems in $\mathrm{NP}^{\mathrm{NP}}$.

Theorem 21. *The computational complexity of* UNIQUE MAXIMAL I-COVER.

1. UNIQUE MAXIMAL I-COVER *is in* $\mathrm{FP}^{(\mathrm{NP}^{\mathrm{NP}})}$.
2. $\mathrm{SAT} \leq_{\mathrm{RP}} \mathrm{UEC3SET} \leq_{\mathrm{P}}$ UNIQUE I-COVER $\leq_{\mathrm{P}}$ UNIQUE MAXIMAL I-COVER. *So* UNIQUE MAXIMAL I-COVER *is* NP*-hard provided that* $\mathrm{P} = \mathrm{RP}$.

Proof. Part 1: We specify a program that computes the output of the fastest SMC-optimal learner $\Psi^{\mathcal{I}}_{\mathrm{fast}}$ given a set of n input nodes, an independency oracle O, and an oracle for computational problems in $\mathrm{NP}^{\mathrm{NP}}$. A pattern G is an I-cover of O just in case the answer to the question "is there a dependency $X \perp\!\!\!\perp Y|\mathbf{S}$ such that $X \perp\!\!\!\perp Y|\mathbf{S} \in (I_O - I_G)$" is no. So an oracle for NP decides in constant time whether a pattern G is an I-cover of O.

1. Find the maximum number of edges k, for $k = 1, \ldots, \binom{n}{2}$, such that there is a pattern G that is an I-cover of O. Given an NP-oracle for deciding whether G is an I-cover of O, for a given k this query can be answered in polynomial time with a nondeterministic computation. Using binary search, this step therefore can be carried out with $O(log(n))$ queries to the $\mathrm{NP}^{\mathrm{NP}}$ oracle.
2. Having determined the maximum number k of edges, ask "are there two distinct patterns G, G' such that both G and G' are I-covers of O and G and G' contain k edges?". Given an NP-oracle for deciding whether G and G' are I-covers of O, this query can be answered in polynomial time with a nondeterministic computation, hence with a single query to the $\mathrm{NP}^{\mathrm{NP}}$ oracle. If the answer is "yes", output ?, since there is no unique I-cover with a maximum number of edges. If the answer is "no", there is a unique I-cover with a maximum number of edges; continue to the next step.
3. Using a standard method, construct an output pattern R as follows. For each pair of nodes X, Y and each of the possible link types $X - Y, X \rightarrow Y, X \leftarrow Y$, ask "is there a pattern G with k edges that contains the given link type between X and Y and is an I-cover of I_O". As in the previous step, this query can be answered in constant time with an $\mathrm{NP}^{\mathrm{NP}}$ oracle for a given pair of nodes X, Y and type of link between them. Since the edge-maximizing I-cover is unique, the answer is "yes", for at most one

of these link types. If the test indicates a link of a certain type between X and Y, add the link to R. This requires $O(\binom{n}{2})$ queries to the $\mathrm{NP}^{\mathrm{NP}}$ oracle.

This procedure returns the output of the fastest SMC-optimal learner $\Psi^{\mathcal{I}}_{\mathrm{fast}}$ with total runtime bounded by $O(\binom{n}{2})$.

Part 2: We give a reduction from UNIQUE X3SET to UNIQUE I-COVER WITH AT LEAST K EDGES. Consider an instance of UEC3SET with sets universe U of size $|U| = 3m$, and $c_1, \ldots, c_p$, where $|c_i| = 3$ for $i = 1, \ldots, p$ and $U = \cup_{i=1}^{m} c_i$. Define the following set $\mathbf{V}$ of variables.

1. For every set c_i, a *set variable* C_i.
2. A *root variable* R.

We write x for the element corresponding to node X, and similarly c for the set corresponding to node C. Set the bound $k = m$. The following program implements an independency oracle O over the variables $\mathbf{V}$, in time polynomial in the size of the given UEC3SET instance.

Definition of Independency Oracle

Input An independency query $V_1 \perp\!\!\!\perp V_2 | \mathbf{S}$.

Output Oracle Clauses

1. If $V_1 = C_i, V_2 = C_j$, and $\mathbf{S} = \emptyset$, then return "independent".
2. If $V_1 = C_i, V_2 = C_j, \mathbf{S} = \{R\}$ and $c_i \cap c_j \neq \emptyset$, then return "independent".
3. In all other cases, return ?.

We use the terms adjacency, edge, and link interchangeably. Let $\mathcal{I}_O$ be the set of independencies associated with oracle O. We argue that every I-cover for $\mathcal{I}_O$ with at least m adjacencies corresponds to an exact set cover for the given instance of UEC3SET, and vice versa.

Suppose that G is an I-cover for $\mathcal{I}_O$ with at least m adjacencies. Clause 1 implies that all adjacencies are of the form $C - R$: the only other possibility is an adjacency $C - C'$ between set variables, but then C and C' are d-connected in G given the empty set, whereas Clause 1 requires that they be d-separated given the empty set. The case with $m = 1$ is trivial; for $m > 1$, Clause 1 implies that the adjacencies are directed as $C \to R$: for suppose otherwise. Then there is a link of the form $C \leftarrow R$. Since $m > 1$, there is at least one other link $C' - R$ with $C \neq C'$. Now R is not a collider on the path $C \leftarrow R - C'$, so C and C' are d-connected in G given the empty set, whereas Clause 1 requires that they be d-separated given the empty set. Given that R is therefore a collider on any path $C - R - C'$, it follows that $C \not\perp\!\!\!\perp C' | \{R\}$ whenever C, C' are neighbors of R. The contrapositive of Clause 2 says that if $C \not\perp\!\!\!\perp C' | \{R\}$, then $c \cap c' = \emptyset$. Thus the neighbors of R correspond to mutually disjoint sets, so there are exactly m neighbors of R forming an exact set cover.

