An inclusion optimal algorithm for chain graph structure learning

Jose M. Peña
ADIT, IDA, Linköping University
Sweden

Dag Sonntag
ADIT, IDA, Linköping University
Sweden

Jens D. Nielsen
CLC bio, a Quiagen company
Denmark

Abstract

This paper presents and proves an extension of Meek's conjecture to chain graphs under the Lauritzen-Wermuth-Frydenberg interpretation. The proof of the conjecture leads to the development of a structure learning algorithm that finds an inclusion optimal chain graph for any given probability distribution satisfying the composition property. Finally, the new algorithm is experimentally evaluated.

1 INTRODUCTION

This paper deals with chain graphs under the Lauritzen-Wermuth-Frydenberg interpretation. Although these chain graphs were introduced to model independencies fairly early (Lauritzen and Wermuth, 1989), there has been relatively little research on them compared to, for example, Bayesian networks. This has mainly to do with the additional complexity that follows from the fact that chain graphs may have both directed and undirected edges, compared to Bayesian networks that only have directed edges. Lately, however, chain graphs have got renewed interest due to their ability to represent more independence models than Bayesian networks (Drton, 2009; Ma et al., 2008; Peña, 2009, 2011; Studený, 1997; Studený, 2005; Studený and Bouckaert, 1998; Studený et al., 2009; Volf and Studený, 1999).

A key component that chain graphs still lack compared to Bayesian networks is an inclusion optimal structure learning algorithm. This has mainly to do with that Meek's conjecture has been proven for Bayesian networks (Chickering, 2002) but not for chain graphs. We will in this article prove it and, then, use it to develop an algorithm that finds an inclusion optimal chain graph for any probability distribution satisfying the composition property. We will also provide an implementation of the algorithm, and experimentally compare it with the earlier published LCD algorithm (Ma et al., 2008). This comparison may seem unfair because the LCD algorithm finds a chain graph a given probability distribution is faithful to and, thus, it imposes a stronger requirement on the input compared to our algorithm. However, this is the best we can do given that we are the first to weaken the faithfulness requirement. To avoid biasing the results in our favour, the experiments only involve faithful probability distributions.

The rest of the article organized as follows. In Section 2, we introduce the notation used in the paper. In Section 3, we prove Meek's conjecture for chain graphs. In section 4, we describe an inclusion optimal learning algorithm. We implement and evaluate the algorithm in Section 5. We close in Section 6 with some conclusions.

2 PRELIMINARIES

In this section, we review some concepts from probabilistic graphical models that are used later in this paper. See, for instance, Lauritzen (1996) and Studený (2005) for further information. All the graphs, independence models, and probability distributions in this paper are defined over a finite set V. All the graphs in this paper are hybrid graphs, i.e. they have (possibly) both directed and undirected edges. We assume throughout the paper that the union and the intersection of sets precede the set difference when evaluating an expression.

If a graph G has a directed (resp. undirected) edge between two nodes X_1 and X_2, then we write that $X_1 \rightarrow X_2$ (resp. $X_1 - X_2$) is in G. When there is a directed or undirected edge between two nodes of G, we say that the two nodes are adjacent in G. The parents of a set of nodes Y of G is the set $Pa_G(Y) = \{X_1 | X_1 \rightarrow X_2 \text{ is in } G \text{ and } X_2 \in Y\}$. The neighbors of

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

a set of nodes Y of G is the set $Ne_G(Y) = \{X_1 | X_1 - X_2$ is in G and $X_2 \in Y\}$. The boundary of a node X_2 of G is the set $Bd_G(X_2) = Pa_G(X_2) \cup Ne_G(X_2)$. A route between two nodes X_1 and X_n of G is a sequence of nodes $X_1, \ldots, X_n$ s.t. X_i and X_{i+1} are adjacent in G for all $1 \leq i < n$. The length of a route is the number of (not necessarily distinct) edges in the route. We treat all singletons as routes of length zero. A route in G is called undirected if $X_i - X_{i+1}$ is in G for all $1 \leq i < n$. A route in G is called descending from X_1 to X_n if $X_i - X_{i+1}$ or $X_i \to X_{i+1}$ is in G for all $1 \leq i < n$. If there is a descending route from X_1 to X_n in G, then X_n is called a descendant of X_1. Note that X_1 is a descendant of itself, since we allow routes of length zero. The descendants of a set of nodes Y of G is the union of the descendants of each node of Y in G and are denoted $De_G(Y)$. If X_n is a descendant of X_1 in G but X_1 is not a descendant of X_n in G, then X_n is called a strict descendant of X_1. The strict descendants of a set of nodes Y of G is the union of the strict descendants of each node of Y in G and are denoted $Sd_G(Y)$. Given a route ρ between X_1 and X_n in G and a route ρ' between X_n and X_m in G, $\rho \cup \rho'$ denotes the route between X_1 and X_m in G resulting from appending ρ' to ρ.

