Learning Belief Networks in Pseudo-independent Domains

Yang Xiang

Technical Report CS-96-07
September, 1996

Department of Computer Science
University of Regina
Regina, Saskatchewan, CANADA
S4S 0A2

ISSN 0828-3494
ISBN 0-7731-0331-7

Learning Belief Networks in Pseudo-independent Domains

Y. Xiang
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2, yxiang@cs.uregina.ca

Abstract

A *pseudo-independent* (PI) model is a probabilistic domain model (PDM) where proper subsets of a set of collectively dependent variables display marginal independence. It has been shown that several commonly used algorithms for learning belief networks will fail when the data generating model is a PI model. A better understanding of these models will facilitate the design of better learning algorithms to copy with them. We propose formal definitions of the whole spectrum of discrete PI models. We present a parameterization of PI models which leads to a better understanding of the mechanism that forms PI models. We then show that the well-known parity problems in machine learning is a degeneration of PI models. Therefore, PI models form a more general class of challenging problems for learning from data.

Using an abstraction of a class of algorithms for learning belief networks, we show that PI models form the dividing line between success and failure of an algorithm using some form of conditional independence test and a single link search. We proposed several strategies to manage the computational complexity when a single link search is extended into a multi-link search. The effectiveness of these strategies are demonstrated with experimental results.

Topics: Computational theory of learning, heuristic search, belief networks, probabilistic reasoning, probabilistic dependence models.

1 Introduction

Belief networks (Bayesian networks, Markov networks and influence diagrams) [22, 20, 11] are becoming widely applied to AI tasks where representing and reasoning with uncertain knowledge are essential [2, 5, 7, 10, 19]. As an alternative and supplement to encoding probabilistic knowledge from domain experts, and driven by the data overload in the society, learning belief networks from data is being actively studied by many [9, 14, 24, 3, 28, 1, 4, 18, 8]. The learned network is commonly used as a compact probabilistic model capable of answering probabilistic queries effectively, although researchers are also exploring learning a belief network as a causal model [25]. The former type of learning is the focus of this paper. The task in such learning is to take as input a dataset generated by an underlying probabilistic domain model (PDM), and to recover one or more sparse belief network(s) that approximate(s) the underlying PDM.

A *pseudo-independent* (PI) model is a PDM where proper subsets of a set of collectively dependent variables display marginal independence (hence pseudo-independent). Commonly used algorithms for learning belief networks from data rely on single link search to identify local dependence among variables [9, 14, 24, 3, 28]. It has been shown [30] that these algorithms will fail when the domain model is a PI model. However, only a very restricted class of PI models (full binary PI models, defined below, with uniform variable marginals) was formalized. A better understanding of these models will facilitate the design of better learning algorithms to copy with them.

In this paper, we propose formal definitions of the whole spectrum (*full*, *partial* and *embedded*, defined below) of discrete PI models. We present a parameterization of PI models which leads to a better understanding of the mechanism that forms PI models. We show how a full binary PI model is composed of a small set of meaningful parameters. We then show that the well-known parity problems in machine learning is a degeneration of the full binary PI models. Therefore, PI models form a more general class of challenging problems for learning belief networks from data.

The pseudo-independence property of PI models requires more complex search procedures in learning. In order to manage the computational complexity, we derive the relation between PI models and the success of learning algorithms. This leads to a halting condition under which we are sure that the learned model is correct and no further search is necessary. We also propose control strategies in learning to direct the search effort towards the most beneficial direction.

2 Probabilistic dependence

The concept of conditional independence is well known (see for example, [22]). In this section, we distinguish several notions of probabilistic dependence (particularly the collective and general dependence) that are less commonly used but are essential to the understanding of PI models.

Let N be a set of discrete variables in a problem domain. Each variable is associated with a finite domain, a set of possible values that it can take. We shall denote the possible values by consecutive integers 0, 1, 2, ... A *configuration* or a *tuple* of $N' \subseteq N$ is an assignment of values to every variable in N', e.g.,

$$(X_1 = 0, X_2 = 1, \ldots)$$

which we shall denote by

$$(x_{1,0}, x_{2,1}, \ldots).$$

A *probabilistic domain model* (PDM) over N determines the probability of every tuple of N' for each $N' \subseteq N$.

For three disjoint sets A, B and C of variables, A and B are *conditionally independent* given C if

$$P(A|B, C) = P(A|C) \quad whenever \; P(B, C) > 0.$$

When $C = \phi$, A and B are *marginally independent.* If each variable X in a subset A is marginally independent of $A \setminus \{X\}$, then

$$P(A) = \prod_{X \in A} P(X).$$

We shall say that variables in A are marginally independent.

A pair of variables X and Y are *pairwise dependent* if

$$P(X|Y) \neq P(X).$$

Pairwise dependence is the opposite of marginal independence between the pair.

