A 'Microscopic' Study of Minimum Entropy Search in Learning Decomposable Markov Networks

Y. Xiang, S.K.M Wong and N. Cercone
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-mail: yxiang, wong, nick@cs.uregina.ca
Tel: (306) 585-4088, Fax: (306) 585-4745

Abstract

Several scoring metrics are used in different search procedures for learning probabilistic networks. We study the properties of cross entropy in learning a decomposable Markov network. Though entropy and related scoring metrics were widely used, its 'microscopic' properties and asymptotic behavior in a search have not been analyzed. We present such a 'microscopic' study of a minimum entropy search algorithm, and show that it learns an I-map of the domain model when the data size is large.

Search procedures that modify a network structure one link at a time have been commonly used for efficiency. Our study indicates that a class of domain models cannot be learned by such procedures. This suggests that prior knowledge about the problem domain together with a multi-link search strategy would provide an effective way to uncover many domain models.

Keywords: Inductive learning, reasoning under uncertainty, knowledge acquisition, Markov networks, probabilistic networks.

1 Introduction

A probabilistic network [35, 32, 18, 6] combines a *qualitative* graphic structure which encodes domain dependencies, with a *quantitative* probability distribution which encodes the strength of the dependencies. The network structure can be a directed or undirected graph. A Bayesian network (BN) structure is a directed acyclic graph and a decomposable Markov network (DMN) structure is an undirected chordal graph. As many effective probabilistic inference techniques have been developed [34, 21, 28, 24, 45] and the applicability of probabilistic networks have been amply demonstrated in many artificial intelligence

domains [6], many researchers turn their attention to automatic learning of such networks from data.

Chow and Liu [9] pioneered learning of probabilistic networks. They developed an algorithm to approximate a joint probability distribution (jpd) by a tree-structured BN. Rebane and Pearl [36] extended their method to learn a polytree-structured BN. However, many real world domain models cannot be represented adequately with a tree-structured network. The following algorithms are all applicable to learning a multiply connected network. Herskovits and Cooper [22] developed the Kutato algorithm to learn a BN from a database of cases by minimizing the entropy of the distribution defined by the BN. Their method starts with an empty graph (no links) and adds one link at each pass during search. Later, they proposed the K2 algorithm [10] that learns a BN based on a Bayesian method which selects a BN with the highest posterior probability given a database. A similar algorithm was independently developed by Buntine [3]. Recently, Heckerman et al [20] applied the Bayesian method to learning a BN by combining prior knowledge and statistical data. Spirtes and Glymour [39] developed the PC algorithm that learns a BN by deleting links from a complete graph. Lam and Bacchus [27] applied a minimal description length (MDL) method to learning a BN. A BN is evaluated as the best if it has the minimal sum of its own encoding length and the encoding length of the data given the BN. Instead of learning a BN, Fung and Crawford [14] developed the Constructor algorithm that learns a DMN. Dawid and Lauritzen [11] studies 'hyper Markov laws' in learning numerical parameters of a DMN with a given decomposable graph. Madigan and Raftery [29] proposed algorithms for learning a set of acceptable models expressed as BNs or DMNs. A more extensive review of literature for learning probabilistic networks can be found in [22, 10, 5, 19].

In this paper we consider learning a DMN from a database. Pearl [35] showed that directionality makes BNs a richer language in expressing dependencies. For instance, an induced dependence can be expressed by a BN but not by a DMN. In general, fewer numerical parameters are required to specify a BN than those required to specify a corresponding DMN. However, learning of DMNs is useful for several reasons.

One important application of BNs is to compute posterior probabilities. One efficient exact algorithm [24] for doing that in a sparse multiply connected network uses a DMN, in terms of its junction tree (JT), as the run time representation of a BN. The method can be extended to probabilistic inference with multiply sectioned Bayesian networks in a single agent oriented system [45, 44] as well as in a multi-agent distributed interpretation system [43]. The run time representation is a set of DMNs (in terms of a set of JTs). It has been shown [42, 40] that computation of posterior probabilities of a BN can be performed using an extended relational database once the BN is converted into its corresponding DMN. This implies that once a probabilistic model is expressed in terms of a DMN, inference can be performed using standard relational DBMSs. Most importantly, as BNs and DMNs are so closely related, knowledge gained in learning one of them will benefit the learning of the other.

It has been shown that learning probabilistic networks is NP-hard [2, 8]. Therefore, using heuristic methods in learning is justified. Many algorithms developed use a scoring metric and a search procedure. The scoring metric evaluates the goodness-of-fit of a structure to the data, and the search procedure generates alternative structures and selects the best based on the evaluation.

Out of many possible scoring metrics, Bayesian metrics[1], description length metrics and entropy metrics have been used and studied by several researchers [22, 3, 10, 27, 29, 20, 2, 41]. In many common cases, a Bayesian metric can be constructed that is equivalent to a description length metric, or at least approximately equal. See for instance [7, 38] for a more detailed discussion. Lam and Bacchus [27] showed that, in their scheme for learning a BN based on the MDL principle, the encoding length of the data is a monotonically increasing function of the Kullback-Leibler cross entropy between the distribution defined by the BN and the true distribution. It has also been shown [41] (see Section 3) that the cross entropy of a DMN can be expressed as the difference between the entropy of the distribution defined by the DMN and the entropy of the true distribution which is a constant given a static domain. Entropy has also been used as a means to test conditional independence in learning BNs [36]. Therefore, the maximization of the posterior probability of a network given a database [10, 20], the minimization of description length [27], the minimization of cross entropy between a network and the true model [27], the minimization of entropy of a network [22, 41], and conditional independence tests are all closely related. A better understanding of any of them will lead to a better understanding of all of them.

In all the methods mentioned above, a heuristic method with a single-link lookahead search is adopted in order to avoid the exponential complexity of exhaustive comparison of all possible networks. However, as far as we know, the interplay of the scoring metric and the search process has not been analyzed. Many questions have not been answered. For example, how does the current score determine the next link (dependence) that will be selected? How does the inclusion of a new link change the score and why? Is it possible that once a superfluous link is added, the search may continue until a complete graph structure is generated? We have already had a good 'macroscopic' perspective about which network(s) should be chosen if an exhaustive comparison is possible according to a particular scoring metric. However, in viewing the search process as a chain that connects the initial network to some learned network, we do not seem to have a satisfactory 'microscopic' understanding about what is occurring during the transition from one link to the next on the chain. We do not seem to know how good or how bad the learned network is relative to the global optimal. As pointed out by Spirtes and Glymour [39] and acknowledged by Cooper and Herskovits [10], the "asymptotic reliability

[1] All Bayesian metrics of a model are defined as the posterior probability of the model given the data. However, we have used the plural 'metrics' here since the posterior probability depends on the prior probability of the model, and there can be many possible priors to encode the model learner's a priori bias.

of the procedure is unknown."

In this paper we provide such a 'microscopic' study under the context of learning a DMN from a database by using an entropy scoring metric and a minimum entropy search procedure. The 'microscopic' understanding leads to the identification of drawbacks of a single-link lookahead search which is commonly used in learning probabilistic networks.

It is well known that *parity* functions cause failure of many decision tree learning algorithms (see [33, 25] for example). We show that a class of probabilistic domain models, that forms a generalization of parity functions, cannot be learned by a single-link lookahead search procedure.[2] Although our observation is based on the entropy scoring metric, because of the close relationship between the entropy metric and other metrics described above, the results we obtain are valid for other algorithms as well. We demonstrate that these domain models cannot be learned by many learning algorithms [22, 39, 27]. We therefore propose a multi-link lookahead learning algorithm. We will analyze the computational complexity of this algorithm and suggest solutions to alleviate the problem.

This 'microscopic' study also establishes the 'asymptotic' behavior of the minimum entropy search algorithm. We will show that, when the number of cases in a database becomes very large, the algorithm will halt and return an I-map of the domain model.

In practice, learning is performed on a database of a finite size. A finite database may contain *false* dependencies that do not exist among the domain variables.[3] They cause the learning algorithms to generate superfluous links. These links and their associated numerical probability values tend to encode 'noise' and bias the jpd of the learned networks. Even when the database is very large and contains no false dependencies, learning using heuristic search may generate superfluous links that do not reflect the true domain dependencies. These superfluous links tend to make the inference using the resultant network unnecessarily more complex. Fortunately, after we classify superfluous links generated under different conditions, it is revealed that the entropy metric has the built-in resistance to adding some superfluous links. Thus, learning a trivial I-map is unlikely.

Section 2 provides the background and terminology. We present in Section 3 the rational of the minimum entropy approach. In Section 4, we study the 'microscopic' mechanism of the minimum entropy search in learning a decomposable Markov network as an I-map of a domain model. We will also discuss the built-in partial resistance of the entropy metric to adding superfluous links. In Section 5, we demonstrate the limitation of a single-link lookahead search. We present in Section 6 a multi-link lookahead algorithm based on the minimum entropy search. Experimental results are presented in Section 7,

[2]However, other methods such as stochastic search [30, 17] may not have this problem.

[3]We consider learning processes that infer dependencies contained in the domain model. As we have no direct access of the model, we must infer dependencies from the data generated by the model. Put differently, we must infer true dependencies from those dependencies that are 'contained' in the data.

followed by a concluding discussion.

2 Background and Terminology

2.1 Graph related terminology

A *chord* in an undirected graph is a link that connects two nonadjacent nodes. A graph is *chordal* if every cycle of length > 3 has a chord. The undirected graph G_1 in Figure 1 is *not* chordal since the cycle a, (a, b), b, (b, d), d, (d, c), c, (c, a), a of length 4 has a pair of nonadjacent nodes b and c that are unconnected. If we add the chord (b, c) to G_1, it becomes G_2 which is chordal. A *clique* of a graph is a maximal set of nodes pairwise linked. G_2 has four cliques $\{a, b, c\}$, $\{b, c, d\}$,$\{c, e\}$ and $\{c, f\}$. A *component* of a graph is a maximal subgraph that is connected. In Figure 1, G_2 has a single component and G_3 has two components. An undirected graph G of a set N of nodes is *collapsible* onto $Z \subset N$ if every component of $N \setminus Z$ has complete boundary in G. We shall refer to Z as a *core*. In Figure 1, $\{b, c, f\}$ is a core of G_3, and so is ϕ.

