Efficient Markov Network Structure Discovery using Independence Tests*

Facundo Bromberg, Dimitris Margaritis, Vasant Honavar
Dept. of Computer Science
Iowa State University
Ames, IA 50011
{bromberg,dmarg,honavar}@cs.iastate.edu

Abstract

We present two algorithms for learning the structure of a Markov network from discrete data: GSMN and GSIMN. Both algorithms use statistical conditional independence tests on data to infer the structure by successively constraining the set of structures consistent with the results of these tests. GSMN is a natural adaptation of the Grow-Shrink algorithm of Margaritis and Thrun for learning the structure of Bayesian networks. GSIMN extends GSMN by additionally exploiting Pearl's well-known properties of conditional independence relations to infer novel independencies from known independencies, thus avoiding the need to perform these tests. Experiments on artificial and real data sets show GSIMN can yield savings of up to 70% with respect to GSMN, while generating a Markov network with comparable or in several cases considerably improved quality. In addition to GSMN, we also compare GSIMN to a forward-chaining implementation, called GSIMN-FCH, that produces all possible conditional independence results by repeatedly applying Pearl's theorems on the known conditional independence tests. The results of this comparison show that GSIMN is nearly optimal in terms of the number of tests it can infer, under a fixed ordering of the tests performed.

1 Introduction and Related Work

Graphical models (Bayesian and Markov networks) are an important subclass of statistical models that possess advantages that include clear semantics and a sound and widely accepted theoretical foundation (probability theory). Graphical models can be used to represent efficiently the joint probability distribution of a domain. They have been used in numerous application domains, ranging from discovering gene expression pathways in bioinformatics [6] to computer vision ([7, 2], and more recently [12]). One problem that naturally arises is the construction of such models from data [9, 3]. A solution to this problem, besides being theoretically interesting in itself, also holds the potential of advancing the state-of-the-art in application domains where such models are used.

In this paper we focus on the task of learning the structure of Markov networks (**MNs**) from data in discrete domains. MNs are graphical models that consist of two parts: an undirected graph (the model structure), and a set of parameters. An example Markov network is shown in Fig. 1. Learning such models from data therefore consists of two interdependent problems: learning the structure of the network, and, given the learned structure, learning the parameters. In this work we only focus on the learning of the structure of the MN of a domain from data. The structure of a Markov network encodes graphically the independencies existing among the variables in the domain. These independencies are by themselves a valuable source of information for several fields of study (e.g., social sciences) that rely more on qualitative than quantitative models.

There exist two broad classes of algorithms for learning the structure of graphical models: *score-based* [14, 8] and *independence-based* or *constraint-based* [20]. Score-based approaches conduct a search in the space of legal structures (of size super-exponential in the number of variables in the domain) in an attempt to discover a model structure of maximum score. Independence-based algorithms are based on the fact that a graphical model implies that a set of independencies exist in the distribution of the domain, and therefore in the data set (under assumptions, see below) provided as input to the algorithm; they work by conducting a set of conditional independence tests on data, successively restricting the number of possible structures consistent with the results of those tests to a singleton (if possible), and inferring that structure as the only possible one.

While score-based algorithms are more robust for smaller data sets, independence-based approaches have the advantage of requiring no search, and are amenable

---

to proofs of correctness (under assumptions). In this work we present two algorithms that belong to the latter class. For Bayesian networks, the independence-based approach has been mainly exemplified by the SGS [20], PC [20], and the Grow-Shrink (GS) [15] algorithms, as well as algorithms for restricted classes such as trees [4] and polytrees [17]. Markov networks have been used in the physics and computer vision communities [7, 2] where they have been historically called Markov random fields. Recently there has been interest in their use for spatial data mining, which has applications in geography, transportation, agriculture, climatology, ecology and others [19].

Considerable work in the area of structure learning of undirected graphical models has concentrated on the learning of decomposable (also called chordal) MNs [21], as they render parameters learning and inference more tractable. These approaches proceed by constraining the output network to be in the class of decomposable structures. In this work we concentrate on obtaining the most accurate model (i.e., the one that best encodes the independencies in the domain), without any further restrictions other than the results of independence tests conducted on the data.

An example of learning (non-decomposable) MNs is presented by Hofmann and Tresp in [11], which is a score-based approach for learning structure in continuous domains with non-linear relationships among the domain attributes. There are no cases in the literature of independence-based structure learning of Markov networks that the authors are aware of. The present paper introduces two such algorithms: **GSMN** (Grow-Shrink Markov Network) and **GSIMN** (Grow-Shrink Inference Markov Network).

The GSMN algorithm is an adaptation to MNs of the GS algorithm by Margaritis and Thrun in [15], originally for learning the structure of Bayesian networks. GSMN works by first learning the local neighborhood of each variable in the domain (also called the **Markov blanket** of the variable), and then using this information in subsequent steps to improve efficiency. Although interesting in itself, GSMN serves as a point of reference (given the lack of alternatives) of the performance in regard to time complexity and accuracy achieved by GSIMN, which is the main result of this work. The GSIMN algorithm extends GSMN by using Pearl's theorems on the properties of the conditional independence relation [16] to infer additional independencies from a set of independencies resulting from statistical tests and previous inferences.

The rest of the paper is organized as follows: Section 2 introduces notation, definitions and presents intuition behind the two algorithms. In Section 3
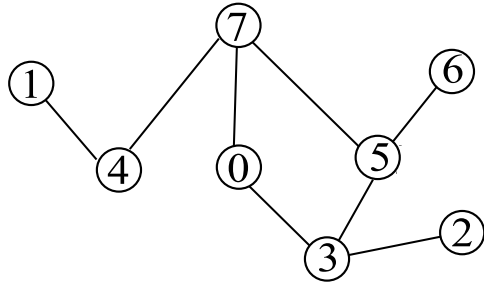

Figure 1: Example Markov network. The nodes represent variables in the domain $\mathbf{V} = \{0, 1, 2, 3, 4, 5, 6, 7\}$.

we present the GSMN algorithm and in Section 4 the Triangle theorem, which is used in the GSIMN algorithm of Section 5. We evaluate GSMN and GSIMN and present out results in Section 6, which is followed by a summary of our work and possible directions of future research in Section 7.

2   Notation and Preliminaries

We denote random variables with capitals (e.g., X, Y, Z) and sets of variables with bold capitals (e.g., $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$). In particular, we denote by $\mathbf{V} = \{1, \ldots, n\}$ the set of all n variables in the domain. Note that we name the variables by their indices in $\mathbf{V}$. For instance, we refer to the third variable in $\mathbf{V}$ simply by 3.

We assume that all variables in the domain are discrete. We denote the data set as D and its size in numbers of data points as N. We use the notation $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ to denote that $\mathbf{X}$ is independent of $\mathbf{Y}$ conditioned on $\mathbf{Z}$, for disjoint sets of variables $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$. $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ denotes conditional dependence. We will use $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ as shorthand for $(\{X\} \perp\!\!\!\perp \{Y\} \mid \mathbf{Z})$.