A chain is a partition of V into ordered subsets, which we call the blocks of the chain. We say that an element $X \in V$ is to the left of another element $Y \in V$ in a chain α if the block of α containing X precedes the block of α containing Y in α. Equivalently, we can say that Y is to the right of X in α. We say that a graph G and a chain α are consistent when (i) for every edge $X \to Y$ in G, X is to the left of Y in α, and (ii) for every edge $X - Y$ in G, X and Y are in the same block of α. A chain graph (CG) is a graph that is consistent with a chain. A set of nodes of a CG is connected if there exists an undirected route in the CG between every pair of nodes of the set. A component of a CG is a maximal (wrt set inclusion) connected set of its nodes. A block of a CG is a set of components of the CG s.t. there is no directed edge between their nodes in the CG. Note that a component of a CG is connected, whereas a block of a CG or a block of a chain that is consistent with a CG is not necessarily connected. Given a set K of components of G, a component $C \in K$ is called maximal in G if none of its nodes is a descendant of $K \setminus \{C\}$ in G. A component C of G is called terminal in G if its descendants in G are exactly C. Let a component C of G be partitioned into two non-empty connected subsets $C \setminus L$ and L. By splitting C into $C \setminus L$ and L in G, we mean replacing every edge $X - Y$ in G s.t. $X \in C \setminus L$ and $Y \in L$ with an edge $X \to Y$. Moreover, we say that the split is feasible if (i) $X - Y$ is in G for all $X, Y \in Ne_G(L) \cap (C \setminus L)$, and (ii) $X \to Y$ is in G for all $X \in Pa_G(L)$ and $Y \in Ne_G(L) \cap (C \setminus L)$. Let L and R denote two components of G s.t. $Pa_G(R) \cap L \neq \emptyset$. By merging L and R in G, we mean replacing every edge $X \to Y$ in G s.t. $X \in L$ and $Y \in R$ with an edge $X - Y$. Moreover, we say that the merging is feasible if (i) $X - Y$ is in G for all $X, Y \in Pa_G(R) \cap L$, and (ii) $X \to Y$ is in G for all $X \in Pa_G(R) \setminus L$ and $Y \in Pa_G(R) \cap L$.

A section of a route ρ in a CG is a maximal undirected subroute of ρ. A section $X_2 - \ldots - X_{n-1}$ of ρ is a collider section of ρ if $X_1 \to X_2 - \ldots - X_{n-1} \leftarrow X_n$ is a subroute of ρ. Moreover, the edges $X_1 \to X_2$ and $X_{n-1} \leftarrow X_n$ are called collider edges. Let X, Y and Z denote three disjoint subsets of V. A route ρ in a CG is said to be Z-active when (i) every collider section of ρ has a node in Z, and (ii) every non-collider section of ρ has no node in Z. When there is no route in a CG G between a node of X and a node of Y that is Z-active, we say that X is separated from Y given Z in G and denote it as $X \perp_G Y | Z$. We denote by $X \not\perp_G Y | Z$ that $X \perp_G Y | Z$ does not hold.

Let X, Y, Z and W denote four disjoint subsets of V. An independence model M is a set of statements of the form $X \perp_M Y | Z$, meaning that X is independent of Y given Z. Given two independence models M and N, we denote by $M \subseteq N$ that if $X \perp_M Y | Z$ then $X \perp_N Y | Z$. We say that M is a graphoid if it satisfies the following properties: Symmetry $X \perp_M Y | Z \Rightarrow Y \perp_M X | Z$, decomposition $X \perp_M Y \cup W | Z \Rightarrow X \perp_M Y | Z$, weak union $X \perp_M Y \cup W | Z \Rightarrow X \perp_M Y | Z \cup W$, contraction $X \perp_M Y | Z \cup W \wedge X \perp_M W | Z \Rightarrow X \perp_M Y \cup W | Z$, and intersection $X \perp_M Y | Z \cup W \wedge X \perp_M W | Z \cup Y \Rightarrow X \perp_M Y \cup W | Z$. An independence model M is also said to satisfy the composition property when $X \perp_M Y | Z \wedge X \perp_M W | Z \Rightarrow X \perp_M YW | Z$. The independence model induced by a CG G, denoted as $I(G)$, is the set of separation statements $X \perp_G Y | Z$ that holds in G. It is known that $I(G)$ is a graphoid (Studený and Bouckaert, 1998, Lemma 3.1). Let H denote the graph resulting from a feasible split or merging in a CG G. Then, H is a CG and $I(H) = I(G)$ (Studený et al., 2009, Lemma 5 and Corollary 9). Two CGs H and G are said to be in the same equivalence class if $I(H) = I(G)$.

A CG G is an independence (I) map of an independence model M if $I(G) \subseteq M$. Moreover, G is a minimal independence (MI) map of M if removing any edge from G makes it cease to be an I map of M. An independence model M is said to be faithful w.r.t. a CG G if $I(G) = M$. We also say that an independence model M is faithful if it is faithful to some CG G. A CG G is said to satisfy the local Markov property w.r.t. an independence model M iff $X \perp_M V \setminus X \setminus Sd_G(X) \setminus Bd_G(X) | Bd_G(X)$ for all $X \in V$.