Conversely, it is easy to see that if the collection $\{c^1, \ldots, c^m\}$ forms an exact cover of the universe U, then the graph containing all edges $C^i \rightarrow R$, for $i = 1, \ldots, m$, is an I-cover of $\mathcal{I}_O$. So there is a 1-1 onto mapping between I-covers of $\mathcal{I}_O$ with at least m edges and exact set covers for the given set cover instance. Hence there is a unique I-cover of $\mathcal{I}_O$ with at least m edges if and only if there is a unique exact set cover. ■

The gap between our upper and lower bounds is typical for problems in the higher levels of the polynomial hierarchy, where both lower bounds and completeness in a class are difficult to establish (see [23, Ch.17], especially the discussion of the Minimum Circuit problem, and [28, Ch.5], especially the discussion of UNIQUE SAT).

The next section establishes a corresponding upper and lower bound for the problem of computing the output of the fastest SMC-optimal learner from dependency data, which requires a much more difficult construction for the lower bound. The reason why the hardness proof for dependency data is much more complex than for independency data is that there are many ways to achieve d-connection between two variables A and B, whereas d-separation immediately entails the absence of a link between A and B.

7. Computational Complexity of Fast Mind-Change Optimal Identification of Bayes Net Structure from Dependency Data

The definitions for the complexity analysis for dependency data parallel those for independency data.

Definition 22. *A dependency oracle O for a variable set $\mathbf{V}$ is a function that takes as input dependency queries from the dependency space $\mathcal{D}_{\mathbf{V}}$ and returns, in constant time, either "yes" or "?".*

The dependency relation associated with oracle O is given by $\mathcal{D}_O = \{X \not\perp\!\!\!\perp Y | \mathbf{S} \in \mathcal{D}_{\mathbf{V}} : O \text{ returns "yes" on input } X \not\perp\!\!\!\perp Y | \mathbf{S}\}$. The oracle represents the dependencies observed on a finite data sample, which may be incomplete; that is, there need not be any graph G such that $\mathcal{D}_G = \mathcal{D}_O$. As with independency data, our model of learning Bayes net structure can be reformulated in terms of a sequence of dependency oracles, where instead of a complete sequence of dependence statements, the learner is presented with a sequence of dependency oracles converging to the target independency relation. The mind change and convergence time results remain the same in the dependency oracle model.

A pattern G is an **I-map** of a set of dependencies $\mathcal{D}$ if G entails all the dependencies in $\mathcal{D}$ (i.e., $\mathcal{D} \subseteq \mathcal{D}_G$, cf. [24]). Computing the conjectures of the learner $\Psi^{\mathcal{D}}_{\text{fast}}$ poses the following computational problem.

Unique Minimal I-map

Input A set of variables $\mathbf{V}$ and a dependency oracle O for $\mathbf{V}$.

Output If there is a *unique* DAG pattern G that entails the dependencies in O with a minimal number of edges, output G. Otherwise output ?.

The corresponding decision problem is the following.

Unique I-map

Instance A set of variables $\mathbf{V}$, a dependency oracle O for $\mathbf{V}$, and a bound k.

Question Is there a DAG pattern G such that: G covers the dependencies in O, every other DAG pattern G' entailing the dependencies in O has more edges than G, and G has at most k edges?

Clearly an efficient algorithm for the function minimization problem yields an efficient algorithm for UNIQUE I-MAP. We will show that UEC3SET reduces to UNIQUE I-MAP. We also give an upper bound on the problem complexity in terms of oracle computations.

Theorem 23. *The computational complexity of* UNIQUE MINIMAL I-MAP.

1. UNIQUE MINIMAL I-MAP *is in* $\mathrm{FP}^{(\mathrm{NP}^{\mathrm{NP}})}$.
2. $\mathrm{SAT} \leq_{\mathrm{RP}} \mathrm{UEC3SET} \leq_{\mathrm{P}}$ UNIQUE I-MAP $\leq_{\mathrm{P}}$ UNIQUE MINIMAL I-MAP. *So* UNIQUE MINIMAL I-MAP *is* NP*-hard provided that* $\mathrm{P} = \mathrm{RP}$.

Proof. Part 1 can be proven like Theorem 21(1). For Part 2, we give a reduction from UEC3SET to UNIQUE I-MAP. Consider an instance of UEC3SET with sets universe U of size $|U| = 3m$, and $c_1, \ldots, c_p$, where $|c_i| = 3$ for $i = 1, \ldots, p$ and $U = \cup_{i=1}^{m} c_i$. Define the following set $\mathbf{V}$ of variables.

1. For every set c_i, a *set variable* C_i.
2. For every element x_j of the universe U, a *member variable* X_j.
3. A *root variable* R.

We write x for the element corresponding to node X, and similarly c for the set corresponding to node C. Set the bound $k = 3p+m$. The following program M implements a dependency oracle O over the variables V, in time polynomial in the size of the given UEC3SET instance.

Definition of Dependency Oracle

Input A dependency query $V_1 \not\perp V_2 | \mathbf{S}$.