A Markov network is an undirected graphical model which represents the joint probability distribution over $\mathbf{V}$. Each node in the graph represents one of the random variables in the domain, while the absence of edges encodes conditional independencies among them. We assume the underlying probability distribution to be **graph-isomorph** [16], which means that it has a faithful undirected graph. A graph G is said to be **faithful** to some distribution if its graph connectivity represents exactly those dependencies and independencies existent in the distribution. This is equivalent to saying that for all disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$, $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$ if and only if the set of vertices $\mathbf{Z}$ graphically separates the set of vertices $\mathbf{X}$ from the set of vertices $\mathbf{Y}$ in G (e.g., in figure 1, set of variables $\{0, 5\}$ separates variable 7 from variable 3). It has been shown [16] that a necessary and sufficient condition for a distribution to be graph-isomorph is that its set of independencies satisfies the properties shown in Eqs. (1.1). In this paper

$$
\begin{array}{lrcl}
[\ \textbf{Symmetry}\ ] & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) & \Longleftrightarrow & (\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z}) \\
[\ \textbf{Decomposition}\ ] & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) & \Longleftrightarrow & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z}) \\
[\ \textbf{Intersection}\ ] \quad (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{W} \mid \mathbf{Z} \cup \mathbf{Y}) & \Longrightarrow & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \cup \mathbf{W} \mid \mathbf{Z}) & \quad (1.1) \\
[\ \textbf{Strong Union}\ ] & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) & \Longrightarrow & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \mathbf{W}) \\
[\ \textbf{Transitivity}\ ] & (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) & \Longrightarrow & (\mathbf{X} \perp\!\!\!\perp \gamma \mid \mathbf{Z}) \vee (\gamma \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})
\end{array}
$$

we assume faithfulness as well as no errors in the statistical tests conducted, which are standard assumptions for formally proving the correctness of independence-based structure learning algorithms [20].

2.1 Statistical Independence Testing. To determine conditional independence between two variables X and Y given a set $\mathbf{S}$ from data we use Pearson's conditional independence chi-square (χ^2) test (see [1] for details of its calculation). The χ^2 test returns a *p-value*, denoted as p, which is the probability of the error of assuming that the two variables are dependent when in fact they are not. We conclude independence if and only if $1 - p$ is greater than a certain confidence threshold α. We use the standard value of $\alpha = 0.95$ in all our experiments. Denoting by $\chi^2(X \perp\!\!\!\perp Y \mid \mathbf{S})$ the *p-value* for the χ^2 test on $(X, Y \mid \mathbf{S})$, we conclude that

$$(X \perp\!\!\!\perp Y \mid \mathbf{S}) \iff \chi^2(X \perp\!\!\!\perp Y \mid \mathbf{S}) < 1 - \alpha.$$

A practical consideration regarding the reliability of a conditional independence test is the size of the conditioning set as measured by the number of variables in the set, which in turn determines the number of values that the variables in the set may jointly take. Large conditioning sets produce sparse contingency tables (count histograms) and unreliable tests: The number of possible configurations of the variables grows exponentially with the size of the conditioning set e.g., there are 2^n cells in a test involving n binary variables. To fill such a table with one data point per cell we would need a data set of exponential size i.e., $N = 2^n$. Exacerbating this problem, more than that is typically necessary for a reliable test: As recommended by [5], if more than 20% of the contingency tables has less than 5 counts the tests is deemed unreliable. Both GSMN and GSIMN algorithms (presented later in the paper) attempt to minimize the conditioning set size by choosing an order of visiting the variables such that irrelevant variables are visited last. However, it is unavoidable that for small data sets we may not have enough data for a reliable determination of independence and therefore some tests will fail the aforementioned test of reliability. In these cases we must make an *a priori* decision: assume dependence or assume independence. While the assumption of dependence might appear to be the safer choice, in practice this would result in overly dense (sometimes completely connected) networks, which are hard to use in practice and of limited value to a researcher e.g., inference in Markov networks is exponential in the size of the largest clique. We therefore assume independence in these cases. As will be seen in the Experiments section, this does not affect the accuracy of the resulting networks, as measured by the correctness of the independencies present in the domain that they represent.

Finally, a subtle but important point is the fact that the time complexity of a statistical test on triplet $(X, Y \mid \mathbf{S})$ is linear in the size N of the data set and the size of the conditioning set i.e., $O(N(|\mathbf{S}| + 2))$, and not exponential in the number of variables involved (as a naïve implementation might assume). This is because we can construct all non-zero entries in the contingency table by examining each data point in the data set exactly once, in time proportional to the number of variables involved in the test i.e., proportional to $|\{X, Y\} \cup \mathbf{S}|$.

However, due to the linearity with the data set size N, the run time of the tests still has a major influence on the overall run time of an independence-based structure learning algorithm (as the algorithms presented in this paper). For this reason, the work presented in this paper concentrates on reducing the *weighted number of tests*, defined as the sum of all tests performed weighted by the size of their conditioning sets (plus two).

2.2 Independence Based Approach to Structure Learning. GSMN and GSIMN are independence-based algorithms for learning the structure of the network. This approach works by conducting a number of statistical conditional independence tests on the input data set D, reducing the set of structures consistent with the results of these tests to a singleton (if possible), and inferring that structure as the only possible one.

In a faithful domain, an edge exists between two variables $X \neq Y \in \mathbf{V}$ in the Markov network of that domain if an only if they are dependent conditioned on all remaining variables in the domain, i.e.,

$$(X, Y) \text{ is an edge iff } (X \not\perp\!\!\!\perp Y \mid \mathbf{V} - \{X, Y\})$$

Thus, to learn the structure, it suffices to perform only $n(n-1)/2$ tests, one for each triplet $(X, Y \mid \mathbf{V} - \{X, Y\})$ with $X, Y \in \mathbf{V}, X \neq Y$. Unfortunately this approach requires a data set of exponential size due to the size of the conditioning sets, as discussed in section 2.1. To address this problem, the approach taken by previous algorithms (e.g., the PC algorithm [20] for learning the structure of Bayesian networks) was to perform conditional tests of increasing conditioning set size. Since there are 2^{n-2} possible conditioning sets for each pair of variables X and Y, one question is how to order these possible tests—different algorithms present different orderings for the conditioning sets. In this paper we use an existing algorithm, namely GS, as a starting point. Since GS was designed for Bayesian networks, we first present an adaptation for Markov networks, and then present its extension GSIMN.

3   GSMN algorithm

In this section we discuss our first algorithm, GSMN (Grow-Shrink Markov Network), for learning the structure of a MN. Given as input a data set D and a set of variables $\mathbf{V}$, GSMN returns the sets of nodes (variables) $\mathbf{B}^X$ adjacent to each variable $X \in \mathbf{V}$, which completely determine the structure of the domain MN. The algorithm is shown in two parts, the main part (algorithm 1) and the independence test (algorithm 2).

The algorithm starts with an initialization stage. For reasons of clarity of exposition, we explain this stage near the end of this section. GSMN then executes its main loop that examines each variable in $\mathbf{V}$ (lines 14–39), according to the **visit order** π. Each iteration of the main loop (i.e., a visit) includes three phases: the **grow phase** (lines 18–26), the **shrink phase** (lines 27–33), and the **collaboration phase** (lines 35–38). The order that variables are examined during the grow phase of variable X is called the **grow order** λ_X of variable X.