Given any chain $C_1, \ldots, C_n$ that is consistent with G, we say that G satisfies the pairwise block-recursive Markov property w.r.t. M if $X \perp_M Y | \cup_{j=1}^{k^*} C_j \setminus \{X, Y\}$ for all non-adjacent nodes X and Y of G and where k^* is the smallest k s.t. $X, Y \in \cup_{j=1}^{k} C_j$. If M is a graphoid and G satisfies the local Markov property or the pairwise block-recursive Markov property w.r.t. M, then G is an I map of M (Lauritzen, 1996, Theorem 3.34). We say that a CG G_α is a MI map of an independence model M relative to a chain α if G_α is a MI map of M and G_α is consistent with α. A CG G is said to include an independence model M iff $I(G) \subseteq M$. G is also said to be inclusion optimal w.r.t. M iff $I(G) \subseteq M$ and there exists no other CG H such that $I(G) \subset I(H) \subseteq M$.

3 MEEK'S CONJECTURE FOR CHAIN GRAPHS

In this section we extend Meek's conjecture to chain graphs. This will later be used to prove that the CG structure learning algorithm provided in the next section is inclusion optimal for those probability distributions that satisfy the composition property.

Given two directed and acyclic graphs G and H s.t. $I(H) \subseteq I(G)$, Meek's conjecture states that we can transform G into H by a sequence of directed edge additions and covered edge reversals s.t. after each operation in the sequence G is a directed and acyclic graph and $I(H) \subseteq I(G)$ (Meek, 1997). Meek's conjecture was proven to be true in (Chickering, 2002, Theorem 4) by developing an algorithm that constructs a valid sequence of operations. In this section, we extend Meek's conjecture from directed and acyclic graphs to CGs, and prove that the extended conjecture is true. Specifically, given two CGs G and H s.t. $I(H) \subseteq I(G)$, we prove that G can be transformed into H by a sequence of directed and undirected edge additions and feasible splits and mergings s.t. after each operation in the sequence G is a CG and $I(H) \subseteq I(G)$. The proof is constructive in the sense that we give an algorithm that constructs a valid sequence of operations. See Appendix A for an example run of the algorithm.

We start by introducing two new operations on CGs. It is worth mentioning that all the algorithms in this paper use a "by reference" calling convention, meaning that the algorithms can modify the arguments passed to them. Let K denote a block of a CG G. Let $L \subseteq K$. By feasible block splitting (fbsplitting) K into $K \setminus L$ and L in G, we mean running the algorithm at the top of Figure 1. The algorithm repeatedly splits a component of G until L becomes a block of G. Before the splits, the algorithm adds to G the smallest set of edges so that the splits are feasible. Let L and R denote two

Fbsplit(K, L, G)

1 Let $L_1, \ldots, L_n$ denote the maximal connected subsets of L in G
2 For $i = 1$ to n do
3 Add an edge $X - Y$ to G for all $X, Y \in Ne_G(L_i) \cap (K \setminus L)$
4 Add an edge $X \rightarrow Y$ to G for all $X \in Pa_G(L_i)$ and $Y \in Ne_G(L_i) \cap (K \setminus L)$
5 For $i = 1$ to n do
6 Let K_j denote the component of G s.t. $L_i \subseteq K_j$
7 If $K_j \setminus L_i \neq \emptyset$ then
8 Split K_j into $K_j \setminus L_i$ and L_i in G

Fbmerge(L, R, G)

1 Let $R_1, \ldots, R_n$ denote the components of G that are in R
2 For $i = 1$ to n do
3 Add an edge $X - Y$ to G for all $X, Y \in Pa_G(R_i) \cap L$
4 Add an edge $X \rightarrow Y$ to G for all $X \in Pa_G(R_i) \setminus L$ and $Y \in Pa_G(R_i) \cap L$
5 For $i = 1$ to n do
6 Let L_j denote the component of G st $L_j \subseteq L \cup R$ and $Pa_G(R_i) \cap L_j \neq \emptyset$
7 If $L_j \neq \emptyset$ then
8 Merge L_j and R_i in G

Figure 1: Fbsplit And Fbmerge.

blocks of a CG G. By feasible block merging (fbmerging) L and R in G, we mean running the algorithm at the bottom of Figure 1. The algorithm repeatedly merges two components of G until $L \cup R$ becomes a block of G. Before the mergings, the algorithm adds to G the smallest set of edges so that the mergings are feasible. It is worth mentioning that the component L_j in line 6 is guaranteed to be unique by the edges added in lines 3 and 4.