Output Oracle Clauses

1. If $V_1 = C_i, V_2 = X_j$, and $x_j \in c_i$, then return "dependent".
2. If $V_1 = X_i, V_2 = X_j$, and there is a set $c_k \supseteq \{x_i, x_j\}$ such that $C_k \in \mathbf{S}$, then return "dependent".
3. If $V_1 = R, V_2 = X_j, \mathbf{S} = \emptyset$ then return "dependent".
4. If $V_1 = R, V_2 = X_j, |\mathbf{S}| = 1$, and $\mathbf{S} \neq \{C\}$ where $x_j \in c$, then return "dependent".

5. In all other cases, return ?.

We argue that there is a unique exact set cover for an instance $\langle U, \{c_i\}\rangle$ iff there is a unique I-map with at most k edges for O. So if there were a polynomial time algorithm A for UNIQUE I-MAP, we could solve the UEC3SET instance in polynomial time by using the program M to "simulate" the oracle O and use A to solve the corresponding instance of UNIQUE I-MAP. Our proof strategy is as follows. Consider what we refer to as the *basic graph* for O, shown in Figure 4. The basic graph is also a pattern because all arrows correspond to unshielded colliders. We show that if there is a unique I-map G for O with at most k

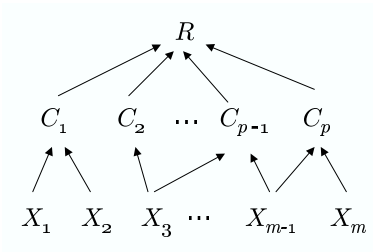

Figure 4: The basic graph for the NP-hardness proof. A set cover of size m corresponds to m edges of the form $C \rightarrow R$.

edges, then G is a subgraph of the basic graph, with possibly edges $C_i \rightarrow R$ missing for some sets c_i, such that the set of variables $\{C_1, C_2, \ldots, C_m\}$ with the edge $C_i \rightarrow R$ in G corresponds to an exact cover $\{c_1, \ldots, c_m\}$. Conversely, any unique exact cover corresponds to a subgraph of the basic graph in the same manner.

In the following arguments we use the terms adjacency, edge, and link interchangeably. For notation, we use an undirected link as in $A - B$ to indicate that nodes A and B are adjacent in a given graph, but the direction of the link is not determined by the context. The notation $A - \cdots - B$ indicates a path starting with A and ending in B where the direction of the links involving A and B is not determined. The notation $A - \cdots \rightarrow B$ indicates that the last edge points into B, etc. It is easiest to consider separately the constraints imposed by each clause of O. Let $\mathcal{D}_i$ be the *set of dependencies corresponding to Clause i*. For example, $D_1 = \{\langle C_i, X_j, \mathbf{S}\rangle : x_j \in c_i\}$. The implications of $\mathcal{D}_1$ are as follows.

Assertion 24. *Let DAG G be an I-map for $\mathcal{D}_1$. Then any two nodes X and C are adjacent whenever $x \in c$.*

Proof. The d-separation lemma 3 implies that if G entails the dependencies in $\mathcal{D}_1$, then x and C are adjacent whenever $x \in c$. ■

An implication of the constraints in $\mathcal{D}_2$ is that any two element nodes X_i, X_j with x_i, x_j contained in the same set c are adjacent to each other or have variable C as a common child (possibly both may be the case).

Assertion 25. *Let DAG G be an I-map for $\mathcal{D}_1 \cup \mathcal{D}_2$, and suppose that x_i, x_j*

are two elements of a set c. Then X_i and X_j are adjacent in G, or G contains a component $X_i \to C \leftarrow X_j$.

Proof. By Assertion 24, nodes X_i and X_j are respectively adjacent to C. If X_i and X_j are adjacent, then the assertion holds trivially. A basic fact about d-separation is that in a triple $X - Y - Z$ such that X and Z are not adjacent, the node Y is a collider if and only if all sets that contain Y d-connect X and Z [21, Lemma 2.5]. So the dependencies $\mathcal{D}_2$ require that C is a collider in G. ■

Clause 3 requires that every member variable X be d-connected to the root variable. The intuition behind our reduction is that the basic graph B contains the most edge-efficient way to achieve the connection because with just one edge $C \to R$ the graph d-connects three member variables at once. Since there are many ways to achieve d-connection in a graph, proving the correctness of this intuition is the main difficulty of our proof.

The bulk of the proof shows that any I-map for $\mathcal{D}_3$ can be transformed into a subgraph of the basic graph B without increasing the number of edges. This requires a number of intermediate results. We begin by establishing that in an I-map G of $\mathcal{D}_3$, all arcs originating in the root variable R can be reversed with the result G' still an I-map of $\mathcal{D}_3$.

Lemma 26. *Let DAG G be an I-map of $\mathcal{D}_3$. Let G' be the graph obtained by reversing all arcs of the form $R \to V$ to be $V \to R$. Then G' is an I-map of $\mathcal{D}_3$.*

Proof. First we argue that G' is acyclic. Suppose for contradiction that G' contains a directed cycle that is not contained in G. Then the cycle involves a reversed edge of the form $X \to R$, and we can write the cycle as $R \to Y \to \cdots \to R$. However, since all edges involving the root R are reversed in G', there is no edge $R \to Y$ in G'. So G' is acyclic since G is.

Second, we show that G' entails the dependencies in $\mathcal{D}_3$. Let X be any member variable and consider a path p in G of the form

$$p = R - \cdots - X$$

that d-connects R and X in G.