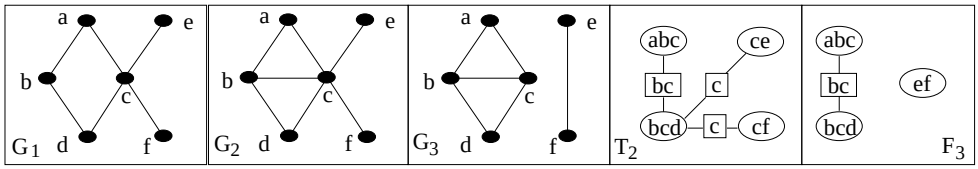

Figure 1: G_1: a non-chordal graph. G_2: a chordal graph with a single component. G_3: a chordal graph with two components. T_2: a junction tree of G_2, where nodes are drawn as ovals and sepsets are drawn as boxes. F_3: a junction forest of G_3.

Let G be a connected chordal graph. A *junction tree* (JT) T of G is a tree whose nodes are labeled by cliques of G such that for each pair of nodes of T, their intersection is contained in every node on the unique path between them. A connected graph has a JT iff (if and only if) the graph is chordal [16]. In Figure 1, T_2 is a JT of G_2. Without confusion, we sometimes refer to a *node* C in T as a *clique* when the nodes of G contained in C are of interest. For instance, we may say that T_2 has a clique $\{a, b, c\}$. The intersection of two adjacent cliques in T is called the *sepset* of the two cliques. In general, G may not be connected. A *junction forest* (JF) F of G is a set of JTs each of which is a JT of one component of G. In Figure 1, F_3 is a JF of G_3 and F_3 consists of two JTs. T_2 is a (trivial) JF of G_2. Extending the relation between chordal graphs and JTs to JFs, we have that a graph has a JF iff it is chordal. Due to the equivalence relation, we shall switch freely between the two graphical representations (chordal graphs and JFs) at convenience.

Let X, Y and Z be three subsets of nodes in a graph. We use $< X|Z|Y >$ to mean that nodes in Z intercept all paths between nodes of X and nodes of

Y. In G_2 of Figure 1, we have $< \{a\}|\{b,c\}|\{d\} >$. In a JF, we use $< X|Z|Y >$ to mean that Z is a sepset or a clique or the union of a set of cliques which intercept the unique path between the clique that contains X and the clique that contains Y. For example, $< \{a\}|\{b,c\}|\{d\} >$ and $< \{e\}|\{b,c,d\}|\{f\} >$ are true in T_2 and $< \{a,b,c\}|\phi|\{ef\} >$ is true in T_3.

2.2 Dependency graphs

Let N be a set of discrete variables in a problem domain and $X \subseteq N$. A *configuration* x of X is an assignment of values to every variable in X. A *probabilistic model* (PM) over N is an encoding of probabilistic information that determines the probability of every configuration of X for every $X \subseteq N$. A PM over N can be specified by a jpd over N. The entropy of X defined by a probability distribution P over X is $H(X) = -\sum_x P(x) \log P(x)$.

We will denote a PM by $\mathcal{M}$. Our task is to learn a probabilistic network from the data generated by $\mathcal{M}$. In practice, we usually have less data than what is necessary to reliably estimate the jpd over N. However, we may be able to estimate reliably the marginal distribution over $X \subset N$ if $|X|$ is small. Therefore, the jpd over N is mainly used in this paper as a conceptual entity.

Let X, Y and Z be three subsets of N. X and Y are *conditionally independent* given Z, denoted $Ind(X, Z, Y)$, iff $P(x|y\ z) = P(x|z)$ whenever $P(y\ z) > 0$.

Since we use graphs to represent independence relations among variables, we will use *nodes* and *variables* interchangeably. An undirected graph G is an *independence map (I-map)* of $\mathcal{M}$ over N if there is an one-to-one correspondence between nodes of G and variables in N such that for all disjoint subsets X, Y and Z of N, we have $< X|Z|Y > \ \Rightarrow \ Ind(X, Z, Y)$. That is, in an I-map, variables that are graphically separated are independent. Variables not graphically separated, however, are not necessarily dependent. See Pearl [35] for more details on graphical representation of dependence models.

Let $G = (N, E)$ be a chordal graph, F be a JF of G, and $\mathcal{M}$ be a PM over N. Let C be a clique of F and S be a sepset of F. Let $P_{\mathcal{M}}(C)$ and $P_{\mathcal{M}}(S)$ be the marginal distributions over C and S, respectively, defined by $\mathcal{M}$. The jpd

$$P(v) = (\prod_C P_{\mathcal{M}}(c))/(\prod_S P_{\mathcal{M}}(s)) \quad whenever\ P(v) > 0$$

is called the *projected distribution of* $\mathcal{M}$ *on* G *(or on* F*)*, where v is a configuration of N, c is the projection of v to C, and s is the projection of v to S. Note that since the structure of a DMN is chordal, positivity is not required [18, 11]. The pair (G, P) is the *decomposable Markov network* (DMN) obtained by projecting $\mathcal{M}$ to G, where G is the *structure* of the DMN and P is the distribution of the DMN[4]. For simplicity, we shall call (G, P) the DMN from $\mathcal{M}$.

[4]What we call a *decomposable Markov network* has been termed differently in the literature. It is called simply *Markov network* in [14, 41] and *Markov graph* in [10]. The term *decomposable Markov network* is implicitly used in [35] to mean the similar thing as defined above. However, there the term *Markov network* is restricted to a minimal I-map of a given dependence model. We do not require the structure of a DMN to be an I-map.

In practice, $P_{\mathcal{M}}(C)$ is estimated from the data generated by $\mathcal{M}$. Note that (G, P) defines a PM which may or may not be equivalent to $\mathcal{M}$. The entropy of N defined by P can be shown [41] to be

$$H(N) = \sum_C H(C) - \sum_S H(S). \tag{1}$$

Whenever $< X|Z|Y >$ holds in G (or F), $Ind(X, Z, Y)$ must hold in P. Therefore, we say that $Ind(X, Z, Y)$ is *implied* by G (or F).

3 The Rational of the Minimum Entropy Approach

This section briefly reviews the rational behind the minimum entropy approach originally presented in [41].

Given a probabilistic model $\mathcal{M}$ over a set N of variables, we would like to learn a DMN (G, P) that is an approximation of $\mathcal{M}$. To measure the *closeness* of (G, P) to $\mathcal{M}$, we adopt the Kullback-Leibler cross entropy [26], $K(P_{\mathcal{M}}, P) = \sum_v P_{\mathcal{M}}(v) \log(P_{\mathcal{M}}(v)/P(v))$, where $P_{\mathcal{M}}$ is the *true* jpd defined by $\mathcal{M}$ and v is a configuration of N. A DMN that minimizes $K(P_{\mathcal{M}}, P)$ will be considered as the *best* approximation of $\mathcal{M}$. It has been shown [41] that

$$K(P_{\mathcal{M}}, P) = H(N) - H_{\mathcal{M}}(N), \tag{2}$$

where $H(N)$ is the entropy of N defined by P and $H_{\mathcal{M}}(N)$ is defined by $\mathcal{M}$. We include the proof here to make the paper self-contained.

Since $K(P_{\mathcal{M}}, P) = -\sum_v P_{\mathcal{M}}(v) \log P(v) - H_{\mathcal{M}}(N)$, it suffices to show $-\sum_v P_{\mathcal{M}}(v) \log P(v) = H(N)$. From the definition of $P(N)$, we obtain

$$\begin{aligned}
-\sum_v P_{\mathcal{M}}(v) \log P(v) &= -\sum_v P_{\mathcal{M}}(v)(\sum_C \log P_{\mathcal{M}}(c) - \sum_S \log P_{\mathcal{M}}(s)) \\
&= -\sum_C \sum_v P_{\mathcal{M}}(v) \log P_{\mathcal{M}}(c) + \sum_S \sum_v P_{\mathcal{M}}(v) \log P_{\mathcal{M}}(s) \\
&= -\sum_C \sum_c P_{\mathcal{M}}(c) \log P_{\mathcal{M}}(c) + \sum_S \sum_s P_{\mathcal{M}}(s) \log P_{\mathcal{M}}(s) \\
&= \sum_C H(C) - \sum_S H(S).
\end{aligned}$$

where c (s) is the projection of x to C (S). The result follows from (1).

According to (2), minimizing $K(P_{\mathcal{M}}, P)$ can be achieved by simply minimizing $H(N)$. We call this the *minimum entropy approach*.

In Section 4.1, we will show that a DMN that minimizes $K(P_{\mathcal{M}}, P)$ is actually an I-map of $\mathcal{M}$. Thus, the best DMN is a *minimal* I-map, i.e., an I-map that contains no superfluous links. The problem of learning a minimal I-map is NP-hard [2]. Therefore, using heuristic methods in learning is justified. We can design a learning algorithm by combining the entropy metric with a single-link lookahead search strategy. We will refer to such an algorithm as

learning by minimum entropy search. One such algorithm [41] starts with an empty graph. At each pass, it searches all possible links and adds to the current graph the link that minimizes the entropy. It terminates when no additional link can decrease the entropy *significantly*. In Section 5, we identify a class of PMs that cannot be learned by such a single-link lookahead search. A multi-link lookahead search is required to discover the dependencies in these PMs. In the following discussion, we will assume a more general search procedure with the single-link lookahead as a special case.

4 The Minimum Entropy Search

In this section, we analyze how the dependence relations in a DMN are derived in the minimum entropy search.

Recall that the pair (G, P) is a DMN from $\mathcal{M}$. That is, P is defined by the marginals of $P_{\mathcal{M}}$ on cliques of G. In practice, we can only estimate these marginals from a database of cases, e.g., using the maximum-likelihood estimator (the relative frequencies) or the Bayes's estimator [10]. According to the *law of large numbers*, the relative frequency of each configuration approaches its true probability as the size of the database approaches infinity. Since our objective here is to analyze the 'microscopic' mechanism of the minimum entropy search and its asymptotic behavior, one may assume that P is obtained directly from the projection of $P_{\mathcal{M}}$. As we move from the theoretical analysis to practical implementation in Section 6, we will discuss the related issues.

Let us outline the theorems to be presented in this section. Theorem 2 establishes the relationship between the entropy of a DMN and its I-mapness. Theorem 3 identifies a false independence relation in a DMN if its entropy is not the minimum. Theorem 5 says that if the inclusion of one or more links can remove a false independence relation, the entropy of the DMN will decrease. Together, Theorems 3 and 5 state that the process of decreases in entropy closely parallels the process of removal of false independence relations contained in the intermediate DMNs. Theorem 6 summerizes Theorems 2, 3 and 5. It asserts that the minimum entropy search algorithm will produce an I-map.