The grow phase of X proceeds by attempting to add each variable Y to the current set of hypothesized neighbors of X, contained in $\mathbf{S}$. $\mathbf{S}$ is initially empty but at each iteration of the grow loop of X a variable Y is added to $\mathbf{S}$ if and only if Y is found dependent with X given the current value of $\mathbf{S}$. Due to the heuristic ordering that the variables are examined (determined by the priority queue λ_X), at the end of the grow phase, some of the variables in $\mathbf{S}$ might not be true neighbors of X in the underlying MN—these are called *false positives*. This justifies the shrink phase of the algorithm, which removes each false positive Y by testing for independence with X conditioned on $\mathbf{S} - \{Y\}$. If Y is found independent of X, it cannot be a true neighbor (i.e., there cannot be an edge $X - Y$), and GSMN removes it from $\mathbf{S}$. Assuming faithfulness

---

Algorithm 1 GSMN($\mathbf{V}, D$)

1: /* *Initialization.* */
2: **for all** $X, Y \in \mathbf{V}, X \neq Y$ **do**
3:    $K_{XY} \leftarrow \varnothing$
4:    $p_{XY} \leftarrow \chi^2(X \perp\!\!\!\perp Y \mid \varnothing)$
5:    $t \leftarrow (p_{XY} < 1 - \alpha)$
6:    add $(\varnothing, t)$ to K_{XY} and K_{YX}
7: **end for**
8: initialize π s.t. $i < i'$ iff $\mathrm{avg}_j\ p_{\pi_i j} < \mathrm{avg}_j\ p_{\pi_{i'} j}$
9: **for all** $X \in \mathbf{V}$ **do**
10:    $\mathbf{B}^X \leftarrow \varnothing$
11:    initialize λ_X s.t. $j < j'$ iff $p_{X\lambda_{Xj}} < p_{X\lambda_{Xj'}}$
12:    remove X from λ_X
13: **end for**
14: /* *Main loop.* */
15: **while** π not empty **do**
16:    $X \leftarrow dequeue(\pi)$
17:    $\mathbf{S} \leftarrow \varnothing$
18:    /* *Grow phase.* */
19:    **while** λ_X not empty **do**
20:       $Y \leftarrow dequeue(\lambda_X)$
21:       **if** $\neg I(X, Y, \mathbf{S})$ **then**
22:          $\mathbf{S} \leftarrow \mathbf{S} \cup \{Y\}$
23:       **else**
24:          remove X from λ_Y
25:       **end if**
26:    **end while**
27:    /* *Shrink phase.* */
28:    **for all** $Y \in \mathbf{S}$ **do**
29:       **if** $I(X, Y \mid \mathbf{S} - \{Y\})$ **then**
30:          $\mathbf{S} \leftarrow \mathbf{S} - \{Y\}$
31:          remove X from λ_Y
32:       **end if**
33:    **end for**
34:    $\mathbf{B}^X \leftarrow \mathbf{S}$
35:    /* *Collaboration phase.* */
36:    **for all** $Y \in \mathbf{B}^X$ **do**
37:       $\mathbf{B}^Y \leftarrow \mathbf{B}^Y \cup \{X\}$
38:    **end for**
39: **end while**

Algorithm 2 $I(X, Y \mid \mathbf{S})$

1: **if** $(\mathbf{S}, \texttt{true}) \in K_{XY}$ return $\texttt{true}$
2: **if** $(\mathbf{S}, \texttt{false}) \in K_{XY}$ or $Y \in \mathbf{B}^X$ return $\texttt{false}$
3: $t \leftarrow (\chi^2(X \perp\!\!\!\perp Y \mid \mathbf{S}) < 1 - \alpha)$ /* *Statistical test.* */
4: add $(\mathbf{S}, t)$ to K_{XY} and K_{YX}
5: **return** t

---

no errors in the statistical tests conducted, by the end of the shrink phase $\mathbf{B}^X$ contains exactly the neighbors of X (proof of correctness omitted due to space restriction).

a) $\pi = [5,3,7,0,1,6,2,4]$
$\lambda_5 = [3,4,1,6,2,7,0]$

b) $\pi = [3,7,0,1,6,2,4]$
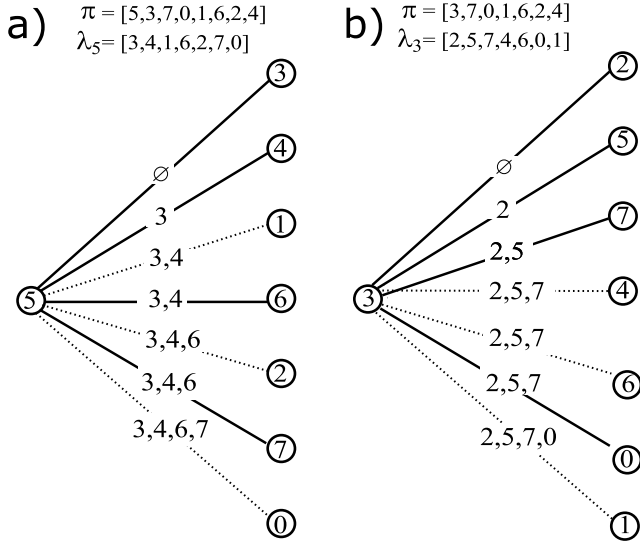$\lambda_3 = [2,5,7,4,6,0,1]$

Figure 2: Illustration of the operation of GSMN. The figure shows the growing phase of two consecutively visited variables 5 and 3 as dictated by visit ordering π.

After the neighbors of each X are produced in $\mathbf{B}^X$, GSMN executes a collaboration phase. During this phase, the algorithm adds X to $\mathbf{B}^Y$ of every node Y that is in $\mathbf{B}^X$. This is justified by the fact that in undirected graphs, X is adjacent to Y if and only if Y is adjacent to X.

As mentioned above, the order that variables are examined in the main loop and the grow phase is completely determined by the visit order π and grow orders λ_X. These are implemented as priority queues and are (initially) permutations of $\mathbf{V}$ (λ_X is a permutation of $\mathbf{V} - \{X\}$) such that the position of a variable in the queue denotes its priority e.g., $\pi = [2,0,1]$ means that variable 2 has higher priority (will be visited first), followed by 0 and 1. In this way, the first variable in the visit (grow) order is X (Y) where $X = \pi_0$ and $Y = \lambda_X = \lambda_{\pi_0}$.
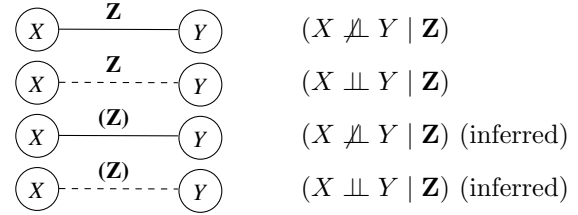
The visit (π) and grow (λ) orders are determined in the initialization phase. During this phase the algorithm computes the unconditional *p-value* $\chi^2(X \perp\!\!\!\perp Y \mid \varnothing)$ for each pair of variables $X \neq Y$, denoted p_{XY} in the algorithm. For visit order π, we give higher priority (i.e., visit earlier) to those variables with lower average *p-value* (line 8). This average is defined as follows:

$$\operatorname*{avg}_{Y} p_{XY} = \frac{1}{|\mathbf{V}| - 1} \sum_{Y \neq X \in \mathbf{V}} p_{XY}$$

whereas for growing order λ_X of variable X, we give higher priority to those variables Y whose *p-value* with variable X is smaller (line 11). This ordering is a heuristic justified by the intuition of a well-known "folk-

theorem" (as Koller and Sahami [13] put it) that states that probabilistic influence or association between attributes tends to attenuate over distance in a graphical model. This suggests that pair of variables X and Y with low unconditional *p-value* are less likely to be directly linked. It should also be noted that the computational price for p_{XY} is low due to the empty conditioning set.

We can represent the operation of GSMN graphically using an *independence graph*. An independence graph is an undirected graph where conditional independencies and dependencies between single variables are represented by one or more annotated edges between them. A solid (dotted) edge between variables X and Y annotated by $\mathbf{Z}$ represents the fact that X and Y are dependent (independent) given $\mathbf{Z}$. If the conditioning set $\mathbf{Z}$ is enclosed in parentheses then this edge represents an independence or dependence that was *inferred* from Eqs. (1.1) (as opposed to computed from statistical tests). Shown graphically:

$$(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$$
$$(X \perp\!\!\!\perp Y \mid \mathbf{Z})$$
$$(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ (inferred)}$$
$$(X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ (inferred)}$$

For instance, in figure 2, the dotted edge between 5 and 1 annotated with $3,4$ represents the fact that $(5 \perp\!\!\!\perp 1 \mid \{3,4\})$. The absence of an edge between two variables indicates the absence of information about the independence or dependence between these variables under any conditioning set.