Case 1: The path p contains no reversed edge. Then the edges in p are the same in G as in G', and p d-connects R and X in G'.

Case 2: The path p contains some reversed edge. Let Z be the last such member. Then p is of the form

$$p = R - \cdots - Z - \cdots - X$$

where the final path segment $Z - \cdots - X$ contains no nodes with their edges changed. The following path p' d-connects R and X in G': set

$$p' = R \leftarrow Z - \cdots - X.$$

Clearly the node Z is not a collider in p', and so p' d-connects R and X in G' as the final segment $Z - \cdots - X$ is oriented the same in both graphs. So in either case, R and X are d-connected in G' and so G' entails $\mathcal{D}_3$. ■

The next assertion shows that if the root node is a sink in a graph G, then for any parent A of R, all edges pointing out of A except for $A \to R$ can be reversed.

Lemma 27. *Let DAG G be an I-map of $\mathcal{D}_3$. Suppose that the root variable R has outdegree 0 and that A is a parent of R in G. Let G' be the graph obtained by reversing all edges of the form $A \to B$ where $B \neq R$. Then G' is an I-map of $\mathcal{D}_3$.*

Proof. First we argue that G' is acyclic. Suppose for contradiction that G' contains a directed cycle that is not contained in G. Then the cycle involves a reversed edge and hence contains A, so without loss of generality, we may assume the cycle is of the form $A \to \cdots \to A$. But the only child of A in G' is the root R, which is a sink in G and hence in G'. So there is no directed path from R to A and hence no directed cycle in G' from A to A. Thus G' is acyclic since G is.

To see that G' entails the dependencies in D_3, let X be any member variable; since G entails the dependencies in $\mathcal{D}_3$, there is a path p in G of the form $R - \cdots - X$ that d-connects R and X in G. Suppose that path p uses an edge of the form $A \to B$ that is reversed in G'.

Case 1: p is of the form

$$X - \cdots - A \to B \to \cdots \to R.$$

Then the path

$$X - \cdots - A \to R$$

d-connects X and R in G'.

Case 2: p is of the form

$$X - \cdots \leftarrow B \leftarrow A - \cdots - R.$$

Then the path

$$X - \cdots \leftarrow B \leftarrow A \to R$$

introduces no new collider and thus d-connects X and R in G'. So all member variables X are d-connected to the root variable R in G' and G' is an I-map of the dependencies D_3. ■

The next assertion shows that if a node has two directed paths towards the root, then one of them is superfluous for entailing the dependencies in $\mathcal{D}_3$, and it is possible to delete an edge. This is a key fact for constraining the structure of edge-minimal graphs.

Lemma 28. *Let DAG G be an I-map of $\mathcal{D}_3$. Suppose that for some node X, there are two directed paths $X \to U_1 \to \cdots \to U_p \to R$ and $X \to W_1 \to \cdots \to W_q \to R$ where $U_1 \neq W_1$. Let G' be the graph obtained from G by deleting the edge $X \to U_1$. Then G' is an I-map of $\mathcal{D}_3$.*

Proof. Consider a path p in G of the form

$$p = R - \cdots - Y$$

that d-connects R and Y in G. If p does not use the edge $X \to U_1$, then p d-connects R and X in G'. Otherwise there are two cases.

Case 1: The path p is of the form

$$p = R - \cdots - X \to U_1 - \cdots - Y.$$

Then Y is d-connected to U_1 in G and hence in G'. Thus the path

$$Y - \cdots - U_1 \to \cdots \to U_p \to R$$

d-connects Y to R.

Case 2: The path p is of the form

$$p = R - \cdots - U_1 \leftarrow X - \cdots - Y.$$

Since $U_1 \neq W_1$, the path

$$X \to W_1 \to \cdots \to W_q \to R$$

exists in G'. So the path

$$Y - \cdots - X \to W_1 \to \cdots \to W_q \to R$$

d-connects Y to R.

So in either case, any member variable Y is d-connected to the root variable R in the graph G', and so G' is an I-map of $\mathcal{D}_3$. ■

Lemma 29. *Let DAG G be an I-map of $\mathcal{D}_3$. Suppose G contains an edge $A \to B$ where B is not an ancestor of R. Add an edge $B \to R$ and delete the edge $A \to B$. The resulting graph G' is an I-map of $\mathcal{D}_3$ that contains no more edges than G.*

Proof. Consider any path in G that d-connects a member variable X with root variable R and uses the edge $A \to B$. Since B is not an ancestor of R, the path must be of the form

$$X - \cdots \leftarrow B \leftarrow A - \cdots - R.$$

So in G there is a path $X - \cdots \leftarrow B$ that does not involve the edge $A \to B$, hence in G' we have the path

$$X - \cdots \leftarrow B \to R$$

d-connecting X and R. So any member variable d-connected to R in G is also d-connected in G', which establishes the lemma. ■

The previous lemmas showed that an I-map G of $\mathcal{D}_3$ can be transformed in various ways that bring it closer to the basic graph while still remaining

an I-map of $\mathcal{D}_3$. The next key Assertion 30 shows that an I-map G of $\mathcal{D}_3$ of minimum size must not contain certain types of edges outside of the basic graph; we refer to these as 'inefficient' edges. Formally, say that an edge is *inefficient* in DAG G if and only if it has one of the following forms: (1) $X - Y$ where X and Y are member variables, (2) $X - R$ where X is a member variable, (3) $X \to C$ where $x \notin c$. An edge is efficient if it is not inefficient. So the efficient edges have one of the following forms: (1) $C - R$, or (2) $C - C'$, or (3) $C - X$ with $x \in c$, or (4) $X \leftarrow C$ where $x \notin c$; here C, C' are set variables, and X is a member variable. We refer to an adjacency of the form $X \leftarrow C$ where $x \notin c$ as an *efficient set-nonmember link.* We show that the presence of an inefficient edge in an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ requires more than k edges total in G, by the following construction: First, if an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ contains no efficient set-nonmember links, but contains inefficient edges, then it has more than k edges total. Second, we can replace efficient set-nonmember links with other links, without reducing the number of inefficient edges, until the first case applies.