4.1 Characterization of the search space

Let us first show that the entropy of a DMN cannot be smaller than that of the underlying $\mathcal{M}$. This means that the search space of DMNs is lower-bounded in terms of the entropy scoring metric as indicated by the following corollary.

Corollary 1 *Let $\mathcal{M}$ be a probabilistic model over a set N of variables. Let (G, P) be a decomposable Markov network from $\mathcal{M}$. Let $H_{\mathcal{M}}(N)$ be the entropy of N defined by $\mathcal{M}$ and $H(N)$ be the entropy of N defined by P. Then $H(N) \geq H_{\mathcal{M}}(N)$.*

Proof:

Let $P_{\mathcal{M}}$ be the jpd defined by $\mathcal{M}$. The cross entropy $K(P_{\mathcal{M}}, P) \geq 0$ [26]. Since $K(P_{\mathcal{M}}, P) = H(N) - H_{\mathcal{M}}(N)$ by (2), we obtain $H(N) \geq H_{\mathcal{M}}(N)$. □

The following theorem says that the lower bound of the search space can only be reached by a DMN that is an I-map of $\mathcal{M}$. Therefore, it shows clearly that the minimum entropy search targets an I-map.

Theorem 2 *Let $\mathcal{M}$ be a probabilistic model over a set N of variables. Let (G, P) be a decomposable Markov network from $\mathcal{M}$. Let $H_{\mathcal{M}}(N)$ be the entropy of N defined by $\mathcal{M}$ and $H(N)$ be the entropy of N defined by P. Then $H(N) = H_{\mathcal{M}}(N)$ iff G is an I-map of $\mathcal{M}$.*

Proof:

The cross entropy $K(P_{\mathcal{M}}, P) = 0$ iff $P = P_{\mathcal{M}}$ [26]. According to (2), $H(N) = H_{\mathcal{M}}(N)$ is equivalent to $P = P_{\mathcal{M}}$. Since (G, P) is a DMN, we form a JF of G and have $P = \prod_C P_{\mathcal{M}}(C) / \prod_S P_{\mathcal{M}}(S) = P_{\mathcal{M}}$. This means that every independence relation implied by G is true in $\mathcal{M}$, namely, G is an I-map of $\mathcal{M}$. □

4.2 Construction of an I-map

Theorem 2 implies that if the entropy of a DMN is not the minimum, it must contain a false independence relation. The next theorem describes such a false independence relation more specifically in terms of its topological features.

Theorem 3 *Let $\mathcal{M}$ be a probabilistic model over a set N of variables. Let (G, P) be a decomposable Markov network from $\mathcal{M}$. Let $H_{\mathcal{M}}(N)$ be the entropy of N defined by $\mathcal{M}$ and $H(N)$ be the entropy of N defined by P.*

Then $H(N) > H_{\mathcal{M}}(N)$ iff there exists a core $X \cup Y \cup Z$ in G such that $Ind(X, Z, Y)$ holds in P but not in $\mathcal{M}$, where X, Y and Z are disjoint subsets of N, $X \neq \phi$, $Y \neq \phi$, and Z is also a core in G.

Proof:

The sufficiency is an immediate result of Theorem 2. We prove the necessity by contraposition. Since (G, P) is a DMN from $\mathcal{M}$, G implies a set of independence relations of the form $Ind(X, Z, Y)$ where X, Z and Y are defined as in the theorem. Note that $< X|Z|Y >$ in G follows from the condition that Z and $X \cup Y \cup Z$ are cores in G. If every such $Ind(X, Z, Y)$ holds in $\mathcal{M}$, then we have

$$P = \prod_C P_{\mathcal{M}}(C) / \prod_S P_{\mathcal{M}}(S) = P_{\mathcal{M}},$$

namely, $H(N) = H_{\mathcal{M}}(N)$. □

Proposition 4 provides a method to augment the structure of a DMN such that a false independence relation as identified in Theorem 3 can be removed, and the resultant graph is chordal.

Proposition 4 *Let G be a chordal graph of a set N of nodes. Let $X \cup Y \cup Z$ be a core in G such that $< X|Z|Y >$ holds in G, where X, Y and Z are disjoint subsets of N, $X \neq \phi$, $Y \neq \phi$, and Z is also a core in G. Let G' be a graph obtained from G by completing $Z \cup \{x, y\}$, where x and y are selected as follows: If there exist $x \in X$ and $z \in Z$ such that x and z are connected in G, then select x. Otherwise select $x \in X$ arbitrarily. Select $y \in Y$ similarly.*

Then G' is chordal and $< X|Z|Y >$ is false in G'.

Proof:

When $Z = \phi$, X and Y are in different components. The proposition is trivially true in this case. We consider $Z \neq \phi$ (X, Y and Z are in the same component) below.

To show G' is chordal, we first modify G into G_1 by completing Z if Z is not already complete. Since G is chordal and Z is a core, it follows that G_1 is chordal.

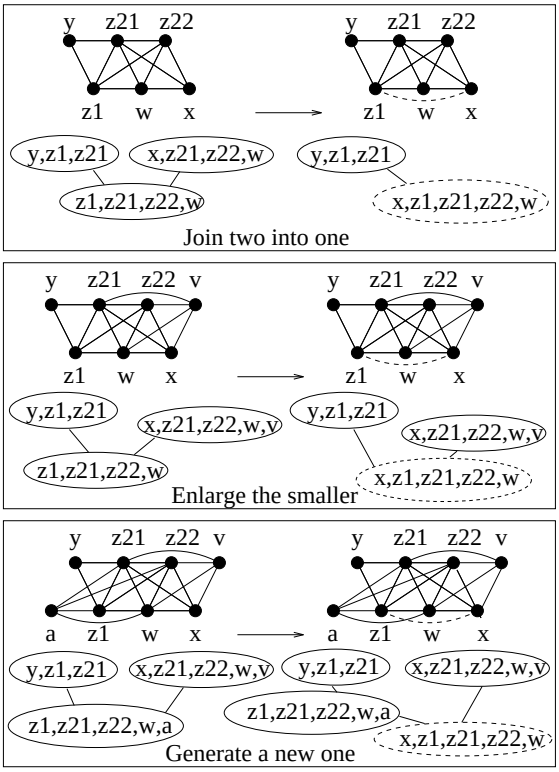

Figure 2: Three cases of completing $Z \cup \{x\}$. In each box, the upper-left is a chordal graph where $< x|Z|y >$ holds. The lower-left is a JT of the chordal graph. The upper right is the chordal graph obtained by adding the dashed link to the upper left graph such that $Z \cup \{x\}$ is completed. The lower-right is a JT of the augmented graph where the modified or newly created clique is shown in dashed oval. Top box: two cliques are joined into one. Middle box: the smaller clique of the two involved is enlarged. Bottom box: a new clique is created.

Next, we modify G_1 into G_2 (F_1 into F_2) by completing $Z \cup \{x\}$ if they are not already complete. Since x and Z are connected but not complete, there exists a JF F_1 of G_1 in which x is in a clique adjacent to the clique that contains Z. The three exhaustive and exclusive consequences of completing $Z \cup \{x\}$ are (1) the two adjacent cliques are joined into one (e.g., Figure 2, top), (2) the smaller of the two is enlarged (e.g., Figure 2, middle), and (3) a new clique is generated on the path between the two (e.g., Figure 2, bottom). Which one of them occurs depends on the composition of the two cliques. In all cases, the junction forest property is unchanged and hence F_2 is a JF, which implies that G_2 is chordal.

Using a similar argument to completing $Z \cup \{x, y\}$, we conclude that G' is chordal. □

Now we want to show that if a DMN is augmented such that a false independence relation is removed, then the augmentation will decease the entropy. This implies that the minimum entropy search is precisely a process of removal of false independence relations.

Theorem 5 *Let $\mathcal{M}$ be a probabilistic model over a set N of variables. Let (G_0, P_0) and (G_1, P_1) be two decomposable Markov networks of $\mathcal{M}$ where G_0 is a subgraph of G_1. Let $H_i(N)$ ($i = 0, 1$) be the entropy defined by P_i.*

Then $H_1(N) < H_0(N)$ iff there exist three disjoint subsets $X \neq \phi$, $Y \neq \phi$ and Z of N such that $Ind(X, Z, Y)$ is implied by G_0 but not by G_1 and $\mathcal{M}$.

Proof:

First, we show $H_1(N) \leq H_0(N)$. Note that (G_1, P_1) defines a PM and therefore (G_0, P_0) is a DMN of (G_1, P_1) since G_0 is a subgraph of G_1. From Corollary 1, $H_1(N) \leq H_0(N)$. The equality holds iff G_0 is an I-map of (G_1, P_1) due to Theorem 2. That is, every independence relation implied by G_0 must be implied by G_1, otherwise the relation must hold in $\mathcal{M}$. □

Theorem 6 says that, started with an arbitrary DMN, if the entropy of the current DMN is not the minimum, a sequence of DMNs can be found which monotonically decreases the entropy to its minimum. This therefore establishes the asymptotic behavior of the minimum entropy search.

Theorem 6 *Let $\mathcal{M}$ be a probabilistic model over a set N of variables. Let (G, P) be a decomposable Markov network from $\mathcal{M}$. Let $H(N)$ be the entropy of N defined by P, and $H_{\mathcal{M}}(N)$ defined by $\mathcal{M}$. If $H(N) > H_{\mathcal{M}}(N)$, there exists a sequence $(G, P) = (G_0, P_0), \ldots, (G_k, P_k)$ of decomposable Markov networks[5] from $\mathcal{M}$ with the corresponding sequence of entropies $H(N) = H_0(N) > \ldots > H_k(N) = H_{\mathcal{M}}(N)$, where G_i ($i = 1, \ldots, k$) is constructed by adding links to G_{i-1}, and the last graph G_k is an I-map of $\mathcal{M}$.*

[5]Frydenberg and Lauritzen [13] (p553) proved that, given two chordal graphs with one being the subgraph of the other, there is an increasing sequence of chordal graphs between them that differ by exactly one link. DMNs differing by one link are *not* sufficient to decrease entropy as will be shown in Section 5. Our result here involves a sequence of chordal graphs that differ by more than one links and fix some false independence relations.