Example. *Fig. 2 illustrates the operation of GSMN in the domain whose underlying Markov network is shown in Fig. 1. The figure shows the independence graph of the grow phases of the first two variables (5 and 3) according to visit order π. We do not discuss in this example the initialization phase of GSMN. Instead, we assume that the visit (π) and grow (λ) orders are given, and are as shown in the figure.*

Variable 5 is examined first by the algorithm (i.e., first in queue π). According to d-separation (equivalent to vertex separation in faithful domains) on the underlying network (Fig.1), variables 3, 4, 6, and 7 are found dependent with 5 during the growing phase i.e.,

$$\neg I(5,3 \mid \varnothing),$$
$$\neg I(5,4 \mid \{3\}),$$
$$\neg I(5,6 \mid \{3,4\}),$$
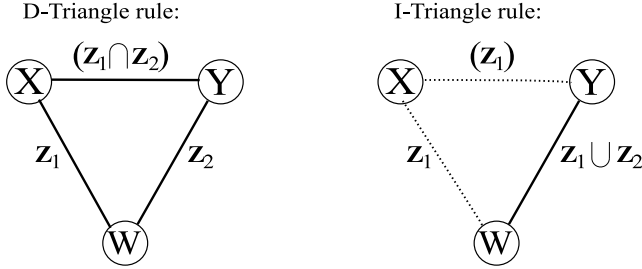$$\neg I(5,7 \mid \{3,4,6\})$$

Figure 3: Independence graph depicting the Triangle theorem. Edges in the graph are labeled by sets and represent conditional independencies or dependencies. A solid (dotted) edge between X and Y labeled by $\mathbf{Z}$ means that X and Y are dependent (independent) given $\mathbf{Z}$. A set label enclosed in parentheses means the edge was inferred by the theorem.

and are therefore successively added to **S**. *Variables* 1, 2, *and* 0 *are found independent i.e.,*

$$I(5, 1 \mid \{3, 4\}),$$
$$I(5, 2 \mid \{3, 4, 6\}),$$
$$I(5, 0 \mid \{3, 4, 6, 7\}),$$

and are not incorporated into **S**. *The final value of* **S** *at the end of the growing phase of variable* 5 *is* $\mathbf{S} = \{3, 4, 6, 7\}$.

4   The Triangle Theorem

In this section we present and prove a theorem that is used in the subsequent GSIMN algorithm. As will be seen, the main idea behind the GSIMN algorithm is to attempt to decrease the number of tests by exploiting the properties of the conditional independence relation, i.e., Eqs. (1.1). These properties can be seen as inference rules that can be used to derive new independencies from ones that we know to be true. A careful study of these axioms suggests that only two simple inference rules, stated in the *Triangle theorem* below, are sufficient for inferring most of the useful independence information that can be inferred by a systematic application of the inference rules (as will be confirmed by our experiments).

THEOREM 4.1. (TRIANGLE THEOREM) *Given Eq. (1.1), for every variable X, Y, W and sets $\mathbf{Z}_1$ and $\mathbf{Z}_2$ such that $\{X, Y, W\} \cap \mathbf{Z}_1 = \{X, Y, W\} \cap \mathbf{Z}_2 = \varnothing$,*

$$(X \not\perp\!\!\!\perp W \mid \mathbf{Z}_1) \wedge (W \not\perp\!\!\!\perp Y \mid \mathbf{Z}_2) \implies (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1 \cap \mathbf{Z}_2)$$
$$(X \perp\!\!\!\perp W \mid \mathbf{Z}_1) \wedge (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1 \cup \mathbf{Z}_2) \implies (X \perp\!\!\!\perp Y \mid \mathbf{Z}_1).$$

We call the first relation the "D-triangle rule" and the second the "I-triangle rule."

Proof. We are using the Strong Union and Transitivity of Eq. (1.1) as shown or in contrapositive form.
(Proof of D-triangle rule):

- From Strong Union and $(X \not\perp\!\!\!\perp W \mid \mathbf{Z}_1)$ we get $(X \not\perp\!\!\!\perp W \mid \mathbf{Z}_1 \cap \mathbf{Z}_2)$.
- From Strong Union and $(W \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1)$ we get $(W \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1 \cap \mathbf{Z}_2)$.
- From Transitivity, $(X \not\perp\!\!\!\perp W \mid \mathbf{Z}_1 \cap \mathbf{Z}_2)$, and $(W \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1 \cap \mathbf{Z}_2)$, we get $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1 \cap \mathbf{Z}_2)$.

(Proof of I-triangle rule):

- From Strong Union and $(X \not\perp\!\!\!\perp W \mid \mathbf{Z}_1 \cup \mathbf{Z}_2)$ we get $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1)$.
- From Transitivity, $(X \perp\!\!\!\perp W \mid \mathbf{Z}_1)$ and $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}_1)$ we get $(X \perp\!\!\!\perp Y \mid \mathbf{Z}_1)$. $\square$

We can represent the Triangle theorem graphically using the independence graph construct, defined in section 3. Fig. 3 depicts the two rules of the Triangle theorem using two independence graphs.

The Triangle theorem can be used to infer additional conditional independencies from tests conducted during the operation of GSMN. An example of this is shown in figure 4(a), which illustrates the application of the Triangle theorem to the example presented in figure 2(a). The independence information inferred from the Triangle theorem is shown by curved edges (note that the conditioning set of each such edge is enclosed in parentheses). For example, edge $(4, 7)$ can be inferred by the D-triangle rule from the adjacent edges $(5, 4)$ and $(5, 7)$, annotated by $\{3\}$ and $\{3, 4, 6\}$ respectively. The annotation for this inferred edge is $\{3\}$, which is the intersection of the annotations $\{3\}$ and $\{3, 4, 6\}$. An example application of the I-triangle rule is edge $(1, 7)$, which is inferred from edges $(5, 1)$ and $(5, 7)$ with annotations $\{3, 4\}$ and $\{3, 4, 6\}$ respectively. The annotation for this inferred edge is $\{3, 4\}$, which is the difference of the annotations $\{3, 4, 6\}$ and $\{3, 4\}$.

5   The GSIMN Algorithm

This section introduces the GSIMN algorithm. In the previous section we saw the possibility of using the triangle rules to infer the result of novel tests during the grow phase. The GSIMN algorithm uses the Triangle theorem in a similar fashion to extend GSMN and infer the value of a number of tests that GSMN executes, thus making it unnecessary to conduct these on the data set.