Assertion 30. *Let G be an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. If G contains inefficient edges, then G contains more than $k = 3p + m$ edges.*

Proof. The proof is by induction on the number of efficient set-nonmember links, denoted as a.

Base Case, $a = 0$. We proceed with a further induction on the number of inefficient edges, denoted as i. Base Case, $i = 1$. Let $A \to B$ be the only inefficient edge in G, and suppose for contradiction that G contains no more than $3p + m$ edges. Assertion 24 implies that G contains $3p$ adjacencies of the form $X - C$ where $x \in c$, which are efficient. Since G contains also an inefficient edge, the number of remaining efficient edges not of the form $X - C$ is at most

$$(3p + m) - 3p - 1 = m - 1.$$

Let C be any set variable; as c contains three elements, and there is only one inefficient edge in G, there is an element $x \in c$ such that X is not adjacent to any other member variable X' with x' in c. So by Assertion 25, all $3p$ adjacencies $X - C$ with $x \in c$ are oriented as $X \to C$. So no edge of the form $C \to A$ is included in the $3p$ links $X \to C$, and since all edges of the form $C \to A$ are efficient by definition, it follows that there are at most $m - 1$ such edges. In other words, there are at most $m - 1$ set variables that have outdegree greater than 0. So there are at most $3(m - 1) = 3m - 3$ elements x of the universe such that X is adjacent to some set variable C with outdegree greater than 0. The contrapositive of this conclusion is that at least three elements x, y, z are adjacent only to set variables with outdegree 0. At least one of the corresponding nodes X, Y, Z is distinct from A and B; without loss of generality, choose X to be the distinct one. As all edges linking X are efficient, edges linking X are of the form $X \to C$ with $x \in c$ and $X \leftarrow C$ where $x \notin c$. But since X is adjacent only to set variables with outdegree 0, the latter type of edge does not exist, and so all edges linking X are of the form $X \to C$ where $x \in c$ and the set

variable C has outdegree 0. Any path that d-connects X and R has to start with an edge $X \to C \to \cdots - R$; since there is no such path, the graph G fails to entail the dependencies $\mathcal{D}_3$. This contradiction shows that G has more than $3p + m$ edges.

Inductive Step (still with $a = 0$): Assume the hypothesis for i and consider $i + 1$.

We show how to transform G into an I-map G' of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ with one less inefficient edge and no more edges overall. The inductive hypothesis then implies that G' and hence G has more than k edges.

1. Reorient all edges to point into the root R. By Lemma 26, the result G_1 is an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ with the same number of edges; the number of inefficient edges is the same.
2. If G_1 contains an edge $X \to R$, where X is a member variable, then:
   (a) Choose a set variable C with $x \in c$, and add an edge $C \to R$ (such an edge may already exist in G_1).
   (b) Reorient any adjacency $B \leftarrow C$ to point into C, where $B \neq R$. By Lemma 27, the result G_2 is an I-map of $\mathcal{D}_3$. And by Assertions 24 and 25 G' is also an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2$. Since there are no efficient set-nonmember links ($a = 0$), it is not possible to have $B = X$ with $x \notin c$, so this step adds no inefficient edges.
   (c) Delete the edge $X \to R$. Since in G_2 there are two directed paths connecting X to R, namely $X \to R$ and $X \to C \to R$, Lemma 28 guarantees that the resulting graph is an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. As the construction adds an edge in step 2a and deletes one in this step, the overall number of edges is the same as in G. Since no inefficient edge was added and this step deletes an inefficient edge, the number of inefficient edges has decreased by 1 as required.
3. Else if G_1 contains an adjacency $X - Y$ where X and Y are member variables, suppose without loss of generality that the link is oriented as $X \to Y$.
   (a) If Y is an ancestor of the root R, then:
      i. Choose a set variable C with $x \in c$, and add an edge $C \to R$ (such an edge may already exist in G_1).
      ii. Reorient any adjacency $B \leftarrow C$ to point into C, where $B \neq R$. As in Steps 2a and 2b above, the result G_2 is an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ with no added inefficient edges and at most one more edge overall.
      iii. Delete the edge $X \to Y$. Since in G_2 there are two directed paths connecting X to R, namely $X \to Y \to \cdots \to R$ and $X \to C \to R$, Lemma 28 guarantees that the resulting graph is an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. As the construction adds an edge in step 2a and deletes one in this step, the overall number of edges is the same as in G. Since no inefficient edge was added and this step deletes an inefficient edge, the number of inefficient edges has decreased by 1 as required.

(b) Else if Y is not an ancestor of the root R, add an edge $Y \to R$ and delete the edge $X \to Y$. Lemma 29 implies that the result is an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. Then proceed as in Step 2 to eliminate the edge $Y \to R$. In the resulting graph, the number of edges is the same as in the original graph G and there is one less inefficient edge, as required.