Proof:

Suppose $H(N) > H_{\mathcal{M}}(N)$. By Theorem 3, an independence relation $Ind(X, Z, Y)$ that holds in P but not in $\mathcal{M}$ can be found, where $X \neq \phi$, $Y \neq \phi$ and Z are disjoint, Z is a core in G and so is $X \cup Y \cup Z$.

By Proposition 4, a chordal graph G_1 can be obtained by augmenting G such that $Ind(X, Z, Y)$ is false in G_1.

Projecting $P_{\mathcal{M}}$ to G_1, we obtain a new DMN (G_1, P_1). Since $Ind(X, Z, Y)$ is implied by G but not by G_1 (a supergraph of G) and $\mathcal{M}$, according to Theorem 5, (G_1, P_1) satisfies $H_1(N) < H_0(N)$. If $H_1(N) > H_{\mathcal{M}}(N)$, the above arguments lead to (G_2, P_2) that satisfies $H_2(N) < H_1(N)$. Since only a finite number of links can be added to G and the entropy of a DMN with a complete graph is equal to $H_{\mathcal{M}}(N)$, the sequence $(G_0, P_0), \ldots, (G_k, P_k)$ of DMNs does exist. By Theorem 2, G_k is an I-map of $\mathcal{M}$. □

Theorem 6 illustrates the 'microscopic' working mechanism of the minimum entropy search. The entropy acts as a motor that drives the search for identifying a false independence relation. The removal of the false independence moves the current state forward in a chain leading the starting DMN to the goal I-map.

4.3 Superfluous links

Theorem 6 ensures that the minimum entropy search halts and produces an I-map. It does not, however, eliminate the possibility of producing a trivial I-map. Now we want to show that in practice halting at a trivial I-map rarely occurs. We identify two types of superfluous links that may be added. In fact, the entropy scoring metric has some built-in resistance to adding these two types of superfluous links. However, the entropy scoring metric has no resistance to adding a third type of superfluous links. We discuss this third type in Section 6.

We start the search with an empty DMN. At each pass, links are added to correct a false independence relation and thus the entropy is reduced. Eventually we will obtain an I-map of $\mathcal{M}$. To examine the possibility of halting at a trivial I-map, we ask the following two questions:

1. Will those links that do not correct a false independence reduce the entropy?

2. Can the entropy scoring metric distinguish a *direct* dependence from an *indirect* dependence?

The first question concerns the possibility of adding what we refer to as *uncalled-for* links. For example, suppose the graph in the left of Figure 3 is the minimal I-map of a PM. Assume that the current learned structure is the graph in the right with the links (c, e) and (d, f) missing. If the link (a, c) is added next, it is an uncalled-for link since it does not correct any false independence in the current structure. The answer to the first question is definitely *no*, which is a direct result of Theorem 5.

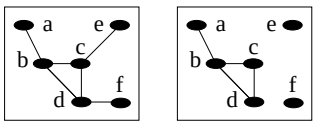

Figure 3: Left: A minimal I-map of a PM. Right: An intermediate structure during learning.

The second question concerns the inclusion of what we refer to as *redundant* links. Redundant links repair a false independence but not in the most direct way. In Figure 3 (right), since e is disconnected from the rest of the graph, it implies that e is independent of every other variable. This is a false independence since e is connected to every other variable in the minimal I-map (left). In Figure 3 (right), if the link (a, e) is added next, it is a redundant link. It repairs the false independence between a and e. Since it does not repair the false independence between c and e, the link (c, e) must eventually be included, rendering (a, e) redundant.

Note that the classification of a particular superfluous link into uncalled-for versus redundant depends on the current structure. If the current structure already contains (c, e), the link (a, e) would be classified as uncalled-for rather than redundant.

An intermediate structure may imply many false independence relations. In order not to include too many redundant links, we must not correct just any false relation. Proposition 7 shows that the number of redundant links can be reduced if we choose to correct the false relation that maximizes the decrement of entropy. It says that, given three subsets A, B and C of variables, if A and B are dependent, and A and C are either marginally independent or conditionally independent given B, then including links between A and B reduces entropy more than including links between A and C. This result formally justifies the use of a *greedy* search and provides a partial answer to the second question.

Proposition 7 *Let $\mathcal{M}$ be a probabilistic model over a set N of variables. Let $G = (N, E)$ be a chordal graph, A, B and C be three distinct cliques of G and A be disconnected from B and C. Let G_1 be a chordal graph formed by only adding links to G such that $A \cup B$ becomes a clique. Let G_2 be a chordal graph formed by only adding links to G such that $A \cup C$ becomes a clique. Let $H_1(N)$ and $H_2(N)$ be entropies defined by decomposable Markov networks from $\mathcal{M}$ with structures G_1 and G_2, respectively.*

Then $H_1(N) < H_2(N)$ if (1) $Ind(A, \phi, B)$ does not hold in $\mathcal{M}$ and (2) either $Ind(A, \phi, C)$ holds in $\mathcal{M}$, or $Ind(A, B, C)$ holds in $\mathcal{M}$ but $Ind(A, C, B)$ does not.

Proof:

In G_1, a new clique AB replaces cliques A and B. Hence, we have $H_1(N) = H(N) + H_{\mathcal{M}}(AB) - H_{\mathcal{M}}(A) - H_{\mathcal{M}}(B)$, where $H(N)$ is the entropy of the DMN with the structure G, and $H_{\mathcal{M}}(AB)$ is the entropy of the new clique defined

by $\mathcal{M}$. Similarly, $H_2(N) = H(N) + H_{\mathcal{M}}(AC) - H_{\mathcal{M}}(A) - H_{\mathcal{M}}(C)$. Therefore, we have

$$H_2(N) - H_1(N) = H_{\mathcal{M}}(AC) - H_{\mathcal{M}}(C) - H_{\mathcal{M}}(AB) + H_{\mathcal{M}}(B).$$

Using the well known *average mutual information* between two sets U and V of variables,

$$I(U;V) = \sum_{UV} P(UV) \log \frac{P(UV)}{P(U)P(V)}$$

we obtain

$$\begin{aligned} & H_2(N) - H_1(N) \\ = \; & [H_{\mathcal{M}}(A) + H_{\mathcal{M}}(C) - I_{\mathcal{M}}(A;C)] - H_{\mathcal{M}}(C) - [H_{\mathcal{M}}(A) + H_{\mathcal{M}}(B) - I_{\mathcal{M}}(A;B)] + H_{\mathcal{M}}(B) \\ = \; & I_{\mathcal{M}}(A;B) - I_{\mathcal{M}}(A;C). \end{aligned}$$

If $Ind(A, \phi, C)$ holds in $\mathcal{M}$, then $I_{\mathcal{M}}(A;C) = 0$. Since $Ind(A, \phi, B)$ does not hold in $\mathcal{M}$, we have $H_2(N) - H_1(N) = I_{\mathcal{M}}(A;B) > 0$.

On the other hand, if $Ind(A, B, C)$ holds in $\mathcal{M}$, then $I_{\mathcal{M}}(A;B) = I_{\mathcal{M}}(A;C) + I_{\mathcal{M}}(A;B|C)$ [15] (equation 2.3.18), where $I(A;B|C)$ is the *average conditional mutual information* between A and B given C,

$$I(A;B|C) = \sum_{ACB} P(ACB) \log \frac{P(A|CB)}{P(A|C)}.$$

Hence, $I_{\mathcal{M}}(A;B) - I_{\mathcal{M}}(A;C) = I_{\mathcal{M}}(A;B|C) \geq 0$, with equality iff $Ind(A, C, B)$ holds in $\mathcal{M}$. Since $Ind(A, C, B)$ does not hold by assumption, $I_{\mathcal{M}}(A;B) - I_{\mathcal{M}}(A;C) > 0$. □

Proposition 7 answers the second question partially. It only asserts that redundant links will never be added under certain conditions, but it does not guarantee the total avoidance of such links. Since finding the minimal I-map is NP-hard, it is unlikely that any heuristic search using any scoring metric would be able to eliminate all redundant links.

5 When Will a Single-Link Search Fail?

Theorem 6 states that as long as the current DMN is not yet an I-map, a set of links can always be added such that the new DMN is closer to an I-map. No upper bound is given for the number of links that must be added each time. If we use a greedy algorithm as suggested by Proposition 7, a single-link lookahead search needs only to explore $O(|N|^2)$ links before one link is added. The number of links to be explored increases to $O(|N|^{2i})$ for an i-link lookahead (see Section 6). The single-link lookahead search has been adopted by several learning algorithms [22, 10, 3, 39, 27, 41] for computational efficiency. However, an important question is unanswered: What might be compromised by using a single-link lookahead search?

With the understanding of the minimum entropy search, we answer the above question in this section. Theorem 8 shows the existence of a class of

PMs that displays a special pattern of dependence relations. Theorem 9 shows that a single-link lookahead search is unable to learn the I-map for this class of PMs.

Theorem 8 *For any integer $\eta \geq 3$, there exists a collection $\mathcal{C}$ of probabilistic domain models over a set N of η binary variables such that the following holds for each $\mathcal{M} \in \mathcal{C}$.*

(S1) *For each $Y \in N$, $P_{\mathcal{M}}(N \setminus \{Y\}) = \prod_{X \in N, X \neq Y} P_{\mathcal{M}}(X)$.*

(S2) *For each pair $X, Y \in N$ and $X \neq Y$, $Ind(\{X\}, N \setminus \{X, Y\}, \{Y\})$ does not hold in $\mathcal{M}$.*

We shall refer to each $\mathcal{M}$ as a `pseudo-independent` *(PI) model.*

Before proving the theorem, we intuitively describe the dependency pattern displayed by the PI models. S2 implies that no pair of variables of N are independent given all other variables. Therefore, in any I-map $G_{\mathcal{M}}$ of $\mathcal{M}$, there must be a *direct* line between each pair of them, i.e., $G_{\mathcal{M}}$ is a complete graph. We say that variables in such PMs are *collectively dependent*. On the other hand, S1 implies that variables in any subset of N of size $\eta - 1$ are *pairwise marginally independent*.

Proof:

It is sufficient to construct a parameterized jpd given η such that both S1 and S2 hold and the parameter can take on infinite possible values.