GSIMN works similarly to GSMN but differs in two important ways. First, it updates the visit and grow orders (the π and λ queues respectively) during its main loop. These updates are driven by the outcomes of the
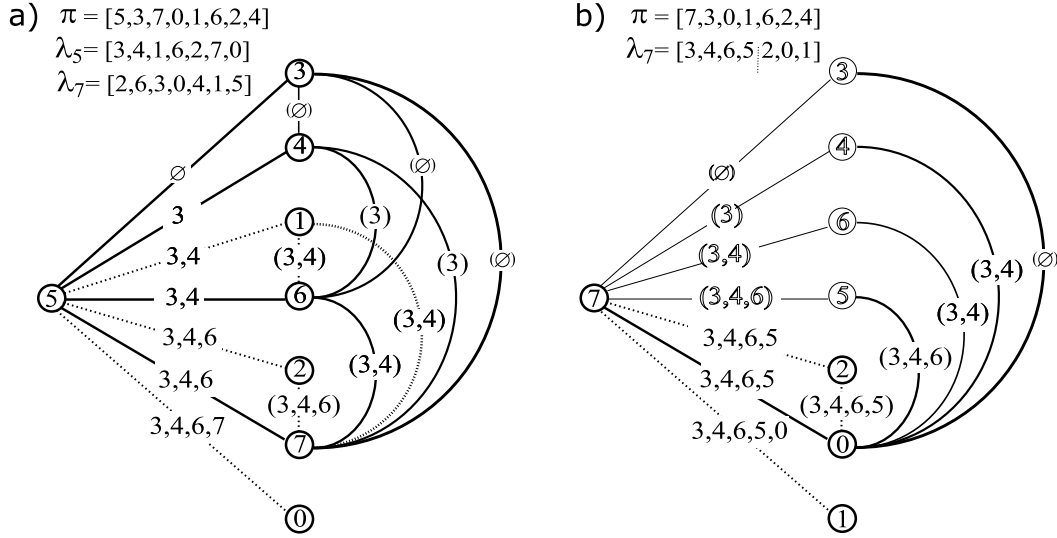
Figure 4: Illustration of the operation of GSIMN. The figure shows the grow phase of two consecutively visited variables 5 and 7. Contrary to GSMN (figure 2), the variable visited second is not 3 but 7, according to the change in the visit order π in line 32. The set of variables enclosed in parentheses correspond to tests inferred by the Triangle theorem using two adjacent edges as antecedents. For example, the results $(7 \not\perp 3 \mid \varnothing)$, $(7 \not\perp 4 \mid \{3\})$, $(7 \not\perp 6 \mid \{3,4\})$, and $(7 \not\perp 5 \mid \{3,4,6\})$ in (b) were not executed but inferred from the tests done in (a).

tests conducted up to that point and their purpose is to maximize the number of inferences possible through the use of the Triangle theorem. Second, it uses a different test procedure I' (algorithm 4) which first attempts to infer the independence value of its input variables by either Strong Union or the Triangle theorem. If it succeeds, it returns that value, otherwise it returns the outcome of a statistical test on the data set.

The operation of GSIMN is illustrated in Fig. 4, and the reader may want to refer to it during the following explanation.

We first describe the novel ordering used in GSIMN. The new visit order (lines 32–37) dictates that the next variable to be visited is the last to be added to $\mathbf{S}$ during the growing phase that still has not been visited (i.e., it is still in π). This is illustrated in Fig. 4 where the variable visited after 5 is 7, instead of 3 (as was done in GSMN and as dictated by the initial π). The change in order is conducted by the subroutine $changepos(q, X, j)$ which moves X from its current position in queue q to position j. For example, if $q = [5, 3, 1, 7]$, after applying $changepos(q, 7, 0)$ the queue changes to $q = [7, 5, 3, 1]$.

The change in growing order (lines 22–26) occurs inside the grow phase of the currently visited variable X. If, for some variable Y, $\neg I'(X, Y, \mathbf{S})$, then all the variables that were dependent with X before Y (i.e., all variables currently in $\mathbf{S}$) are promoted to the beginning of the grow order λ_Y. This is illustrated in Fig. 4 for variable 7, for which the grow order changes from

$\lambda_7 = [2, 6, 3, 0, 4, 1, 5]$ to $\lambda_7 = [3, 4, 6, 5 \mid 2, 0, 1]$ after the grow phase of variable 5 is complete.

The function I' (algorithm 4), which replaces I of GSMN, attempts to infer the independence value of the input triplet $(X, Y \mid \mathbf{S})$. It first attempts to apply Strong Union. This is done by checking whether the knowledge base (K) contains: (i) a test $(X \perp Y \mid \mathbf{A})$ with $\mathbf{A} \subseteq \mathbf{S}$ (line 2), or (ii) a test $(X \not\perp Y \mid \mathbf{A})$ with $\mathbf{A} \supseteq \mathbf{S}$ (line 3). If this fails, it attempts to infer the input triplet using the Triangle theorem. The search for antecedents focuses on those that could be used to infer only the input triplet $(X, Y \mid \mathbf{S})$. For that reason it searches for a variable $W \in \mathbf{V}$ such that (i) $(Y \perp W \mid \mathbf{A})$ and $(X \not\perp W \mid \mathbf{B})$ with $\mathbf{A} \subseteq \mathbf{S}$ and $\mathbf{B} \supseteq \mathbf{A}$; or (ii) $(Y \not\perp W \mid \mathbf{A})$ and $(X \not\perp W \mid \mathbf{B})$ with sets $\mathbf{A}$ and $\mathbf{B}$ both supersets of $\mathbf{S}$. According to I-triangle rule (i) and (ii) implies $(X \perp Y \mid \mathbf{S})$ and $(X \not\perp Y \mid \mathbf{S})$ respectively.

The curved edges in Figure 4 shows those tests that can be inferred during a growing phase. The change in visit and grow order described above was chosen so that these inferred tests are those required by the algorithm in some future stage. In particular, note in the example that the set of inferred dependencies between each variable found dependent with 5 before 7 are exactly those required during initial part of the grow phase of variable 7 (Figure 4(b)).

Similarly to GSMN, assuming faithfulness and no errors in the statistical tests conducted, by the end of

Algorithm 3 $GSIMN(\mathbf{V}, D)$

   /* Initialization. */
   for all $X, Y \in \mathbf{V}, X \neq Y$ **do**
     $K_{XY} \leftarrow \varnothing$
     $p_{XY} \leftarrow \chi^2(X \perp\!\!\!\perp Y \mid \varnothing)$
     $t \leftarrow (p_{XY} < 1 - \alpha)$
     add $(\varnothing, t)$ to K_{XY} and K_{YX}
   end for
   initialize π s.t. $i < i' \iff \underset{j}{\mathrm{avg}}\, p_{\pi_i j} < \underset{j}{\mathrm{avg}}\, p_{\pi_{i'} j}$
   for all $X \in \mathbf{V}$ **do**
     initialize λ_X s.t. $j < j' \iff p_{X\lambda_j} < p_{X\lambda_{j'}}$
     remove X from λ_X
   end for
   /* Main loop. */
   while π not empty **do**
     $X \leftarrow dequeue(\pi)$
     $\mathbf{S} \leftarrow \varnothing$
     /* Grow phase. */
     while λ_X not empty **do**
       $Y \leftarrow dequeue(\lambda_X)$
       if $\neg I'(X, Y, \mathbf{S})$ **then**
         add Y at the end of $\mathbf{S}$
         /* Change grow order. */
         $changepos(\lambda_Y, X, 0)$
         for $W = S_{|\mathbf{S}|-2}$ to S_0 **do**
           $changepos(\lambda_Y, W, 0)$
         end for
       else
         remove X from λ_Y
       end if
     end while
     /* Change visit order. */
     for $W = S_{|\mathbf{S}|-1}$ to S_0 **do**
       if $W \in \pi$ **then**
         $changepos(\pi, W, 0)$
         goto 38
       end if
     end for
     /* Shrink phase. */
     for $Y = S_{|\mathbf{S}|-1}$ to S_0 **do**
       if $I'(X, Y \mid \mathbf{S} - \{Y\})$ **then**
         remove Y from $\mathbf{S}$
         remove X from λ_Y
       end if
     end for
     $\mathbf{B}^X \leftarrow \mathbf{S}$
     /* Collaboration phase. */
     for $y = B_0^X$ to $B_{|\mathbf{B}^X|-1}^X$ **do**
       add $(\mathbf{V} - \{X, Y\}, \mathtt{false})$ to K_{XY} and K_{YX}
     end for
   end while