4. Else if G_1 contains an adjacency $X \to C$ where $x \notin c$, proceed as in Step 3 with C in place of Y.

This construction covers all cases of inefficient edges and completes the induction on i, the number of inefficient edges. Thus if the number of efficient set-nonmember links a is 0, then G contains more than $k = 3p + m$ edges.

Inductive Step: Assume the hypothesis for a and consider $a+1$. Let $X \leftarrow C$ be an efficient set-nonmember link in G such that $x \notin c$. We show how to transform G into an I-map G' of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$, where G' has one less efficient set-nonmember link edge, at least as many inefficient edges as G and no more edges than G overall. The inductive hypothesis for a then implies that G' and hence G has more than k edges.

1. If the variable X is is not an ancestor of R, then add an edge $X \to R$ and delete the edge $X \leftarrow C$. By Lemma 29, the resulting graph G' is an I-map of $\mathcal{D}_3$. And by Assertions 24 and 25 G' is also an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2$. Since one inefficient edge was added and one efficient set-nonmember edge was deleted, the graph G' has the same number of edges overall and more inefficient edges than G.
2. Else add an edge $C \to R$. Then there are two directed paths from C to R, namely $C \to R$ and $C \to X \to \cdots \to R$, so by Lemma 28, we may delete the edge $C \to X$ with the result being an I-map of $\mathcal{D}_3$. And by Assertions 24 and 25 G' is also an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2$. Since one efficient edge was added and one efficient set-nonmember edge was deleted, the graph G' has the same number of edges overall and as many inefficient edges as G.

This construction completes the inductive step; by inductive hypothesis, G' and thus G has more than k edges. Thus any I-map G of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ that contains an inefficient edge has more than k adjacencies. ■ The next Assertion shows that the number $k = 3p + m$ is a lower bound on the number of edges in an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. Combined with the previous assertion, we obtain strong constraints on the structure of a minimum-edge I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$.

Assertion 31. *Let DAG G be an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ with no more than $k = 3p + m$ adjacencies.*

1. *Every member variable X is an ancestor of the root variable R.*
2. *The DAG G contains exactly k edges, and the collection $\{c : C$ has outdegree 1$\}$ is a partition of the universe.*
3. *For every ancestor A of R, there is exactly one d-connecting path to R.*

Proof. Part 1: Let X be any member variable. Suppose for contradiction that X is not an ancestor of R. Then any d-connecting path from X to R must be of the form

$$X \leftarrow C - \cdots - R$$

where $x \notin c$, since by Assertion 30 G contains only efficient edges. Now by Lemma 29 we may add an edge $X \rightarrow R$ and delete the edge $X \leftarrow C$ with the result G' still an I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. But then G' contains the inefficient edge $X \rightarrow R$, so by Assertion 30, it has more than k edges, so G has more than k edges, which is a contradiction.

Part 2: Since G contains only efficient edges, Part 1 implies that for every member variable X, there is a set variable C with an edge $X \rightarrow C$ in G such that C is an ancestor of R, and C has outdegree greater than 0. So the collection $\{c : C \text{ is an ancestor of } R\}$ covers the universe U, and the size of this collection is at least m. On the other hand, for every member c of this collection, the corresponding set variable C has an edge originating in it, and overall there can be at most m such edges: for the graph contains a total of no more than $m+3p$ adjacencies and $3p$ of the adjacencies are of the form $X \rightarrow C'$ for $x \in c'$. So the size of the collection $\{c : C \text{ is an ancestor of } R\}$ is exactly m, which establishes that G contains exactly $m + 3p = k$ edges. Also, the collection is a partition of the universe. Thus (a) every set variable C has outdegree at most 1, and indegree exactly 3, (b) C has outdegree 1 if and only if C is an ancestor of R, and (c) every member variable X points to exactly one set variable C with outdegree 1.

Part 3: First we argue that if p is a path d-connecting an ancestor A to the root variable R, then p is directed. The proof is by induction on the length $l > 0$ of the path p.

Base Case, $l = 1$. Since G contains only efficient edges, the path p has the form $C - R$. If the adjacency $C - R$ is oriented as $C \leftarrow R$, then C has indegree 4, which contradicts (a) above.

Inductive Step: Assume the hypothesis for l and consider $l+1$. Suppose for contradiction that p d-connects A and R and is of the form

$$A \leftarrow B - \cdots - R.$$

Since A is an ancestor of R, so is B. The path segment $p' = B - \cdots - R$ d-connects B and R and has length l, so by inductive hypothesis p' is directed of the form

$$B \rightarrow V \rightarrow - \cdots \rightarrow R.$$

Thus B points to A and to V and has outdegree 2. By (a) above, this implies that B is not a set variable, and the base case implies that $B \neq R$, so B must be a member variable. But since G contains only efficient edges, it follows that A and V are set variables of outdegree 1, so B contradicts (c) above.

This completes the inductive step and establishes that all paths that d-connect an ancestor A to R begin with an edge $A \rightarrow B$. To complete the proof of the assertion, it suffices to show that for every ancestor A in graph G, there

is at most one directed path to the root variable R. We show this by induction on the length l of such a path.

Base Case, $l = 1$. Then the path is of the form $A \to R$, so A must be a set variable since G contains efficient edges only. By (a) above, every set variable has outdegree at most 1, so the only edge originating at A points directly into R.