Our proof of the extension of Meek's conjecture to CGs builds upon an algorithm for efficiently deriving the MI map G_α of the independence model induced by a given CG G relative to a given chain α. The pseudocode of the algorithm, called Method B3, can be seen in Figure 2. Method B3 works iteratively by fbsplitting and fbmerging some blocks of G until the resulting CG is consistent with α. It is not difficult to see that such a way of working results in a CG that is an I map of $I(G)$. However, in order to arrive at G_α, the blocks of G to modify in each iteration must be carefully chosen. For this purpose, Method B3 starts by calling Construct β to derive a chain β that is consistent with G and as close to α as possible (see lines 5-8). By β being as close to α as possible, we mean that the number of blocks Method B3 will later fbsplit and fbmerge is kept at a minimum, because Method B3 will use β

Construct $\beta(G, \alpha, \beta)$

1 Set $\beta = \emptyset$
2 Set $H = G$
3 Let C denote any terminal component of H whose leftmost node in α is rightmost in α
4 Add C as the leftmost block of β
5 Let R denote the right neighbor of C in β
6 If $R \neq \emptyset$, $Pa_G(R) \cap C = \emptyset$, and the nodes of C are to the right of the nodes of R in α then
7 Replace C, R with R, C in β
8 Go to line 5
9 Remove C and all its incoming edges from H
10 If $H \neq \emptyset$ then
11 Go to line 3

Method B3(G, α)

1 Construct $\beta(G, \alpha, \beta)$
2 Let C denote the rightmost block of α that has not been considered before
3 Let K be the leftmost block of β s.t. $K \cap C \neq \emptyset$
4 Set $L = K \cap C$
5 If $K \setminus L \neq \emptyset$ then
6 Fbsplit(K, L, G)
7 Replace K with $K \setminus L, L$ in β
8 Let R denote the right neighbor of L in β
9 If $R \neq \emptyset$ and some node of R is not to the right of the nodes of L in α
10 Fbmerge(L, R, G)
11 Replace L, R with $L \cup R$ in β
12 Go to line 3
13 If $\beta \neq \alpha$ then
14 Go to line 2

Figure 2: Method B3.

to choose the blocks to modify in each iteration. A line of Construct β that is worth explaining is line 3, because it is crucial for the correctness of Method B3 (see Case 3.2.4 in the proof of Lemma 2). This line determines the order in which the components of H (initially $H = G$) are added to β (initially $\beta = \emptyset$). In principle, a component of H may have nodes from several blocks of α. Line 3 labels each terminal component of H with its leftmost node in α and, then, chooses any terminal component whose label node is rightmost in α. This is the next component to add to β.

Once β has been constructed, Method B3 proceeds to transform G into G_α. In particular, Method B3 considers the blocks of α one by one in the reverse order in which they appear in α. For each block C of α, Method B3 iterates through the following steps. First, it finds the leftmost block K of β that has some nodes from C. These nodes, denoted as L, are then moved to the right in β by fbsplitting K to create a new block L of G and β. If the nodes of the right neighbor

Method G2H(G, H)

1 Let α denote a chain that is consistent with H
2 Method B3(G, α)
3 Add to G the edges that are in H but not in G

Figure 3: Method G2H.

R of L in β are to the right of the nodes of L in α, then Method B3 is done with C. Otherwise, Method B3 moves L further to the right in β by fbmerging L and R in G and β.

The proof of the following important intermediate result can be found in Appendix B.

Lemma 1. *Let M denote an independence model, and α a chain $C_1, \ldots, C_n$. If M is a graphoid, then there exits a unique CG G_α that is a MI map of M relative to α. Specifically, for each node X of each block C_k of α, $Bd_{G_\alpha}(X)$ is the smallest subset B of $\cup_{j=1}^{k} C_j \setminus \{X\}$ s.t. $X \perp_M \cup_{j=1}^{k} C_j \setminus \{X\} \setminus B | B$.*[1]

The proof of the following lemma, which guarantees the correctness of Method B3, can be found in Appendix B.

Lemma 2. *Let G_α denote the MI map of the independence model induced by a CG G relative to a chain α. Then, Method B3(G, α) returns G_α.*

We are now ready to prove that the extension of Meek's conjecture to CGs is true. The proof is constructive in the sense that we give an algorithm that constructs a valid sequence of operations. The pseudocode of our algorithm, called Method G2H, can be seen in Figure 3. The proof of the following theorem, which guarantees the correctness of Method G2H, can be found in Appendix B.

Theorem 1. *Given two CGs G and H s.t. $I(H) \subseteq I(G)$, Method G2H(G, H) transforms G into H by a sequence of directed and undirected edge additions and feasible splits and mergings s.t. after each operation in the sequence G is a CG and $I(H) \subseteq I(G)$.*

4 THE CKES ALGORITHM

With the extension of Meek's conjecture proven, we can now present an algorithm which relies upon it to find an inclusion optimal CG structure for a given probability distribution that satisfies the composition property. The algorithm is based on random walks in the equivalence classes and is a generalized version of the KES algorithm used for Bayesian network

[1] By convention, $X \perp_M \emptyset | \cup_{j=1}^{k} C_j \setminus \{X\}$.