A set N of variables are *collectively dependent* if for each proper subset $A \subset N$, there exists no proper subset $C \subset N \setminus A$ such that

$$P(A|N \setminus A) = P(A|C).$$

Collective dependence does not eliminate possible marginal independence *within* proper subsets, as will be seen in Section 3.

A set N of variables are *generally dependent* if for any proper subset A,

$$P(A|N \setminus A) \neq P(A).$$

Like collective dependence, general dependence does not eliminate possible marginal independence within proper subsets. Furthermore, collective dependence is not required. Namely, for some proper subset A, there can be a proper subset $C \subset N \setminus A$ such that

$$P(A|N \setminus A) = P(A|C).$$

General dependence is weaker dependence than collective dependence. Both may coexist with either conditional independence or marginal independence within proper subsets. General dependence is the opposite of marginal independence between a proper subset and the rest of domain variables.

3 What is a PI model?

PI models can be classified into three types. The most restrictive type is the *full* PI models.

Definition 1 (Full PI model) *A PDM over a set N ($|N| \geq 3$) of variables is a* `full` *PI model if the following two conditions hold:*

(S1) *For each $X \in N$, variables in $N \setminus \{X\}$ are marginally independent.*

(S2) *Variables in N are collectively dependent.*

Table 1 shows the joint probability distribution (jpd) of a binary full PI model. It can be easily verified that the marginals of variables are

$$P(x_{1,0}) = 0.7, \; P(x_{2,0}) = 0.6, \; P(x_{3,0}) = 0.35, \; P(x_{4,0}) = 0.45.$$

Any subset of three variables are marginally independent, e.g.,

$$P(x_{1,1}, x_{2,0}, x_{3,1}) = 0.0451 + 0.0719 = 0.117$$

and

$$P(x_{1,1})\ P(x_{2,0})\ P(x_{3,1}) = 0.3 * 0.6 * 0.65 = 0.117.$$

The four variables are collectively dependent, e.g.,

$$P(x_{1,1}|x_{2,0}, x_{3,1}, x_{4,0}) = 0.0451/0.1755 = 0.257$$

and

$$P(x_{1,1}|x_{2,0}, x_{3,1}) = P(x_{1,1}|x_{2,0}, x_{4,0}) = P(x_{1,1}|x_{3,0}, x_{4,0}) = 0.3.$$

(X_1, X_2, X_3, X_4)	$P(.)$	(X_1, X_2, X_3, X_4)	$P(.)$
$(0, 0, 0, 0)$	0.0586	$(1, 0, 0, 0)$	0.0359
$(0, 0, 0, 1)$	0.0884	$(1, 0, 0, 1)$	0.0271
$(0, 0, 1, 0)$	0.1304	$(1, 0, 1, 0)$	0.0451
$(0, 0, 1, 1)$	0.1426	$(1, 0, 1, 1)$	0.0719
$(0, 1, 0, 0)$	0.0517	$(1, 1, 0, 0)$	0.0113
$(0, 1, 0, 1)$	0.0463	$(1, 1, 0, 1)$	0.0307
$(0, 1, 1, 0)$	0.0743	$(1, 1, 1, 0)$	0.0427
$(0, 1, 1, 1)$	0.1077	$(1, 1, 1, 1)$	0.0353

Table 1: A full PI model.

In a full PI model, every proper subset of N displays marginal independence. This is relaxed in the *partial* PI models.

Definition 2 (Partial PI model) *A PDM over a set N ($|N| \geq 3$) of variables is a* `partial` *PI model if the following three conditions hold:*

(S1') *There exists a partition $\{N_1, \ldots, N_k\}$ ($k \geq 2$) of N such that variables in each subset N_i is generally dependent, and for each $X \in N_i$ and each $Y \in N_j$ ($i \neq j$), X and Y are marginally independent.*

(S2) *Variables in N are collectively dependent.*

(X_1, X_2, X_3)	$P(.)$	(X_1, X_2, X_3)	$P(.)$	(X_1, X_2, X_3)	$P(.)$
$(0, 0, 0)$	0.05	$(1, 0, 0)$	0.05	$(2, 0, 0)$	0.10
$(0, 0, 1)$	0.04	$(1, 0, 1)$	0.01	$(2, 0, 1)$	0.05
$(0, 1, 0)$	0.01	$(1, 1, 0)$	0	$(2, 1, 0)$	0.09
$(0, 1, 1)$	0.11	$(1, 1, 1)$	0.08	$(2, 1, 1)$	0.11
$(0, 2, 0)$	0.06	$(1, 2, 0)$	0.03	$(2, 2, 0)$	0.01
$(0, 2, 1)$	0.03	$(1, 2, 1)$	0.03	$(2, 2, 1)$	0.14

Table 2: A partial PI model.