Inductive Step: Assume the hypothesis for l and consider $l + 1$. Consider a directed path

$$A \to B \to \cdots \to R$$

of length $l + 1$. By inductive hypothesis, the final path segment $B \to \cdots \to R$ is the only directed path from B to R. If A is a set variable, then as in the base case, the variable B is its only possible successor, and so p is the only directed path from B to R. If A is a member variable, then since G contains only efficient edges, it follows that B is a set variable with outdegree 1. But by (c) above, the member variable A points to only one set variable, so the path $A \to B \to \cdots \to R$ is the only directed path between A and R.

So there is at most one directed path between an ancestor A and the root variable R. And we previously established that all d-connecting paths between A and R are directed. Hence there is a unique d-connecting path between A and R. ■

Assertion 31 implies that any I-map of $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$ is very close to the basic graph. The only other possibility remaining is an edge of the form $C \to X$ where $x \notin c$. If X is an ancestor of R via a path $X \to C' \to R$, the edge $C \to X$ can be used to d-connect members of c to R; for example if $y \in C$, there is a d-connecting path

$$Y \to C \to X \to C' \to R.$$

However, this path will be blocked by conditioning on the variable C', which is a set variable whose corresponding set does not contain y. So although this path satisfies Clause 3 of the oracle definition, it does not satisfy Clause 4, which is the last clause our proof takes into account.

Assertion 32. *Let DAG G be an I-map of $\mathcal{D}_O$ $(= \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3 \cup \mathcal{D}_4)$ with no more than than $k = 3p + m$ edges. Then G is a subgraph of the basic graph with exactly k adjacencies. That is, G that contains only the following types of edges: $X \to C$ where $x \in c$ and $C \to R$.*

Proof. Assertion 31 establishes that G has exactly k adjacencies. We argue that G contains m edges of the form $C \to R$. For this it suffices to show that the unique path that d-connects a member variable X to the root variable R has length 2. (The unique path exists by Assertion 31(1) and 31(3).) Suppose for contradiction that the path is of length greater than 2. Then since by Assertion 30 the graph G contains only efficient edges, the path p is of the form

$$p = X \to C \to A \to \cdots \to R.$$

Now A is not a set variable C' whose set contains X, for otherwise X is adjacent to two set variables with outdegree 1 (i.e., C and C'), which the proof of the

previous assertion showed is impossible. Therefore Clause 4 of the dependency oracle O implies that X is d-connected to R given A. Since p is the only path d-connecting X and R conditional on the empty set, and conditioning on A blocks this path, it follows that there must be a path p' that d-connects X and R given A on which A is a collider. So p' is of the form

$$X - \cdots \rightarrow A \leftarrow B - \cdots R.$$

Now B is an ancestor of A, and hence of R, that has two d-connecting paths to R: the path segment

$$B - \cdots R$$

and the path

$$B \rightarrow A \rightarrow \cdots \rightarrow R$$

following p. But this contradicts Assertion 31(3) above. So there is no path p' that d-connects X and R given A on which A is a collider, and G does not satisfy Clause 4 of the dependency oracle O, contrary to supposition. Hence the unique d-connecting path from a member variable X to the root R is of the form $X \rightarrow C \rightarrow R$, and so G contains exactly m edges of the type $C \rightarrow R$. Since G contains also $3p$ edges of the form $X \rightarrow C$ with $x \in c$ and a total of $m + 3p$ edges overall, G contains only edges from the basic graph. ■

Assertion 33. *There is a unique edge-minimal I-cover of $\mathcal{D}_O$ with at most $k = 3p + m$ edges $\iff$ there is a unique exact set cover in the corresponding set cover instance with p sets and m elements.*

Proof. By Assertion 32, no I-map of $\mathcal{D}_O$ has less than $3p + m$ edges. For every I-map G of $\mathcal{D}_O$ with exactly $3p + m$ edges, Assertion 31(2) says that the collection $\{c : C \text{ has outdegree } 1\}$ is a partition of the universe. Thus an I-map of $\mathcal{D}_O$ with at most $3p + m$ edges determines a unique exact set cover. Conversely, if there is an exact cover $\{c_1, \ldots, c_m\}$ for U, then there is an I-map G of $\mathcal{D}_O$ with exactly $3p + m$ edges, where $3p$ edges are of the form $X \rightarrow C_i$, with $x \in c_i$, and m edges are of the form $C_i \rightarrow R$ for $i = 1, \ldots, m$. This is the precisely the graph that maps onto the exact cover $\{c_1, \ldots, c_m\}$. Thus for every instance of UEC3SET, there is a 1-1 onto mapping between I-maps G of $\mathcal{D}_O$ with exactly $3p + m$ edges and exact set covers. So a given instance has a unique exact set cover if and only if the dependency oracle $\mathcal{D}_O$ has a unique I-map with exactly $3p + m$ edges. ■

We briefly indicate how our reduction differs from Bouckaert's NP-hardness proof for the problem of finding an I-map (not unique) with at most k edges. Bouckaert reduces the problem of finding an independent set of size at least k in a graph G to the I-map problem. In Bouckaert's reduction, for a given graph G with an independent set of size at least k, there is a corresponding I-map instance with oracle O with at most $\binom{n}{2} - \binom{k}{2}$ edges. The key step in the argument is to observe that any ordering of the nodes in an I-map entailing the dependencies in O can have at most $\binom{n}{2} - \binom{k}{2}$ edges. But his construction does not constrain the ordering of the nodes, so in general there will be more than one I-map solution for the corresponding independent set problem.