CKES algorithm

1 G =Empty graph
2 Repeat until all the CGs in the equivalence class of G have been considered:
3 For every ordered pair of nodes X and Y:
4 If $X \to Y$ is in G but $X \perp_p Y | Bd_G(Y) \setminus X$ then remove $X \to Y$ from G and go to line 2
5 If $X - Y$ is in G but $X \perp_p Y | Bd_G(Y) \setminus X$ and $X \perp_p Y | Bd_G(X) \setminus Y$ then remove $X - Y$ from G and go to line 2
6 If $X \to Y$ is not in G but adding $X \to Y$ to G results in a CG and $X \not\perp_p Y | Bd_G(Y)$ then add $X \to Y$ to G and go to line 2
7 If $X - Y$ is not in G but adding $X - Y$ to G results in a CG and $X \not\perp_p Y | Bd_G(Y)$ or $X \not\perp_p Y | Bd_G(X)$ then add $X - Y$ to G and go to line 2
8 Move to another CG in the same equivalence class of G by performing a random number of random feasible merges or feasible splits on G and thereby updating G
9 Return G

Figure 4: The CKES Algorithm.

structure learning (Nielsen et al., 2003). Unlike KES, the CKES algorithm is not score-based but constraint-based. The algorithm is shown in Figure 4. The proof of the theorem below, which guarantees the correctness of the CKES algorithm, can be found in Appendix B.

Theorem 2. *For any probability distribution p for which the composition property holds, the CKES algorithm finds a CG that is inclusion optimal w.r.t. p.*

It is worth mentioning that it follows from the proof of the theorem above that the four operations in the CKES algorithm are necessary to guarantee its correctness.

Due to the infeasibility to enumerate all CGs in each visited equivalence class and the unavailability of the probability distribution p, some approximations had to be made when implementing the CKES algorithm. There are two major differences between the algorithm described in Figure 4 and its implementation in Figure 5. The first is that we check whether an independence holds in p by running a hypothesis test on a sample from it, which is what we have access to in practice. Such a test is represented in Figure 5 with the function $I(X, Y|Z)$, which returns the p-value of the test. The second major difference is that line 2 in Figure 4 has been replaced by two loops visible in lines 2 and 4 in Figure 5. The loop in line 4 considers m graphs in the current equivalence class to find the best edge addition or removal for that class. However, unless $m \to \infty$ it is unclear whether all graphs in the equivalence class

CKES implementation(α, k, l, m)

1 G =Empty graph
2 Repeat until $I(G)$ does not change for k iterations:
3 $p_r = 0$, $p_a = 1$
4 Repeat for m iterations:
5 For every ordered pair of nodes X and Y:
6 If $X \to Y$ is in G but $I(X, Y|Bd_G(Y) \setminus X) > p_r$, then set $p_r = I(X, Y|Bd_G(Y) \setminus X)$ and $G_r = G \setminus \{X \to Y\}$
7 If $X - Y$ is in G but $min(I(X, Y|Bd_G(Y) \setminus X), I(X, Y|Bd_G(X) \setminus Y)) > p_r$, then set $p_r = min(I(X, Y|Bd_G(Y) \setminus X), I(X, Y|Bd_G(X) \setminus Y))$ and $G_r = G \setminus \{X - Y\}$
8 If $X \to Y$ is not in G but adding $X \to Y$ to G results in a CG and $I(X, Y|Bd_G(Y)) < p_a$, then set $p_a = I(X, Y|Bd_G(Y))$ and $G_a = G \cup \{X \to Y\}$
9 If $X - Y$ is not in G but adding $X - Y$ to G results in a CG and $min(I(X, Y|Bd_G(Y)), I(X, Y|Bd_G(X)) < p_a$, then set $p_a = min(I(X, Y|Bd_G(Y)), I(X, Y|Bd_G(X))$ and $G_a = G \cup \{X - Y\}$
10 Move to another CG in the same equivalence class of G by performing l random feasible merges or feasible splits on G and thereby updating G
11 If $p_r > \alpha$, then set $G = G_r$ and go to line 2
12 If $p_a \leq \alpha$, then set $G = G_a$ and go to line 2
13 Return G

Figure 5: CKES Implementation.

have been considered. This means that not all edge additions and removals may have been considered for the equivalence class and that the edge addition or edge removal with lowest resp. highest p-value may not have been found. This can be acceptable if some edge addition or removal is found but not when determining if the equivalence class is an inclusion optimal CG (i.e. when terminating the algorithm). The loop in line 2 has therefore been added to make sure that the final equivalence class is searched extra thoroughly by repeating the inner loop k times. Finally, there is also the l variable in line 10, which controls how many splits or mergings that are performed when moving between graphs in the same equivalence class. All in all this means that large values on k, l and m increase the probability that the implementation terminates in an inclusion optimal CG, assuming that the independence hypothesis test does not induce any error. Theoretically, an inclusion optimal CG can only be guaranteed to be found if either $m \to \infty$ or $k \to \infty$. Empirical results do however suggest that these conditions can be relaxed considerably and that the parameter values

$k = 10$, $l = 4$, $m = 100$ are high enough to find a local optimum for our experiments in the next section.