Algorithm 4 $I'(X, Y \mid \mathbf{S})$

1: /* Attempt to use Strong Union. */
2: **if** $\exists (\mathbf{A}, \mathtt{true}) \in K_{XY}$ s.t. $\mathbf{A} \subseteq \mathbf{S}$ **return** $\mathtt{true}$
3: **if** $\exists (\mathbf{A}, \mathtt{false}) \in K_{XY}$ s.t. $\mathbf{A} \supseteq \mathbf{S}$ **return** $\mathtt{false}$
4: /* Attempt to use the Triangle theorem. */
5: **for all** $W \in \mathbf{V}$ **do**
6:   **if** $\exists (\mathbf{A}, \mathtt{true}) \in K_{YW}$ such that $\mathbf{A} \subseteq \mathbf{S}$ $\wedge$
     $\exists (\mathbf{B}, \mathtt{false}) \in K_{XW}$ such that $\mathbf{B} \supseteq \mathbf{A}$
    then
7:     /* Infer independence by the I-triangle rule. */
8:     add $(\mathbf{A}, \mathtt{true})$ to K_{XY} and K_{YX}
9:     **return** $\mathtt{true}$
10:   **end if**
11:   **if** $\exists (\mathbf{A}, \mathtt{false}) \in K_{YW}$ such that $\mathbf{A} \supseteq \mathbf{S}$ $\wedge$
     $\exists (\mathbf{B}, \mathtt{false}) \in K_{XW}$ such that $\mathbf{B} \supseteq \mathbf{S}$
    then
12:     /* Infer dependence by the D-triangle rule. */
13:     add $(\mathbf{A} \cap \mathbf{B}, \mathtt{false})$ to K_{XY} and K_{YX}
14:     **return** $\mathtt{false}$
15:   **end if**
16: **end for**
17: /* Else do statistical test on data. */
18: $t \leftarrow (\chi^2(X \perp\!\!\!\perp Y \mid \mathbf{S}) < 1 - \alpha)$
19: add (S, t) to K_{XY} and K_{YX}
20: **return** t

Algorithm 5 $I''(X, Y, \mathbf{S})$

1: /* Query knowledge base. */
2: **if** $\exists (\mathbf{S}, t) \in K_{XY}$ **then**
3:   **return** t
4: **end if**
5: $t \leftarrow (\chi^2(X \perp\!\!\!\perp Y \mid \mathbf{S}) < 1 - \alpha)$ /* Statistical test. */
6: add $(\mathbf{S}, t)$ to K_{XY} and K_{YX}
7: run forward chaining on K, update K
8: **return** t

---

the shrink phase $\mathbf{B}^X$ contains exactly the neighbors of X. The correctness proof follows from the correctness of GSIMN, which was omitted for space restriction. As pointed out above, GSIMN only differs from GSMN by the visit and grow orderings and the test subroutine I'. This subroutine differs from its GSMN counterpart I in that it uses strong union and triangle theorem to infer novel tests. The proof of correctness of GSMN makes no assumptions on any particular visit (π) or grow ($\lambda's$) ordering, and thus the correctness of GSIMN follows from the correctness of GSMN, the triangle theorem and strong union.

5.1 Optimality of GSIMN. One reasonable question about the actual performance of GSIMN is to what extend it is *complete* i.e., from all those tests that

GSIMN needs during its operation, how does the number of them that it infers by use of the Triangle theorem (rather than executing a χ^2 statistical test on data) compares to the number of tests that *can* be inferred (e.g., using an automated theorem prover on Eqs. (1.1))? In this section we describe how we evaluate its completeness, and present the actual results of this evaluation in the Experimental Results section. To evaluate completeness, we compared the number of tests done by GSIMN with the number done by an alternative algorithm, which we call GSIMN-FCH (GSIMN with Forward Chaining). GSIMN-FCH differs from GSIMN only in function I'' (algorithm 5), replacing function I' of GSIMN, that exhaustively produces all independence statements that can be inferred through the properties of Eqs. (1.1). The process iteratively builds a knowledge base K containing the truth value of a set of conditional independence predicates in a fashion similar to previous algorithms. Whenever the outcome of a test is required, it queries K (line 2 of I'' in algorithm 5). If the value of the test is found in K, it is returned (line 3). If not, GSIMN-FCH performs the test on data and uses the result in a standard forward chaining automatic theorem prover subroutine (line 7) to produce all independence statements that can be inferred by the test result and K, adding these new facts to K. Comparisons of the number of tests required by GSIMN vs. GSIMN-FCH are presented and discussed in the results section below.

6   Experimental Results

We conducted experiments on both artificial and real-world data. We measured the following quantities:

- **Total weighted number of tests:** The number of tests executed can be used to assess the benefit of using inference instead of executing statistical tests on data for GSIMN. As discussed in section 2.1, the each conditional independence test should be weighted by the size of the conditioning set (plus two) to properly reflect the execution time required.

- **Accuracy of the resulting network (real-world data only):** Accuracy is used to assess the impact of inference on the quality of the outcome network. Actual statistical tests may be unreliable as discussed in section 2.1, and clearly, inferred independencies based on unreliable antecedents tests could be even less reliable. We explain how we measure accuracy below and explain why it makes sense for real-world data sets only.

- **Average neighborhood size of the resulting network (real-world data only):** Average neighborhood size can be used to assess the com-

Table 1: Comparison of weighted number of tests of GSIMN and GSIMN-FCH algorithms. These values are plotted in figure 5.

n	$\tau = 0.05$		$\tau = 0.3$		$\tau = 0.5$		$\tau = 0.7$		$\tau = 0.9$	
	GSIMN	FCH	GSIMN	FCH	GSIMN	FCH	GSIMN	FCH	GSIMN	FCH
2	2	2	2	2	2	2	2	2	2	2
3	6	6	8.7	8.7	11.7	11.7	13.8	13.8	14.7	14.7
4	12	12	20.5	19.1	32.2	31.5	35.9	35.6	39.2	39.2
5	20.9	20.5	41.7	40.4	78	74.8	81.6	80.8	80.9	80.9
6	30.6	30.6	80	76.3	139	137.1	147	146	146.1	145.5
7	45	45	158.3	151.1	233.5	222.5	243	238.6	240.3	239.3
8	62	61.3	229.5	222.9	357	345.6	378.1	370.8	365.7	365
9	77.7	77.7	378.2	370.9	559.8	551.8	559.3	547.1	527.2	525.5

plexity of the resulting networks. This is important because these output networks are rarely the final result aimed for by a data analyst. Commonly a Markov network is used for statistical inference. The size of the largest clique in the network has a great impact in the complexity of inference procedures, as well as on the learning of the parameters of the graphical model.

6.1   Artificial Data. This section presents two sets of experiments that compare the weighted number of tests of GSIMN vs. GSIMN-FCH, and GSIMN vs. GSMN. The experiments are conducted on randomly generated networks, referred to as *true networks* here.

Experimental results on artificial data has the advantage of allowing a systematic study of the impact of the use of the Triangle theorem by GSIMN on the weighted number of tests. By querying the true network for independence these tests can be performed much faster than actual statistical tests on data, allowing a check of the impact of inference for much larger networks—we were able to conduct experiments of domains containing up to 100 variables.

Each true network with n nodes was generated randomly as follows: the network was initialized with n variables and no edges. The set of edges was determined by a user-specified *connectivity probability* parameter $\tau \in [0, 1]$; an edge was added to the network if a number p, sampled uniformly from interval $[0, 1]$, was smaller than τ. The process was repeated for each pair of variables. Testing for conditional independence in these experiments was conducted using vertex separation (corresponding to d-separation in faithful domains) on the true network. Because graph-isomorphism holds, vertex-separation satisfies all the axioms (1.1), and the network resulting from GSMN, GSIMN or GSIMN-FCH matches exactly the true network. Therefore comparisons of the accuracy of the resulting network and the average neighborhood size are not useful.