Let $X_1, \ldots, X_\eta$ denote η binary variables in N and $P(X_{i,0}) = P(X_{i,1}) = 0.5$ $(i = 1, \ldots, \eta)$ where $X_{i,0}$ and $X_{i,1}$ are the two outcomes of X_i. There are exactly η distinct subsets of N of size $\eta - 1$. For each subset $\{X_{i_1}, \ldots, X_{i_{\eta-1}}\}$ where $1 \leq i_j \leq \eta$, S1 is equivalent to

$$P(X_{i_1}, \ldots, X_{i_{\eta-1}}) = 0.5^{\eta-1}.$$

We have omitted the second index because the particular configuration does not affect the probability value. Models that satisfy S1 do exist. A jpd $P^*(N) = P(X_1, \ldots, X_\eta) = 0.5^\eta$ is one example. However, $P^*(N)$ does not satisfy S2. We will construct a jpd of $\mathcal{M}$ which satisfies both S1 and S2.

We can view the above condition, which is equivalent to S1, as a constraint

$$P(X_{i_1}, \ldots, X_{i_{\eta-1}}) = P(X_{i_1}, \ldots, X_{i_{\eta-1}}, X_{i_\eta,0}) + P(X_{i_1}, \ldots, X_{i_{\eta-1}}, X_{i_\eta,1}) = 0.5^{\eta-1}$$

on the subset $\{X_{i_1}, \ldots, X_{i_{\eta-1}}\}$. We therefore have η constraints, one for each such subset.

To construct a desired jpd, we assign a probability value to each of the 2^η configurations, each of which is denoted by a binary η-tuple. For example, the configuration $(X_{1,0}, \ldots, X_{\eta,0})$ is denoted $(0, \ldots, 0)$. We group the tuples according to the number of 1's contained in each tuple and index the groups as $GP_0, \ldots, GP_\eta$. For example, GP_0 has a single tuple $(0, \ldots, 0)$, GP_1 has η tuples $(0, \ldots, 0, 1)$, $(0, \ldots, 0, 1, 0)$, ..., and $(1, 0, \ldots, 0)$.

We assign probability values to the configurations group by group in ascending order of the group index. To make a new assignment, we check the configurations whose probability values have been assigned, determine how many of the η constraints are involved in the assignment, and ensure that the new assignment conforms to the constraints.

We start by assigning the single configuration in GP_0: $P(0,\ldots,0) = 0.5^{\eta-1}q$, where $q \in [0,1]$ and $q \neq 0.5$. This assignment does not violate any constraints. We then assign a configuration in GP_1:

$$P(0,\ldots,0,1) = P(X_{1,0},\ldots,X_{\eta-1,0}) - P(X_{1,0},\ldots,X_{\eta-1,0},X_{\eta,0}) = 0.5^{\eta-1}(1-q).$$

Note that this assignment involves only one constraint and involves the only configuration whose value has been assigned. We will say that the assignment of probability value to configuration $(0,\ldots,0,1)$ involves the constraint *relative to* the configuration $(0,\ldots,0,0)$.

We make the following observation: If c_1 is a configuration whose probability has been assigned and c_2 is a configuration whose probability is to be assigned, then the assignment involves a constraint relative to c_1 if and only if c_1 and c_2 differ by exactly one attribute.

This observation leads to two implications. First, the assignment of c_2 cannot involve a constraint relative to another configuration in the same group, since configurations in the same group differ by at least two attributes. For example, $(0,\ldots,0,1)$ and $(0,\ldots,1,0)$ in GP_1, and $(0,\ldots,0,1,1)$ and $(0,\ldots,1,0,1)$ in GP_2.

Second, if $c_2 \in GP_i$, the assignment of c_2 can only involve a constraint relative to configurations in GP_{i-1}. This is because configurations in GP_j $(j \leq i-2)$ differ from c_2 by at least two 1's. Therefore, when we assign a configuration, we have only to check configurations in the very last group assigned. Note that the assignment may still involve multiple constraints each relative to a distinct configuration. For example, the assignment of $(0,\ldots,0,1,1,1)$ in GP_3 involves three constraints relative to $(0,\ldots,0,1,1)$, $(0,\ldots,0,1,1,0)$ and $(0,\ldots,0,1,0,1)$ in GP_2, respectively. We show that all of the constraints involved can be satisfied simultaneously.

Each configuration in GP_1 involves a single constraint relative to the single configuration $(0,\ldots,0)$ in GP_0. To satisfy each constraint, we assign the configuration $0.5^{\eta-1}(1-q)$ as we did in the second assignment above. Hence all configurations in GP_1 have the *same* probability value, since all distributions of $\eta-1$ order have the same value $0.5^{\eta-1}$. Therefore, for each configuration $c \in GP_2$, even though it involves two constraints, each relative to a different configuration in GP_1, the assignment $P(c) = 0.5^{\eta-1}q$ satisfies both simultaneously.

Thus, by following this procedure, we can construct a jpd for $\mathcal{M}$ by alternating the assignment of $0.5^{\eta-1}q$ and $0.5^{\eta-1}(1-q)$ to configurations in successive groups. The resultant jpd clearly satisfies S1.

To show that the jpd also satisfies S2, we need to show, for an arbitrary pair X_i, X_j $(i \neq j)$ and $W = N \setminus \{X_i, X_j\}$, that $P(X_i|X_j,W) \neq P(X_i|W)$, or equivalently, $P(X_i,X_j,W) \neq P(X_i|W)P(X_j,W)$. Since $P(X_i|W) = 0.5$

and $P(X_j, W) = 0.5^{\eta-1}$ by S1, we have $P(X_i|W)P(X_j, W) = 0.5^{\eta}$. However, $P(X_i, X_j, W)$ has the value $0.5^{\eta-1}q$ or $0.5^{\eta-1}(1-q)$, where $q \neq 0.5$.

We have now constructed a jpd that satisfies both S1 and S2, and has a parameter q. Since q can take any value in the intervals $[0, 0.5)$ and $(0.5, 1]$, the theorem is proven. □

Consider the following example of a PI model. Suppose we have a digital gate with three inputs X_i $(i = 1, 2, 3)$ and an output X_4. The output $X_4 = 1$ whenever any two inputs are 0 and a third input is 1, or all inputs are 1. Suppose the three inputs are independent to each other and each of them has equal chance to be 0 or 1. Table 1 shows the jpd of these four variables. It can be easily verified that (1) the marginal distribution of each variable is 0.5, (2) any subset of two or three variables are mutually independent, and (3) the jpd is not $0.5^4 = 0.0625$.

(X_1, X_2, X_3, X_4)	$P(X_1, X_2, X_3, X_4)$	(X_1, X_2, X_3, X_4)	$P(X_1, X_2, X_3, X_4)$
$(0, 0, 0, 0)$	0.125	$(1, 0, 0, 0)$	0
$(0, 0, 0, 1)$	0	$(1, 0, 0, 1)$	0.125
$(0, 0, 1, 0)$	0	$(1, 0, 1, 0)$	0.125
$(0, 0, 1, 1)$	0.125	$(1, 0, 1, 1)$	0
$(0, 1, 0, 0)$	0	$(1, 1, 0, 0)$	0.125
$(0, 1, 0, 1)$	0.125	$(1, 1, 0, 1)$	0
$(0, 1, 1, 0)$	0.125	$(1, 1, 1, 0)$	0
$(0, 1, 1, 1)$	0	$(1, 1, 1, 1)$	0.125

Table 1: A PI model.

In the PI models constructed in the proof of Theorem 8, the marginal of each variable is equal to 0.5. However, PI models are not restricted to 0.5 marginals. Table 2 provides a jpd of three variables that have different marginals, in which (1) the marginals are $P(X_{1,0}) = 0.6$, $P(X_{2,0}) = 0.4$ and $P(X_{3,0}) = 0.2$, (2) any two variables are marginally independent, and (3) the jpd is not equal to the product $P(X_1)P(X_2)P(X_3)$.

(X_1, X_2, X_3)	$P(X_1, X_2, X_3)$	(X_1, X_2, X_3)	$P(X_1, X_2, X_3)$
$(0, 0, 0)$	0.024	$(1, 0, 0)$	0.056
$(0, 0, 1)$	0.216	$(1, 0, 1)$	0.104
$(0, 1, 0)$	0.096	$(1, 1, 0)$	0.024
$(0, 1, 1)$	0.264	$(1, 1, 1)$	0.216

Table 2: A PI model where variables have different marginals.

Among all the PMs, PI models represent one extreme. The other extreme is represented by models which display a totally different pattern of dependence relations. In the I-map of those models, no pair of variables connected by a link displays marginal independence. Between these two extremes, a whole

spectrum of PI models exist, in which variables are collectively dependent, marginally independent in some pairs and not marginally independent in other pairs. To classify these models, we shall refer to the models in Theorem 8 as *full* PI models and the models between the two extremes as *partial* PI models. Table 3 depicts such a partial PI model of three variables. The marginal for each variable is 0.5. Any pair of variables are dependent given the third. However, X_1 and X_2 are marginally independent ($P(X_1, X_2) = P(X_1)P(X_2)$), so are X_1 and X_3, but X_2 and X_3 are *not* marginally independent, namely, $P(X_2, X_3) \neq P(X_2)P(X_3)$.

(X_1, X_2, X_3)	$P(X_1, X_2, X_3)$	(X_1, X_2, X_3)	$P(X_1, X_2, X_3)$
$(0, 0, 0)$	0.225	$(1, 0, 0)$	0.20
$(0, 0, 1)$	0.025	$(1, 0, 1)$	0.05
$(0, 1, 0)$	0.025	$(1, 1, 0)$	0.05
$(0, 1, 1)$	0.225	$(1, 1, 1)$	0.20

Table 3: A partial PI model.

(X_1, X_2, X_3, X_4)	$P(X_1, X_2, X_3, X_4)$	(X_1, X_2, X_3, X_4)	$P(X_1, X_2, X_3, X_4)$
$(0, 0, 0, 0)$	0.0225	$(1, 0, 0, 0)$	0.02
$(0, 0, 0, 1)$	0.2025	$(1, 0, 0, 1)$	0.18
$(0, 0, 1, 0)$	0.005	$(1, 0, 1, 0)$	0.01
$(0, 0, 1, 1)$	0.02	$(1, 0, 1, 1)$	0.04
$(0, 1, 0, 0)$	0.0175	$(1, 1, 0, 0)$	0.035
$(0, 1, 0, 1)$	0.0075	$(1, 1, 0, 1)$	0.015
$(0, 1, 1, 0)$	0.135	$(1, 1, 1, 0)$	0.12
$(0, 1, 1, 1)$	0.09	$(1, 1, 1, 1)$	0.08

Table 4: An embedded PI model.