5 EVALUATION

To our knowledge, only two other CG structure learning algorithms have been presented under the Lauritzen-Wermuth-Frydenberg interpretation. These are the LCG recovery algorithm (Studený, 1997) and the LCD algorithm (Ma et al., 2008). The SINful approach (Drton and Perlman, 2008) can also be used to learn a CG structure but only if its components and their order is known in advance. Both the LCG recovery algorithm and the LCD algorithm assume faithfulness. It can be shown that if this assumption does not hold, then these algorithms may find a CG that does not include the probability distribution at hand, even if the algorithms are given perfect information about the independencies that hold in the probability distribution (see Appendix C). In this section we will evaluate our algorithm only against the LCD algorithm, because it is the only that has been presented with an implementation and evaluation.

Recall that for the CKES algorithm to find an inclusion optimal CG of a probability distribution, it suffices that the latter satisfies the composition property. However, the LCD algorithm finds an inclusion optimal CG of a probability distribution only if the latter satisfies the faithfulness assumption. To avoid biasing the experimental results in our favour, we only considered faithful probability distributions in the experiments (see Appendix C for experiments under the composition property assumption). Specifically, we considered CGs with P=10, 20 nodes and N=2, 5 adjacencies per node on average. For each of these four scenarios, we generated 10 random CG structures with the algorithm described in (Ma et al., 2008). Each structure was then parameterized into one binary and one Gaussian probability distribution as proposed by Ma et al. (2008). Then, samples of size n=100, 30000 were obtained from these probability distributions. Using these samples and the independence hypothesis tests provided by Ma et al. with significance level $\alpha = 0.01$, the CKES and LCD algorithms were finally run. Note that the probability distributions sampled are likely to satisfy the faithfulness assumption, but there is no guarantee. Note also that the samples can have additional independencies that are not in the sampled probability distributions. As performance measures in the experiments, we computed the recall of the separations in the CGs sampled, and the precision of the separations in the CGs learnt. Since time did not permit it, we did not check all the separations but a large number of uniformly selected separations. Specifically, for any given CG, we uniformly selected 10000 triplets (X, Y, Z) with X, Y and Z disjoint sets of nodes among all possible such triplets. For each triplet (X, Y, Z), we tested whether X was separated from Y given Z in the given CG. If so, the separation was selected.

The first conclusion that we drew from our experiments is that multiple locally optimal CGs may be found from the same sample. This phenomenon has been previously noted for Bayesian networks (Nielsen et al., 2003). Figure 6 illustrates this phenomenon. Specifically, it plots the recall and precision of CGs obtained by running 100 times the LCD and 100 times the CKES algorithm on the same 30000 samples from a Gaussian probability distribution sampled from a CG with P=10 and N=2. In the case of the LCD implementation this randomization comes from variations in the order in which the variables are considered by the algorithm. The CKES implementation on the other hand contains both the explicit randomness due to lines 4 and 10 in Figure 5 and also the implicit randomness due to the order in which the variables are chosen in line 5 in Figure 5. The explicit randomness can to a large extent be minimized with large enough values on the parameters l and m in the algorithm. Increasing these parameters significantly was however seen to have little impact on the results. Instead, it is the implicit randomness that is most significant.

With the numerous locally optimal CGs found in mind, we then set out to find which of the two algorithms reached the best CG. This was done by running both algorithms 100 times on each sample set in the experiments and, then, selecting the best CGs in terms of recall and precision from those obtained from each sample set. Table 1 presents average recall and precision values over the best CGs obtained from the 10 sample sets created for each scenario in the experiments. In addition to these results, Table 2 presents the following values for each scenario:

- Best recall: The number of sample sets on which the CKES (resp. LCD) algorithm learnt a CG with better or equal recall than any of the CGs learnt by the LCD (resp. CKES) algorithm.
- Best precision: The number of sample sets on which the CKES (resp. LCD) algorithm learnt a CG with better or equal precision than any of the CGs learnt by the LCD (resp. CKES) algorithm.
- Best recall and precision: The number of sample sets on which the CKES (resp. LCD) algorithm learned a CG with better or equal recall and precision than any of the CGs learnt by the LCD (resp. CKES) algorithm.

We can now make the following observations about the results obtained:

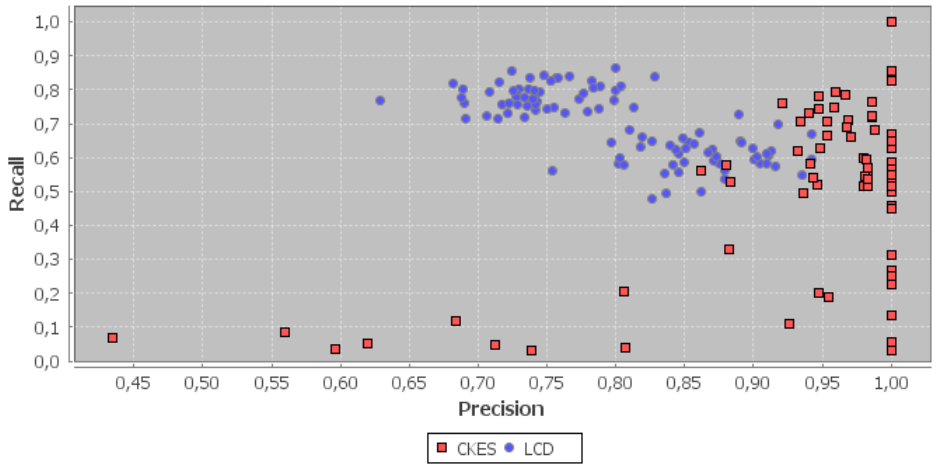

Figure 6: Multiple Locally Optimal CGs May Exist.