In the first set of experiments, we compared the weighted number of tests between GSIMN-FCH and
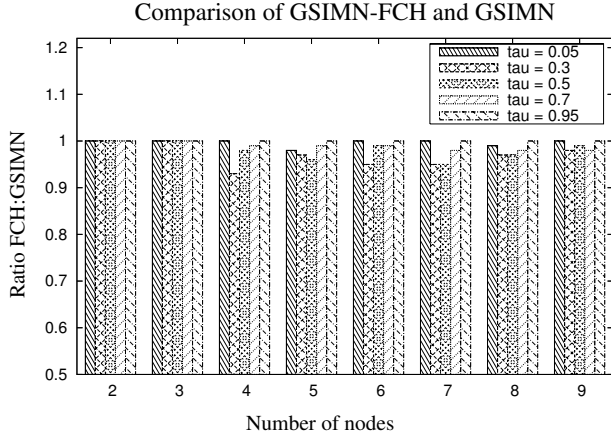
Figure 5: Ratio of number of tests of GSIMN-FCH over GSIMN for network sizes (number of nodes) $n = 2$ to $n = 9$ and connectivity probabilities $\tau = 0.1, 0.3, 0.5, 0.7,$ and 0.95.
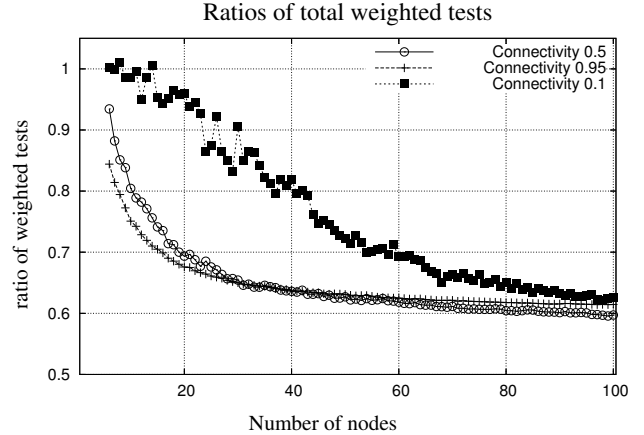

Figure 6: Ratio of the number of tests of GSIMN over GSMN for network sizes (number of nodes) $n = 1$ to $n = 100$ and connectivity probabilities $\tau = 0.1, 0.5,$ and 0.95.

GSIMN. The ratios of these numbers are shown in table 1 and plotted in figure 5. Ten networks were generated randomly for each pair n, τ. We used $\tau = 0.05, 0.3, 0.5$ and 0.9 and n up to 100. The table and the figure shows the mean value over these ten runs. Unfortunately GSIMN-FCH was not able to complete execution on domains containing more than 9 variables. However, we could still make an assessment of the reduction in tests required by GSIMN-FCH for $n \leq 9$. The figure shows that for every n and every τ the ratio is above 0.95 with the exception of a single pair ($n = 4$, $\tau = 0.3$) i.e., almost all inferable tests were produced by the use of the Triangle theorem in GSIMN. The small difference is greatly compensated by the very large increase in efficiency of the inference process (i.e., use of the Triangle theorem vs. automatic theorem proving on Eq. (1.1)).

In the second set of experiments, we compared the weighted number of tests required by GSMN and GSIMN for networks of up to 100 variables. The results are summarized in figure 6, showing the ratio between weighted number of tests of GSIMN vs. GSMN. Again, ten true networks were generated randomly for each pair n, τ, and the figure shows their mean value. The figure shows that, while for small numbers of variables GSIMN performs similar to GSIMN, for large number of nodes GSIMN performs considerably better: The reduction in number of tests is close to 40% for $n = 100$ regardless of the connectivity of the underlying domain. This shows the clear advantage of GSIMN vs. GSMN in terms of number of tests required, especially in large, difficult domains, regardless of the density of the underlying Markov network model.
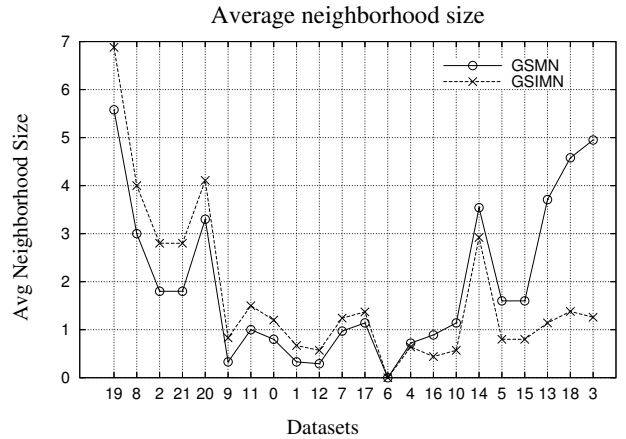
Figure 7: Average neighborhood size of resulting networks from GSMN and GSIMN on real data sets, sorted in increasing order of their difference.

6.2 Real-World Data. While artificial data set studies have a number of unique advantages (see previous section), they have the disadvantage of a random network topology and they fulfill the assumption of faithfulness. Real data on the other hand, may come from non-random topologies (e.g., a lattice in many cases of spatial data), but most importantly the underlying probability distribution may not be faithful, allowing a more realistic assessment of the performance of GSIMN.

We conducted experiments on a substantial number of data sets obtained from the UCI KDD archive [10]. Although some of these data sets are artificially generated (e.g., Alarm), they are not necessarily generated from a probability distribution that is faithful to any Markov network. Data sets containing continuous vari-
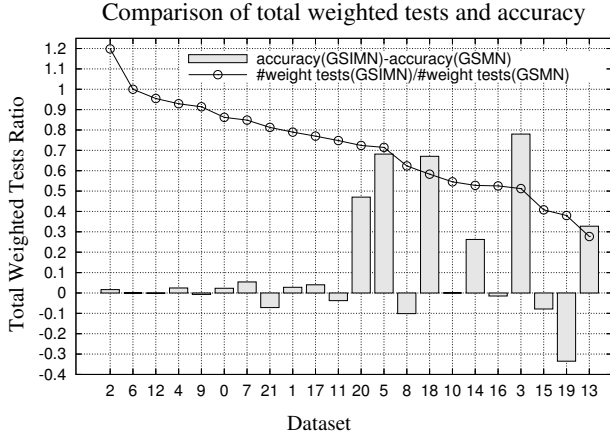
Figure 8: Ratio of the weighted number of tests of GSIMN versus GSMN and difference between the accuracy of GSIMN and GSMN on real data sets. (Positive bars indicate an advantage of GSIMN over GSMN.) The numbers in the x-axis are references to the data sets as shown in Table 2.

ables were discretized using Sturges's method, which is a simple method widely recommended in introductory statistics texts [18] that dictates that the optimal number of equally-spaced discretization bins for each continuous variable is $k = 1 + \log_2 N$, where N is the number of points in the data set. For each data set and each algorithm, we report the weighted number of conditional independence tests conducted to discover the network, the accuracy with respect to the data, and the average neighborhood size.

For real data the true network is unknown. In order to measure the quality of a network produced by GSMN and GSIMN, we propose a measure of accuracy that compares the result (`true` or `false`) of a number of *unseen* conditional independence tests on the network (using vertex separation) to their true value, obtained by performing these tests on the data set (using a χ^2 test). This procedure measures how the output network generalizes on unseen tests.

Since the number of possible tests is exponential, we sampled 1,000 triplets $(X, Y, \mathbf{Z})$ randomly in the following fashion: First, two variables X and Y were drawn randomly from $\mathbf{V}$. Second, a number $k \in \{0, \cdots, |\mathbf{V}| - 2\}$ was picked uniformly to serve as the size of the conditioning set $\mathbf{Z}$. Third, k variables were drawn randomly and without replacement from $\mathbf{V}$ to form set $\mathbf{Z}$. Finally, the sampled triplet was discarded if it corresponded to a test already added to the set of tests generated so far (i.e., if it is a duplicate).