The PI models presented thus far are defined based on the entire domain of variables. In general, a PI model can be *embedded* as a submodel. Table 4 shows a PM with four variables X_i ($i = 1, 2, 3, 4$). It contains an embedded submodel identical to the partial PI model (of X_1, X_2 and X_3) given in Table 3. The marginal for each variable is 0.5 except $P(X_4 = 0) = 0.365$. The marginal for the subset $\{X_1, X_2, X_3\}$ is identical to that of Table 3, so the dependency relations among the three variables remain the same. But for variables X_2, X_3 and X_4, they are both collectively and pairwise dependent. The undirected minimal I-map of the PM has each pair of variables connected except X_1 and X_4.

Theorem 9 shows that the single-link lookahead search cannot learn the PI models.

Theorem 9 *Let $G_{\mathcal{M}}$ be the minimal I-map of a full pseudo-independent model $\mathcal{M}$ over a set X of η variables. Let G_0 be an initial chordal graph from which*

the learning starts and let the number of links of G_0 be $L \leq (\eta(\eta-1)/2) - 2$. Then $G_{\mathcal{M}}$ cannot be recovered by the single-link lookahead minimum entropy search.

Proof:

Since $\mathcal{M}$ is a full PI model, $G_{\mathcal{M}}$ is a complete graph and has $M = \eta(\eta-1)/2$ links. Let (G_0, P_0) be the initial DMN with $L \leq M-2$ links. Then G_0 cannot have two cliques of size $\eta-1$. Otherwise, G_0 will differ from a complete graph by a single link, i.e., $L = M-1$.

Since only a single link can be added each time and the resultant graph must be chordal, at each pass of the search, either a clique of size 2 is formed by joining two nodes in *disconnected* components or a clique of size $k > 2$ is formed by joining two cliques of size $k-1$ with their intersection of size $k-2$. In Figure 4, the clique $\{b, c\}$ $(k = 2)$ is formed by adding the dotted link (b, c), the clique $\{a, b, d\}$ $(k = 3)$ is formed by adding the dotted link (a, d), and the clique $\{d, e, f, g\}$ $(k = 4)$ is formed by adding the dotted link (d, g).

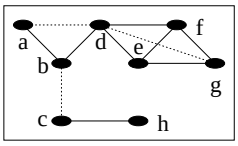

Figure 4: Clique formation by single-link addition.

Let (G_1, P_1) be a candidate DMN such that G_1 is augmented from G_0 by adding a single link (a, b). Denote the new clique formed by $W \cup \{a, b\}$. The entropy difference between the two DMNs is

$$H_1(N) - H_0(N) = H_{\mathcal{M}}(Wab) - (H_{\mathcal{M}}(Wa) + H_{\mathcal{M}}(Wb) - H_{\mathcal{M}}(W)).$$

According to S1 in Theorem 8, variables in any subset of size $\eta-1$ are pairwise marginally independent. Since the largest two cliques of an equal size in G_0 have a size $\eta-2$, we have $|Wab| \leq \eta-1$. Hence $Ind(Wa, \phi, b)$ and $Ind(W, \phi, b)$ hold in $\mathcal{M}$, which implies $H_1(N) - H_0(N) = 0$. Therefore no (G_1, P_1) will be selected and no link will be added to G_0. □

Although Theorem 9 involves learning only full PI models, the conclusion can be generalized to learning partial and embedded PI models as well. For example, if the single-link lookahead search is applied to the model in Table 3, it will only find the dependence between x_2 and x_3 and will output a structure with a single link. A two-link lookahead search after the single-link lookahead search will identify the collective dependence among the three variables.

The existence of PI models poses a challenge to learning probabilistic networks as approximate I-maps. Suppose we have no prior knowledge about the possible size of an embedded PI model. Then Theorem 8 dictates that search of potential cliques up to the size of the entire domain by multi-link lookahead is necessary in general. Since such a search is infeasible, prior knowledge

should be used for restricting the number of links required for the search. We will discuss this problem in more detail in Section 6.

We have shown that the single-link lookahead search combined with the entropy scoring metric is unable to learn PI models. In fact, the same conclusion can be drawn in learning probabilistic networks (including DMNs and BNs) with other scoring metrics. We now show this is indeed the case in some well-known algorithms.

Pseudo-independent models cannot be learned by Kutato [22]. The algorithm starts with an empty graph and uses a single-link lookahead search to learn a BN with an entropy scoring metric. Suppose the data-generating PM has a full PI submodel embedded (i.e., a subset of variables forms a full PI model). Since variables in the submodel are pairwise marginally independent, no link between any pair will decrease the entropy and hence no dependence can be discovered.

Likewise, PI models cannot be learned by the algorithm suggested by Lam and Bacchus [27], which uses the MDL principle to learn a BN. Let us first briefly describe their algorithm. It first computes the mutual information between each pair of nodes (corresponding to a link). It then places all links in a list in descending order of mutual information between the end nodes. The candidate BNs are generated by starting with an empty graph and including one link at a time from the beginning of the link list and down the list. It allocates equal amount of computational resources to explore candidate BNs of identical number of links (having the same complexity). After each complexity class has exhausted its resources, the best candidate BN according to the cross entropy scoring metric is chosen. The BN that has the minimal description length across classes will be the final output. If the data-generating model has a full PI submodel embedded, links between each pair of nodes in the submodel has zero mutual information. These links will be placed at the end of the link list and will be the last to be included in any candidate BNs. If these BNs are ever considered, the algorithm must have exhausted almost all possible BNs, which has an exponential complexity. Therefore, in practice, these BNs would have no chance to be tested and selected as the final output.

The previous two algorithms start with an empty graph. In contrast, the algorithm PC [39] learns a BN by starting from a complete graph. In the first pass, the algorithm removes each link if the end nodes of the link are *marginally* independent. In the second pass, it removes each link if the end nodes of the link are independent conditioned on a third node. In each of the following passes, it removes each link if the end points of the link are independent conditioned on a subset of nodes of higher order until a stopping condition is met. If the data-generating PM has a partial PI submodel embedded, then some pairs of nodes in the submodel are marginally independent. The links between each pair of them will be deleted in the first pass of the search. Therefore the collective dependence of the submodel will not be reflected in the final learned BN.

It can be shown that this limitation also applies to K2 [10] which uses a Bayesian scoring metric to learn a BN. A detailed discussion on this is beyond

the scope of this paper.

It is well known that *parity* functions cause failure of many decision tree learning algorithms (see [33, 25] for example). It should perhaps be emphasized here that PI models are a generalization of parity functions. The PI model in Table 1 is a parity function, but those in Tables 2, 3 and 4 are *not*.

6 A Multi-link Lookahead Learning Algorithm

The existence of pseudo-independent PMs and the inability of single-link lookahead search to learn such models suggest the adoption of more general learning algorithms when prior knowledge about the problem domain cannot eliminate the possible existence of such a model. In this section and the section to follow, we present one such algorithm and discuss related issues. As we are now moving from the theoretical analysis of the minimum entropy search to its practical implementation, we make some assumptions on the context in which the proposed algorithm is to be applied.

Assumption 1 *The database variables are discrete.*

We have assumed a discrete problem domain throughout the paper as indicated in the beginning of Section 2.2. This assumption simply restates it in terms of the feature of the database.

Assumption 2 *No cases in the database have missing variables.*

The above two assumptions are seen in most algorithms for learning probabilistic networks [10, 19]. To reduce the complexity of a multi-link lookahead search, we make the following sparseness assumption.

Assumption 3 *Let η be the size of the largest collectively dependent submodel in the problem domain. The higher the value of η, the less likely that a submodel of size η exists in the problem domain.*

A submodel of size η forms a clique of that size. Given the total number of variables in a problem domain, the larger a given clique is, the less number of alternative chordal graphs there are. Assumption 3 allows us to lookahead a small number of links such that we will not miss many embedded PI models. In the case where the number of variables involved in an embedded PI model is actually large, we probably will not be able to estimate its distribution reliably from the available data anyway. Even if the database is very large and such estimation is possible, the inference computation using such models will be very expensive, making them much less useful. In addition, if the set of evidence and query variables covers only a proper subset of the variables in an embedded PI model, then the posterior probability computed from an independent model will be identical to that from a PI model. Note that Assumption 3 does not differentiate between PI and non-PI submodels. Based on this assumption,

Algorithm 1

Input: A database D over a set N of variables, a maximum size η of clique, a maximum number $\kappa \leq \eta(\eta-1)/2$ of lookahead links, and a threshold δh.

begin

initialize an empty graph $G = (N, E)$;
$G' := G$;
for $i = 1$ to κ, do % search by levels
repeat % search by passes
initialize the entropy decrement $dh' := 0$;
for each set L of i links ($L \cap E = \phi$), do % search by steps
if $G^* = (N, E \cup L)$ is chordal and L is implied by a single clique of size $\leq \eta$, then
compute the entropy decrement dh locally;*
if $dh^* > dh'$, then $dh' := dh^*$, $G' := G^*$;
if $dh' > \delta h$, then $G := G'$,done := $false$; else done := $true$;
until done = true;
return G and the projected distribution P of the database on G;

end

the search is bounded in Algorithm 1 with two parameters κ and η specified by the user. The size of cliques is bounded by η. The size of PI submodels is bounded, by $\kappa \leq m(m-1)/2$, to $m \leq \eta$.

The search is structured into *levels* and each level is a search with the identical number of lookahead links. Each level consists of multiple *passes* and each pass is composed of multiple *steps*. Each pass at the same level tries to add the same number (i) of links. For instance, level one search adds a single link in each pass, level two search adds two links, and so on. Search at each pass selects i links after testing all distinct and legal combinations, one at each search step, of i links. The i links that decrease the entropy maximally are selected. If the corresponding entropy decrement is significant enough, the i links will be adopted and search continues at the same level. Otherwise the next higher level of search starts. Note that each intermediate graph is chordal as indicated by the *if* statement in the inner-most loop.