Table 1: Average Results For Categorical Data (Left) And Gaussian Data (Right).

CG structure	Recall		Precision	
	CKES	LCD	CKES	LCD
P=10,N=2,n=100	0.97	0.98	0.60	0.59
P=10,N=2,n=30000	0.96	0.96	0.95	0.87
P=10,N=5,n=100	1.00	1.00	0.07	0.06
P=10,N=5,n=30000	0.95	0.98	0.92	0.34
P=20,N=2,n=100	0.97	0.99	0.65	0.64
P=20,N=2,n=30000	0.89	0.95	0.92	0.83
P=20,N=5,n=100	0.98	1.00	0.17	0.15
P=20,N=5,n=30000	0.71	0.95	0.89	0.56

CG structure	Recall		Precision	
	CKES	LCD	CKES	LCD
P=10,N=2,n=100	0.95	0.99	0.83	0.76
P=10,N=2,n=30000	0.93	0.99	1.00	0.99
P=10,N=5,n=100	0.91	0.96	0.44	0.11
P=10,N=5,n=30000	1.00	0.92	1.00	0.57
P=20,N=2,n=100	0.92	0.99	0.99	0.87
P=20,N=2,n=30000	0.94	1.00	1.00	0.98
P=20,N=5,n=100	0.65	1.00	0.72	0.13
P=20,N=5,n=30000	0.77	0.91	1.00	0.53

1. According to Table 1, none of the algorithms is generally able to recreate the CG sampled since their recall and precision values do not equal 1. This means that the sets of independencies detected by the hypothesis test on the sample sets are not faithful to the CGs that generated the sample sets. As we will see, this leads the two algorithms in the evaluation to behave differently.

2. According to Table 1, the LCD algorithm on average achieves good recall but not so good precision. The reason for this may be that the faithfulness assumption makes the LCD algorithm search for a CG that represents all the independencies that are detected in the sample set. However, such a CG may also represent many other independencies. Therefore, the LCD algorithm trades precision for recall.

3. According to Table 1, the CKES algorithm on average achieves good precision but not so good recall. The reason for this may be that the composition property assumption makes the CKES algorithm search for a CG that only represent independencies that are detected in the sample set. However, such a CG may not represented many of the detected independencies. Therefore, the CKES algorithm trades off recall for precision. This can also be seen in Figure 6 where the CKES algorithm achieves good recall only if it also achieves good precision, which indicates that it considers precision as more important.

   In other words, it seems that the faithfulness assumption makes the LCD algorithm overconfident and aggressive, whereas the composition property assumption makes the CKES algorithm cautious and conservative. As we have seen, this implies that in practice the LCD algorithm prioritizes recall and the CKES algorithm precision. In our opinion, the latter behaviour is to be preferred: By achieving better precision, the CGs learnt by the CKES algorithm represent to a greater extent an I map of the probability distribution sampled. And this is the goal in structural learning.

4. The last observation is about which algorithm usually returns the best CG. We can see in Table 2 that the LCD algorithm usually returns the best CG in terms of recall, whereas the CKES algorithm usually returns the best CG in terms of precision. However, note that it is the CKES

Table 2: Best Results For Categorical Data (Top) And Gaussian Data (Bottom).

CG structure	Best recall		Best precision		Best recall and precision	
	CKES	LCD	CKES	LCD	CKES	LCD
P=10,N=2,n=100	6	9	6	4	3	3
P=10,N=2,n=30000	8	5	9	1	8	1
P=10,N=5,n=100	10	10	9	1	6	1
P=10,N=5,n=30000	6	10	10	0	6	0
P=20,N=2,n=100	2	9	8	2	0	2
P=20,N=2,n=30000	1	9	10	0	1	0
P=20,N=5,n=100	3	10	9	1	3	1
P=20,N=5,n=30000	0	10	10	0	0	0

CG structure	Best recall		Best precision		Best recall and precision	
	CKES	LCD	CKES	LCD	CKES	LCD
P=10,N=2,n=100	3	9	9	2	1	2
P=10,N=2,n=30000	3	10	10	9	3	9
P=10,N=5,n=100	6	9	10	0	6	0
P=10,N=5,n=30000	9	2	10	0	8	0
P=20,N=2,n=100	1	9	10	0	1	0
P=20,N=2,n=30000	2	9	9	7	2	7
P=20,N=5,n=100	1	10	10	0	1	0
P=20,N=5,n=30000	4	6	10	0	4	0

algorithm which usually returns the best CG in terms of recall and precision jointly.