8. Conclusion

This paper applied learning-theoretic analysis to a practically important learning problem: identifying a correct Bayes net structure. We presented a model of this task in which learning is based on conditional dependencies between variables of interest. This model fits Gold's definition of a language learning problem, so identification criteria from Gold's paradigm apply. We considered mind-change optimality and text efficiency. The mind change complexity of identifying a Bayes net over variable set $\mathbf{V}$ is $\binom{|\mathbf{V}|}{2}$, the maximum number of edges in a graph with node set $\mathbf{V}$. There is a unique mind-change optimal learner $\Psi^{\mathbf{V}}_{\text{fast}}$ whose convergence time dominates that of all other mind-change optimal learners. This learner outputs a BN pattern G if G is the unique graph satisfying the observed dependencies with a minimum number of edges; otherwise $\Psi^{\mathbf{V}}_{\text{fast}}$ outputs ? for "no guess". In many language learning problems, it is plausible to view the mind change complexity of a language as a form of simplicity [17, Sec.4]. Our results establish that the mind-change based notion of simplicity for a Bayes net graph G is the inclusion depth of G, which is measured by the number of edges absent in G. Using the number of edges as a simplicity criterion to guide learning appears to be a new idea in constraint-based Bayes net learning research.

The technically most complex result of the paper shows that an exact implementation of the unique mind-change optimal learner $\Psi^{\mathbf{V}}_{\text{fast}}$ is NP-hard because determining whether there is a uniquely simplest (edge-minimal) Bayes net for a given set of dependencies is NP-hard. To our knowledge, this is the first NP-hardness result for deciding the existence of a uniquely optimal Bayes net structure by any optimality criterion.

Acknowledgements

This research was supported by NSERC discovery grants to the first and third author and by the Ebco/Eppich Visiting Scholar fund. We are indebted to Josh Buresh-Oppenheim for discussions of complexity theory. Parts of this work were previously presented at the Conference on Learning Theory (COLT 2007) and to the Tetrad Group at Carnegie Mellon University; the paper benefitted from helpful suggestions from these audiences. Greiner gratefully acknowledges support from the Alberta Ingenuity Centre for Machine Learning.

References

[1] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980.

[2] R. R. Bouckaert. *Bayesian belief networks : from construction to inference.* PhD thesis, Universiteit Utrecht, 1995.

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

[4] D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2003.

[5] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[6] G. Cooper. An overview of the representation and discovery of causal relationships using Bayesian networks. In C. Glymour and G. Cooper, editors, *Computation, Causation, and Discovery*, pages 4–62. AAAI Press/The MIT Press, 1999.

[7] R. G. Cowell, S. L. Lauritzen, and D. J. Spiegelhater. *Probabilistic Networks and Expert Systems*. Springer, 2005.

[8] L. M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187, 2006.

[9] R. N. Giere. The significance test controversy. *The British Journal for the Philosophy of Science*, 23(2):170–181, 1972.

[10] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[11] S. Jain, D. Osherson, J. S. Royer, and A. Sharma. *Systems That Learn*. MIT Press, 2 edition, 1999.

[12] S. Jain and A. Sharma. Mind change complexity of learning logic programs. *TCS*, 284(1):143–160, 2002.

[13] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2002.

[14] K. Kelly. Justification as truth-finding efficiency: How Ockham's razor works. *Minds and Machines*, 14(4):485–505, 2004.

[15] K. Kelly. Why probability does not capture the logic of scientific justification. In C. Hitchcock, editor, *Contemporary Debates in the philosophy of Science*, pages 94–114. Wiley-Blackwell, London, 2004.

[16] W. Luo and O. Schulte. Mind change efficient learning. In *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005*, pages 398–412, 2005.

[17] W. Luo and O. Schulte. Mind change efficient learning. *Information and Computation*, 204:989–1011, 2006.

[18] Wei Luo. *Mind change optimal learning : Theory and applications.* PhD thesis, Simon Fraser University, 2007.

[19] E. Martin and D. N. Osherson. *Elements of Scientific Inquiry.* The MIT Press, Cambridge, Massachusetts, 1998.

[20] C. Meek. *Graphical Models: Selecting causal and statistical models.* PhD thesis, Carnegie Mellon University, 1997.

[21] R. E. Neapolitan. *Learning Bayesian Networks.* Pearson Education, 2004.

[22] D. N. Osherson, M. Stob, and S. Weinstein. *Systems that learn: an introduction to learning theory for cognitive and computer scientists.* MIT Press, 1986.

[23] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[24] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kauffmann, San Mateo, CA, 1988.

[25] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge university press, 2000.

[26] H. Putnam. Trial and error predicates and the solution to a problem of Mostowski. *The Journal of Symbolic Logic*, 30(1):49–57, 1965.

[27] Joseph Ramsey, Jiji Zhang, and Peter Spirtes. Adjacency-faithfulness and conservative causal inference. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2006)*, pages 401–408. AUAI Press, 2006.

[28] Jörg Rothe. *Complexity Theory and Cryptology.* Springer Verlag, 2005.

[29] R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson. *TETRAD 3:Tools for Causal Modeling – User's Manual.* CMU Philosophy, 1996.

[30] T. Shinohara. Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, 8(4):371–384, 1991.

[31] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search.* MIT Press, 2000.

[32] M. Studeny. *Probabilistic Conditional Independence Structures.* Springer, 2005.

[33] I. Tsamardinos, L. E. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

[34] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.

[35] T. S. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI 1990)*, pages 220–227, 1990.

[36] Y. Xiang, S. K. Wong, and N. Cercone. Critical remarks on single link search in learning belief networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI 1996)*, pages 564–57, 1996.