The entropy decrement dh^* is computed *locally* using a core as illustrated in Figure 5 with the JF representation. The subgraph F (corresponding to a core) contains two groups of cliques whose unions are $X \cup Z \cup A$ and $Y \cup Z \cup B$. After i links are added, the subgraph becomes F^* which contains two cliques $X \cup Z \cup A \cup B$ and $Y \cup Z \cup A \cup B$. Since the entropy contribution from the rest of the DMN does not change, dh^* can be computed using F and F^* only.

In Section 4, we showed that an I-map of a PM can be learned by the minimum entropy search when marginal distributions of cliques can be obtained *accurately*. This is equivalent to a database of infinite size, which contains only *true* dependencies. In Section 4.3, we classified two types (uncalled-for and redundant) of superfluous links that may be generated even when learning

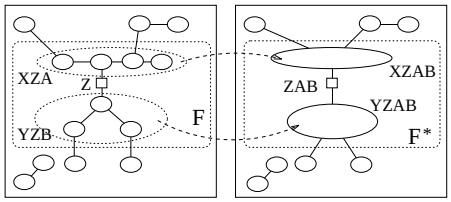

Figure 5: Illustration of local computation in Algorithm 1.

is performed using such databases. Hence, their generation is due to the use of a heuristic search. These links are undesirable because they unnecessarily increase the complexity of inference computation.

In practice, we must learn from a finite database. Such a database may contain *false* dependencies that do not exist in the underlying problem domain. They cause the generation of a third type of superfluous link which we refer to as *false* links. False links have a different undesirable effect *apart from* the complexity increase shared by the other two types of superfluous links. The probability values associated with false links tend to encode noise contained in the database. The encoded noise biases the jpd of the learned network and causes inference errors.

We have shown in Section 4.3 that the entropy metric has total resistance to the generation of uncalled-for links and partial resistance to the generation of redundant links. However, it has *no* resistance to the generation of false links at all. Without additional controls, the minimum entropy search tends to encode all false dependencies contained in the database. The threshold δh used in Algorithm 1 is aimed at reducing false links as well as redundant links. It works similarly as the encoding length of a model in a MDL approach to penalize a complex model, and also similarly as a prior biased towards a simpler structure in a Bayesian approach. It is supplied by the user for specifying to what extent (s)he is willing to trade complexity of the generated network with fitness of data. The necessity of such balance is discussed in [38, 27]. As the value of δh decreases, both the complexity of the network and the degree of fitness to data increase. As will be demonstrated in Section 7, the use of δh helps the learning algorithm to approximate its asymptotic behavior even though the size of the database is *far* from being 'infinite'.

We analyze the worst case time complexity of the algorithm. Testing the chordality of G^* can be performed in $O(|N|)$ time [16].

A JT can be computed by a maximal spanning tree algorithm [23]. A maximal spanning tree of a graph with v nodes and e links can be computed in $O((v+e)\log v)$ time [31]. Since a complete graph has $O(v^2)$ links, a maximal spanning tree can be computed in $O(v^2 \log v)$ time. Equivalently, computation of a JT of a chordal graph with k nodes and v cliques takes $O(v^2 \log v)$ time. Since $v \leq k$, computation of a JT of a chordal graph with k nodes takes $O(k^2 \log k)$ time. In computing dh^*, we need to compute F and F^* from the corresponding chordal subgraphs. Each of them contains no more than 2η variables, where η is the maximum allowable size of a clique. Therefore, we

can compute F and F^* in $O(\eta^2 \log \eta)$ time.

Let n be the number of cases in the database. We can extract the distribution P' on the 2η variables from the database directly in $O(n)$ time. The projected distribution on F and F^* can be computed by marginalizing P' to cliques and multipling clique distributions, which takes $O(\eta\ 2^\eta)$ time. The computation of dh^* from the projected distributions can be performed in $O(2^\eta)$ time. The complexity of each step is then $O(|N|+n+\eta\ (\eta \log \eta + 2^\eta))$. Since n is much larger than $|N|$, the complexity of each search step is $O(n + \eta\ (\eta \log \eta + 2^\eta))$.

The algorithm repeats for $O(\kappa)$ levels. Each level contains $O(|N|^2)$ passes. Each pass has $C(C(|N|, 2), \kappa) = O(|N|^{2\kappa})$ steps. Hence, the algorithm has $O(\kappa\ |N|^{2\kappa})$ search steps. The overall complexity of the algorithm is then $O(\kappa\ |N|^{2\kappa}\ (n + \eta\ (\eta \log \eta + 2^\eta)))$.

The computation is feasible if κ and η are small. This suggests the use of prior knowledge about the problem domain to further constrain the search. By exploring the prior knowledge of the problem domain, if we can partition the problem domain N into β equal subdomains and assert that there is no embedded PI models that crosses subdomain boundaries, then we can safely only perform the single-link lookahead search in the entire problem domain, but restrict the multi-link lookahead search to individual subdomains. We then have $O(|N|^2 + \kappa\ \frac{|N|^{2\kappa}}{\beta^{2\kappa}})$ search steps, which amounts to a complexity reduction of $\beta^{2\kappa}$ times. For example, suppose $|N| = 48$, $\kappa = 5$ and $\eta = 5$. The number of search steps is on the order of 3.2×10^{17}. If we can restrict the multi-link lookahead search to three subdomains of no more than 16 variables each, the number of search steps can be reduced to the order of 5.5×10^{12}.

Another useful heuristic is to apply single-link lookahead search first. If a disconnected network is generated and we have prior knowledge that it should be connected, then we can focus the multi-link lookahead search based on the resultant network. We leave such an investigation to future work. Related work on beam search can be found in Buntine [4].

7 Experimental Results

Algorithm 1 was implemented and a set of experiments were performed to (1) test if an I-map reasonably close to a control model can be learned given a reasonably large database generated from the control model; (2) provide a sense when the algorithm will behave significantly differently from the asymptotic behavior as the size of database becomes smaller; and (3) test if the multi-link lookahead is necessary and effective to learn an embedded PI submodel.

The algorithm was first run with the data on six probable risk factors for coronary thrombosis [37]. With $\kappa = 2$ and $\delta h = 0.004$, the DMN structure in Figure 6 was obtained. Our result is consistent with the models learned by other methods [12, 29].

We then tested the algorithm using the ALARM model [1] with 37 variables. A database of 30000 cases, generated from the BN, was used in the learning. A control DMN was obtained by converting the original BN with

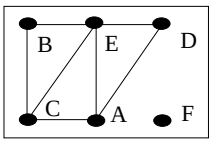

Figure 6: A DMN structure learned from a database of 1841 cases on coronary thrombosis. Variables are defined as follows: A, smoking; B, strenuous mental work; C, strenuous physical work; D, systolic blood pressure; E, ratio of β and α lipoproteins; F, family anamnesis of coronary heart disease.

the method used in a junction tree inference algorithm [24]. In Figure 7, the learned DMN (with $\kappa = 1$ and $\delta h = 0.003$) is compared with such a control DMN. The control DMN has 68 links out of which 60 links (solid) are contained in the learned DMN. The learned DMN contains 6 additional links (dotted) and 8 missing links (dashed). It was found that 7 missing links in the control DMN are due to a single weak link from node 24 to node 6. In the original BN, node 6 has parents 5, 7, 36 and 24. During conversion, links among these parents were added (including missing links $(5, 24)$, $(7, 24)$ and $(36, 24)$). During triangulation, missing links $(9, 24)$, $(26, 24)$ and $(9, 11)$ were filled in. Since the dependence between node 24 and node 6 is very weak (out of 18 combinations of the other three parents of node 6, in only four combinations the value of node 24 affects the distribution of node 6 significantly), the algorithm did not learn the link $(6, 24)$ and consequently missed all the other six links as well.

This result is comparable with the published results on learning ALARM as a BN (with 46 (directed) arcs). For example, Kutato [22] learned ALARM with two missing arcs and two additional arcs.[6] K2 [10] had one arc missing and one arc added. The result by Heckerman et al. [20] had two missing arcs and one reversed arc. A database of 10000 cases were used in all three cases. A total variable ordering consistent with the control model was supplied to Kutato and K2 in addition to the data, and a prior network structure was supplied in [20].

As our formal analysis is performed under the condition of accurate estimation of clique marginals, our next experiment provides a sense when the algorithm will practically deviate from its asymptotic behavior. We used a submodel of ALARM with 17 variables (including nodes 0,...,5, 13,...,23) with the size of the joint probability space $2^6 \times 3^{11} \approx 10^7$. Four databases of 10000, 5000, 2000 and 1000 cases were generated from the submodel BN. Using the database of 10000 cases, the DMN structure in Figure 8 (left) was obtained with $\kappa = 1$ and $\delta h = 0.002$, which contains two addition links (dotted) and no missing links. The identical result was obtained with the databases of 5000 and 2000 cases with δh being 0.003 and 0.006, respectively. The value of δh used was increased as the size of database was decreased in order to exclude super-

[6]Note that when an arc from node Y to node X is added in a BN, several links will be added in the corresponding DMN to complete the parent set of X and to render the structure chordal. A similar effect occurs when an arc is missing.

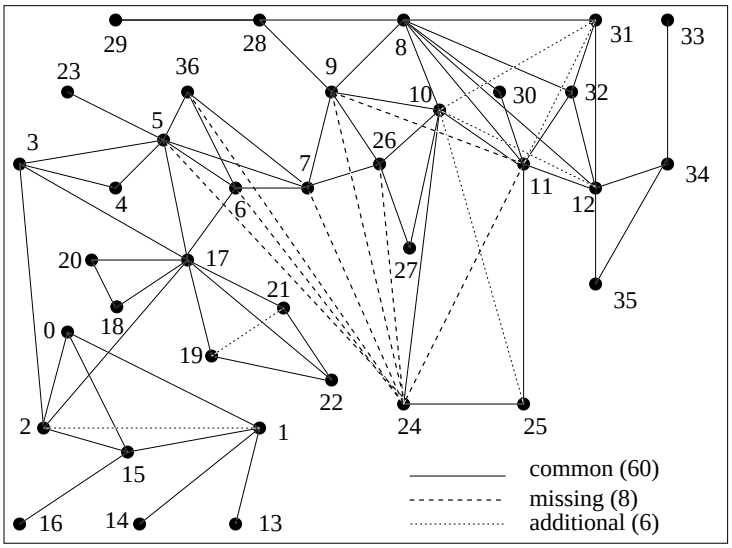

Figure 7: Comparison of the control DMN converted from ALARM BN with the learned DMN.

fluous links. As the size of database was further reduced to 1000, the structure in Figure 8 (middle) was obtained with $\delta h = 0.02$. Four links ($(5, 23)$, $(2, 15)$, $(0, 15)$ and $(1, 2)$) were missing relative to the structure in the left. However, further reducing δh to 0.01 (right), we obtained the two additional links $(2, 19)$ and $(2, 21)$ before any of the missing links were learned. Therefore, we can conclude that for this model, as the size of databases drops below 1000 cases, the algorithm no longer behaves approximately to its asymptotic behavior. This experiment demonstrates how δh can be used to help the learning algorithm to approximate its asymptotic behavior even though the size of the database is *far* from being 'infinite'.