6 CONCLUSIONS

In this article, we have presented and proved an extension of Meek's conjecture to CGs under the Lauritzen-Wermuth-Frydenberg interpretation. We have also shown how this conjecture can be used to develop a CG structure learning algorithm that is inclusion optimal for those probability distributions that satisfy the composition property. This property is weaker than faithfulness, which has been a prerequisite for all earlier presented algorithms. We think that the extension of Meek's conjecture proven in this paper is very relevant, since we believe that it will be used by other researchers to propose their own inclusion optimal learning algorithms under the composition property assumption.

In our experiments, we have seen that the CKES algorithm, which only assumes the composition property, produces CGs that can be interpreted as I maps of the probability distribution at hand to a greater extent than the CGs produced by the LCD algorithm, which assumes faithfulness. The reason seems to be that assuming only the composition property induces a more conservative behaviour as compared to assuming faithfulness. CGs are aimed at helping discovering independencies and/or performing probabilistic inference faster but accurately. In both of these tasks it is more important that the independencies in the learnt CG are true in the probability distribution at hand than that all the true independencies are in the CG. In other words, precision is more important than recall for these tasks. The CKES algorithm is therefore, in our humble opinion, preferred to the other algorithms available today.

In our experiments, we have also seen that there may, and often do, exist many local optima for a given data set. This holds even if the data are sampled from a faithful probability distribution. Therefore, it is not wise to just run the learning algorithm once. Instead, we propose to run the algorithm a number of times on the same data and return the best CG found. However, there remains a question to answer for this approach to be applicable in practice: How do we know which of the learnt CGs is the best ? One option is selecting the CG learnt that has the best recall and/or precision w.r.t. the independencies detected by the hypothesis test in the sample set, rather than w.r.t. the separations in the CG sampled as we did in our experiments. Another option is scoring each learnt CG with a score such as the Bayesian information criterion (BIC). We expect that these scores correlate well with the recall and precision values in our experiments. Using a score inside the CKES algorithm instead of the hypothesis test is considered too time consuming due to the fact that there is no closed-form estimates of the maximum-likelihood parameters of a CG. However, we consider it feasible to use a score to rank the CGs learnt by multiple runs of CKES. We are currently working on these two alternative ways to make our approach to CG structure learning applicable in practice.

References

Andersson, S. A., Madigan, D. and Pearlman, M. D. An Alternative Markov Property for Chain Graphs. *In Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, 40-48, 1996.

Chickering, D. M. Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, 3:507-554, 2002.

Cox, D. R. and Wermuth N. Linear Dependencies Represented by Chain Graphs. *Statistical Science*, 8:204-283, 1993.

Drton, M. Discrete Chain Graph Models. *Bernoulli*, 15:736-753, 2009.

Drton, M. and Perlman, M. D. A SINful Approach to Gaussian Graphical Model Selection. *Journal of Statistical Planning and Inference*, 138:11791200, 2008.

Frydenberg, M. The Chain Graph Markov Property. *Scandinavian Journal of Statistics*, 17:333-353, 1990.

Lauritzen, S. L. *Graphical Models*. Oxford University Press, 1996.

Lauritzen, S. L. and Wermuth, N. Graphical Models for Association Between Variables, Some of which are Qualitative and Some Quantitative. *Annual of Statistics*, 17:31-57, 1989.

Ma, Z., Xie, X. and Geng, Z. Structural Learning of Chain Graphs via Decomposition. *Journal of Machine Learning Research*, 9:2847-2880, 2008.

Meek, C. *Graphical Models: Selecting Causal and Statistical Models*. PhD thesis, Carnegie Mellon University, 1997.

Nielsen, J. D., Kočka, T. and Peña, J. M. On Local Optima in Learning Bayesian Networks. *In Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 435-442, 2003.

Peña, J. M. Faithfulness in Chain Graphs: The Discrete Case. *International Journal of Approximate Reasoning*, 50:1306-1313, 2009.

Peña, J. M. Faithfulness in Chain Graphs: The Gaussian Case. *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 588-599, 2011.

Peña, J. M. Learning AMP Chain Graphs under Faithfulness. *In Proceedings of the 6th European Workshop on Probabilistic Graphical Models*, 251-258, 2012.

Studený, M. *Probabilistic Conditional Independence Structures*. Springer, 2005.

Studený, M. On Recovery Algorithms for Chain Graphs. *International Journal of Approximate Reasoning*, 17:265-293, 1997.

Studený, M. and Bouckaert, R. R. On Chain Graph Models for Description of Conditional Independence Structures. *The Annals of Statistics*, 26:1434-1495, 1998.

Studený, M., Roverato, A. and Štěpánová, S. Two Operations of Merging and Splitting Components in a Chain Graph. *Kybernetika*, 45:208-248, 2009.

Volf, M. and Studený, M. A Graphical Characterization of the Largest Chain Graphs. *International Journal of Approximate Reasoning*, 20:209-236, 1999.