Denoting by $\mathcal{T}$ this set of 1,000 triplets, by $t \in \mathcal{T}$ a triplet, by $I_{\text{data}}(t)$ the result of a test performed on the data, and by $I_{\text{network}}(t)$ the result of a test performed

on the output network produced by either GSMN or GSIMN, the accuracy is defined as:

$$\widehat{accuracy} = \frac{1}{|\mathcal{T}|} \left| \left\{ t \in \mathcal{T} \mid I_{\text{network}}(t) = I_{\text{data}}(t) \right\} \right|.$$

In practice, a triplet $t \in \mathcal{T}$ may result in an unreliable statistical test $I_{\text{data}}(t)$ when tested on data (section 2.1). Counting the comparison of such unreliable test with the outcome of the test performed on the output network $I_{\text{network}}(t)$ can only lead to unrealistic values of the accuracy. Therefore, triplets that yielded unreliable tests were replaced in $\mathcal{T}$ with new random triplet.

For each of the data sets, Table 2 shows the detailed results for accuracy, number of tests conducted and average neighborhood size for GSMN and GSIMN. It also serves as key for each data set index appearing in figures 7 and 8.

Figure 7 shows the average neighborhood size (average $|\mathbf{B}^X|$ over $X \in \mathbf{V}$) of GSMN and GSIMN for different data sets, in decreasing order of difference. The graph indicates that the average neighborhood sizes of the networks produced by GSMN and GSIMN are comparable.

Figure 8 shows the ratio of the weighted number of tests of GSIMN versus GSMN (i.e., a number smaller than 1 shows improvement of GSIMN vs. GSMN) together with the difference of the accuracies of GSIMN and GSMN (i.e., a positive histogram bar shows an improvement of GSIMN vs. GSMN) for different data sets. The numbers in the x-axis are indices to the data sets as shown in Table 2.

As shown in figure 8, GSIMN reduced the weighted number of tests on every data set, with maximum savings of 70%. Moreover, in the 17 out of 21 data sets GSIMN resulted in improved accuracy, with 6 of these showing a considerable improvement in addition to an approximate average savings of 50% in the weighted number of tests.

7   Conclusions and Future Research

In this paper we presented two algorithms, GSMN and GSIMN, for learning the structure of a Markov network of a domain from data using the independence-based approach. We evaluated their performance through the measurement of the weighted number of tests they require to learn the structure of the network, the quality of the networks learned from real-world data sets, and the average neighborhood size. GSIMN has shown a decrease of the number of tests in difficult (large) domains by a factor of 2, with an output network quality comparable to that of GSMN, with some cases showing great improvements. In addition, GSIMN was shown to be nearly optimal in the number of tests executed

Table 2: Total weighted number of tests, accuracy, and average neighborhood size (average $|\mathbf{B}^X|$ over $X \in \mathbf{V}$) for several real-world data sets. For each evaluation measure, the best performance between GSMN and GSIMN is indicated in bold. n denotes the number of variables in the domain and N is the number of instances in each data set.

	Data set			#(tests)		Accuracy		Avg $|B|$	
#	name	n	N	GSMN	GSIMN	GSMN	GSIMN	GSMN	GSIMN
0	hepatitis	20	80	456	**393**	0.885	**0.908**	**0.80**	1.20
1	hayes-roth	6	132	38	**30**	0.905	**0.933**	**0.33**	0.67
2	cmc	10	1473	**227**	272	0.714	**0.731**	**1.80**	2.80
3	bands	38	277	2844	**1457**	0.101	**0.881**	4.95	**1.26**
4	imports-85	25	193	646	**600**	0.945	**0.970**	0.72	**0.64**
5	balance-s	5	625	28	**20**	0.182	**0.864**	1.60	**0.80**
6	baloons	5	20	20	20	1.000	1.000	0.00	0.00
7	flag	29	194	1028	**872**	0.851	**0.905**	**0.97**	1.24
8	tic-tac-toe	10	958	263	**164**	**0.654**	0.552	**3.00**	4.00
9	bridges	12	70	151	**138**	**0.951**	0.944	**0.33**	0.83
10	car	7	1728	77	**42**	0.719	0.719	1.14	**0.57**
11	crx	16	653	341	**255**	**0.918**	0.881	**1.00**	1.50
12	monks-1	7	556	44	**42**	0.969	0.968	**0.29**	0.57
13	echocardio	14	60	689	**191**	0.379	**0.706**	3.71	**1.14**
14	flare2	13	1065	500	**264**	0.438	**0.700**	3.54	**2.92**
15	haberman	5	305	49	**20**	**0.704**	0.625	1.60	**0.80**
16	nursery	9	12960	137	**72**	**0.775**	0.760	0.89	**0.44**
17	dermat	35	358	1707	**1314**	0.853	**0.893**	**1.14**	1.37
18	optdigits	65	5620	12080	**7045**	0.158	**0.864**	4.58	**1.38**
19	connect-4	43	65534	8544	**3246**	**0.747**	0.412	**5.58**	6.88
20	alarm	37	20001	2761	**1999**	0.312	**0.782**	**3.30**	4.11
21	adult	10	32561	219	**178**	**0.695**	0.623	**1.80**	2.80

compared to GSIMN-FCH, which uses an exhaustive search to produce all independence information that can inferred from Pearl's axioms. Some directions of future research include an investigation into the way the topology of the underlying Markov network affect the number of tests required and quality of the resulting network, esp. for popular topologies such as grids. Another research topic in the the impact on execution time of other visit and grow orderings of the variables.

References

[1] Alan Agresti. *Categorical Data Analysis*. Wiley, 2nd edition, 2002.

[2] J. Besag, J. York, and A. Mollie. Bayesian image restoration with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43:1–59, 1991.

[3] Wray L. Buntine. Operations for learning with graphical models. *JAIR*, 2:159–225, 1994.

[4] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462 – 467, May 1968.

[5] W. G. Cochran. Some methods of strengthening the common χ^2 tests. *Biometrics*, 10:417–451, 1954.

[6] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

[7] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian relation of images. *PAMI '84*, 6:721–741, 1984.

[8] D. Heckerman. A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[9] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 1995.

[10] S. Hettich and S. D. Bay. The UCI KDD archive [http://kdd.ics.uci.edu]. *Irvine, CA: University of California, Department of Information and Computer Science.*, 1999.

[11] R. Hofmann and V. Tresp. Nonlinear Markov networks for continuous variables. In *NIPS '98*, volume 10, pages 521–529, 1998.

[12] Michael Isard. Pampas: Real-valued graphical models for computer vision. In *CVPR '03*, volume 1, pages 613–620, 2003.

[13] D. Koller and M. Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[14] W. Lam and F. Bacchus. Learning Bayesian belief networks: an approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[15] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In S. A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press, 2000.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[17] George Rebane and Judea Pearl. The recovery of causal poly-trees from statistical data. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 175–182, Amsterdam, 1989. North-Holland.

[18] D. W. Scott. *Multivariate Density Estimation*. Wiley series in probability and mathematical statistics. John Wiley & Sons, 1992.

[19] Shashi Shekhar, Pusheng Zhang, Yan Huang, and Ranga Raju Vatsavai. *Trends in Spatial Data Mining*, chapter 19, pages 357–379. AAAI Press / The MIT Press, 2004.

[20] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning Series. MIT Press, 2nd edition, January 2000.

[21] Nathan Srebro and David Karger. Learning Markov networks: Maximum bounded tree-width graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, 2001.