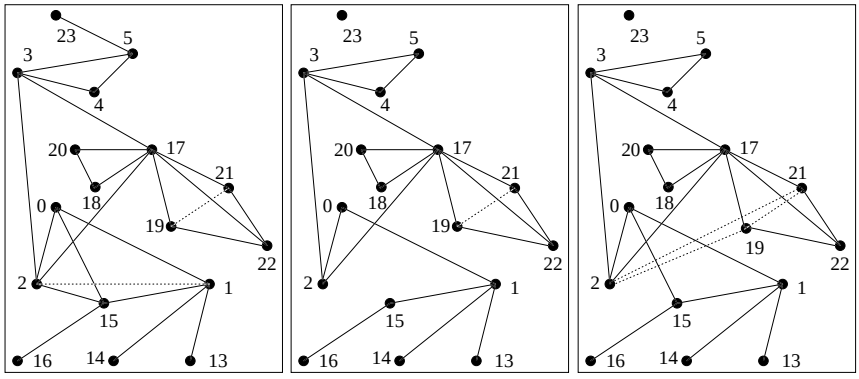

Figure 8: DMN structures learned from databases generated by a submodal of ALARM. Left: The databases have sizes from 10000 to 2000. Middle and right: The database has a size 1000 and δh is 0.02 and 0.01, respectively.

The last experiment demonstrates the effect of using multi-link lookahead

search in learning PI submodels. We randomly generates a BN such that a PI submodel is embedded. The generator randomly selected the number of parents and number of children of each node subject to the corresponding upper bounds specified by the user. Once the structure was generated, conditional probability distributions of each node was then randomly generated. The generator can also embed a PI submodel of up to 5 variables. The distribution of the PI submodel was created as in the proof of Theorem 8. Figure 9 (a) shows a generated BN and (b) is its converted DMN. All variables are binary. Nodes 5, 6 and 9 form an embedded PI submodel. A database of 500 cases was generated. With $\kappa = 1$ and $\delta h = 0.03$, the structure in (c) was learned. With $\kappa = 2$, the structure in (d) was learned. Using smaller δh with either κ value, additional links were obtained before the missing links. Clearly, single and double-link lookahead failed to discover the PI submodel. With $\kappa = 3$ and $\delta h = 0.03$, the structure in (b) was learned. Note that even though an embedded PI submodel was used, the experimental result is also applicable to the case of a 'nearly' PI submodel. This is because there is no difference between a database generated by a PI model and the one generated by a 'nearly' PI model as long as the database is small.

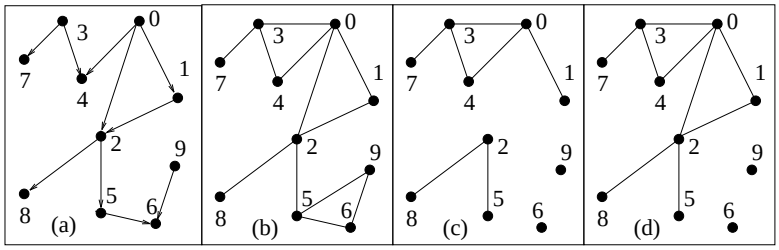

Figure 9: (a) The structure of a control BN. (b) Control DMN converted from (a). (c) The structure learned with single-link lookahead. (d) The structure learned with double-link lookahead.

8 Discussion

In this paper, we studied learning a decomposable Markov network from a database using the entropy scoring metric and a heuristic search. Our analysis has revealed the 'microscopic' mechanism of a minimum entropy search and its asymptotic behavior. We showed that the process to decrease the entropy parallels the process to remove false independence relations in the intermediate networks. The decreasing entropy drives the search forward until an I-map of the domain model is learned when the size of the database is large.

The understanding of this mechanism uncovers that the I-map of a probabilistic model cannot be fully recovered unless some false independence relations (equivalently, a true dependence not yet encoded) can be identified at each search pass. We showed that there exists a class of pseudo-independent models whose dependences can only be detected with a lookahead of multiple

links. These models form a generalization of the well known parity functions. As a single-link lookahead search has been adopted in many learning algorithms for efficiency reasons, our analysis indicates that results obtained by these methods will be compromised if the problem domain contains pseudo-independent submodels.

To uncover the PI models, we have proposed an algorithm that uses the multi-link lookahead search. Although we have suggested some simple ways in which prior knowledge can be applied to reduce the complexity of the multi-link lookahead search, clearly more research is needed in this direction.

Acknowledgement

We thank David Poole for a valuable discussion, anonymous reviewers and M. Hoyle for helpful comments on the earlier draft. We also thank T. Chu, J. Hu and C. C. Liu for the assistance in the implementation and experiment. This work is supported by Research Grant OGP0000982 and OGP0155425 from the Natural Sciences and Engineering Research Council (NSERC). The authors are members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, NSERC, and the participation of PRECARN Associates Inc.

References

[1] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. Technical Report KSL-88-84, Knowledge Systems Lab, Medical Computer Science, Stanford University, 1989.

[2] R.R. Bouckaert. Properties of Bayesian belief network learning algorithms. In R. Lopez de Mantaras and D. Poole, editors, *Proc. of 10th Conf. on Uncertainty in Artificial Intelligence*, pages 102–109, Seattle, Washington, 1994. Morgan Kaufmann.

[3] W. Buntine. Classifiers: a theoretical and empirical study. In R. Lopez de Mantaras and D. Poole, editors, *Proc. of 1991 Inter. Joint Conf. on Artificial Intelligence*, pages 638–644, Sydney, 1991.

[4] W. Buntine. Theory refinement on Bayesian networks. In B. D. D'Ambrosio, P. Smets, and P. P. Bonissone, editors, *Proc. of 7th Conf. on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

[5] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, (2):159–225, 1994.

[6] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[7] P. Cheeseman. Overview of model selection. In *Proc. of 4th Inter. Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, 1993. Society for AI and Statistics.

[8] D. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: serach methods and experimental results. In *Proc. of 5th Conf. on Artificial Intelligence and Statistics*, pages 112–128, Ft. Lauderdale, 1995. Society for AI and Statistics.

[9] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, (14):462–467, 1968.

[10] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, (9):309–347, 1992.

[11] A.P. Dawid and S.L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21(3):1272–1317, 1993.

[12] D. Edwards and T. Havranek. A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72(2):339–351, 1985.

[13] M. Frydenberg and S.L. Lauritzen. Decomposition of maximum likelihood in mixed graphical interaction models. *Biometrika*, 76(3):539–555, 1989.

[14] R.M. Fung and S.L. Crawford. Constructor: A system for the induction of probabilistic models. In *Proc. of AAAI*, pages 762–769, Boston, MA, 1990. MIT Press.

[15] R.G. Gallager. *Information Theory and Reliable Communication*. John Wiley and Sons, 1968.

[16] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[17] P.J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[18] P. Hajek, T. Hovranek, and R. Jirousek. *Uncertain Information Processing in Expert Systems*. CRC Press, 1992.

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

[20] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[21] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J.F. Lemmer and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 149–163. Elsevier Science Publishers, 1988.

[22] E.H. Herskovits and G.F. Cooper. Kutato: an entropy-driven system for construction of probabilistic expert systems from database. In *Proc. 6th Conf. on Uncertainty in Artificial Intelligence*, pages 54–62, Cambridge,, 1990.

[23] F.V. Jensen. Junction tree and decomposable hypergraphs. Technical report, JUDEX, Aalborg, Denmark, February 1988.

[24] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4):269–282, 1990.

[25] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proc. 11th Inter. Conf. on Machine Learning*, pages 121–129, 1994.

[26] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[27] W. Lam and F. Bacchus. Learning Bayesian networks: an approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293, 1994.

[28] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, (50):157–244, 1988.

[29] D. Madigan and A.E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of American Statistical Association*, 89(428):1535–1546, 1994.

[30] D. Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.

[31] U. Manber. *Introduction to Algorithms: a Creative Approach*. Addison-Wesley, 1989.

[32] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons, 1990.

[33] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, (5):71–99, 1990.

[34] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, (29):241–288, 1986.

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

[36] G. Rebane and J. Pearl. The recovery of causal ploy-trees from statistical data. In *Proc. of Workshop on Uncertainty in Artificial Intelligence*, pages 222–228, Seattle, 1987.

[37] Z. Reinis, J. Pokorny, V. Basika, J. Tiserova, K. Gorican, D. Horakova, E. Stuchlikova, T. Havranek, and F. Hrabovsky. Prognostic significance of the risk profile in the prevention of coronary heart disease. *Bratis. lek Listy*, 76:137–150, 1981.

[38] S.L. Sclove. Small-sample and large-sample statistical model selection criteria. In P. Cheeseman and R.W. Oldford, editors, *Selecting Models from Data*, pages 31–39. Springer-Verlag, 1994.

[39] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–73, 1991.

[40] S.K.M. Wong, C.J. Butz, and Y. Xiang. A method for implementing a probabilistic model as a relational database. In *Proc. 11th Conf. on Uncertainty in Artificial Intelligence*, pages 556–564, Montreal, 1995.

[41] S.K.M. Wong and Y. Xiang. Construction of a Markov network from data for probabilistic inference. In *Proc. 3rd Inter. Workshop on Rough Sets and Soft Computing*, pages 562–569, San Jose, 1994.

[42] S.K.M. Wong, Y. Xiang, and X. Nie. Representation of Bayesian networks as relational databases. In *Proc. 5th Inter. Conf. Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU)*, pages 159–165, Paris, 1994.

[43] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, to appear in fall, 1996.

[44] Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned Bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.

[45] Y. Xiang, D. Poole, and M. P. Beddoes. Multiply sectioned Bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9(2):171–220, 1993.