Table 2 shows the jpd of a partial PI model over two trinary variables and one binary variable. The marginals are

$$P(x_{1,0}) = 0.3,\ \ P(x_{1,1}) = 0.2,\ \ P(x_{1,2}) = 0.5,$$

$$P(x_{2,0}) = 0.3,\ \ P(x_{2,1}) = 0.4,\ \ P(x_{2,2}) = 0.3,$$

$$P(x_{3,0}) = 0.4,\ \ P(x_{3,1}) = 0.6.$$

The partition is $\{\{X_1\}, \{X_2, X_3\}\}$. X_1 is marginally independent of each variable in the other subset, e.g.,

$$P(x_{1,1}, x_{2,0}) = P(x_{1,1})\ P(x_{2,0}) = 0.06.$$

However the variables in the other subset are pairwise dependent, e.g.,

$$P(x_{2,0}, x_{3,1}) = 0.1 \neq P(x_{2,0})\ P(x_{3,1}) = 0.18.$$

The three variables are collectively dependent, e.g.,

$$P(x_{1,1}|x_{2,0}, x_{3,1}) = 0.1$$

and

$$P(x_{1,1}|x_{2,0}) = P(x_{1,1}|x_{3,1}) = 0.2.$$

Similarly,

$$P(x_{2,1}|x_{1,0}, x_{3,1}) = 0.61,\ \ P(x_{2,1}|x_{1,0}) = 0.4,\ \ P(x_{2,1}|x_{3,1}) = 0.5.$$

Note that S1' requires marginal independence of each pair of variables from distinct subsets of the partition. Without this restriction, a PDM over variables X, Y and Z, with X and Y being marginally independent causes of Z, will qualify as a partial PI model. However, such models do not cause problems for common learning algorithms using single link search [9, 14, 24, 3, 28].

Proposition 3 establishs the relation between the two types of PI models defined so far.

Proposition 3 *A full PI model is a partial PI model.*

Proof:

S1 implies S1'. $\square$

The converse is not true in general since variables in some subset of a partial PI model may be pairwise dependent as in Table 2.

A partial PI model involves the entire set N of domain variables. An embedded PI submodel displays the same dependence pattern but involves only a proper subset of domain variables.

Definition 4 (Embedded PI submodel) *Let a PDM be over a set N of generally dependent variables. A proper subset $N' \subset N$ ($|N'| \geq 3$) of variables form an* `embedded` *PI submodel if the following two conditions hold:*

(S4) *N' forms a partial PI model.*

(S5) *The partition $\{N_1, \ldots, N_k\}$ of N' by S1' extends into N. That is, there is a partition $\{A_1, \ldots, A_k\}$ of N such that $N_i \subseteq A_i$, ($i = 1, .., k$), and for each $X \in A_i$ and each $Y \in A_j$ ($i \neq j$), X and Y are marginally independent.*

Definition 4 requires that variables in N are generally dependent. It eliminates the possibility that a proper subset is marginally independent of the rest of N.

Table 3 shows the jpd of PDM with an embedded PI model over three variables X_1, X_2 and X_3. The marginals of the five variables are

$$P(x_{1,0}) = 0.3,\ P(x_{2,0}) = 0.6,\ P(x_{3,0}) = 0.3,\ P(x_{4,0}) = 0.34,\ P(x_{5,0}) = 0.59.$$

$(X_1,..,X_5)$	$P(.)$	$(X_1,..,X_5)$	$P(.)$	$(X_1,..,X_5)$	$P(.)$	$(X_1,..,X_5)$	$P(.)$
$(0,0,0,0,0)$	0	$(0,1,0,0,0)$	.0018	$(1,0,0,0,0)$	.0080	$(1,1,0,0,0)$	.0004
$(0,0,0,0,1)$	0	$(0,1,0,0,1)$	.0162	$(1,0,0,0,1)$	.0720	$(1,1,0,0,1)$	.0036
$(0,0,0,1,0)$	0	$(0,1,0,1,0)$	.0072	$(1,0,0,1,0)$	.0120	$(1,1,0,1,0)$	.0006
$(0,0,0,1,1)$	0	$(0,1,0,1,1)$	.0648	$(1,0,0,1,1)$	.1080	$(1,1,0,1,1)$	.0054
$(0,0,1,0,0)$	.0288	$(0,1,1,0,0)$	.0048	$(1,0,1,0,0)$	.0704	$(1,1,1,0,0)$	.0864
$(0,0,1,0,1)$	.0072	$(0,1,1,0,1)$	.0012	$(1,0,1,0,1)$	.0176	$(1,1,1,0,1)$	.0216
$(0,0,1,1,0)$	.1152	$(0,1,1,1,0)$	.0192	$(1,0,1,1,0)$	.1056	$(1,1,1,1,0)$	.1296
$(0,0,1,1,1)$	.0288	$(0,1,1,1,1)$	.0048	$(1,0,1,1,1)$	.0264	$(1,1,1,1,1)$	.0324

Table 3: A PDM containing an embedded PI model.

Within the embedded PI model, the partition consists of subsets $A = \{X_1\}$ and $B = \{X_2, X_3\}$. Outside the PI submodel, A extends to include X_4 and B extends to include X_5. Each variable in one subset is marginally independent of each variable in the other subset, e.g.,

$$P(x_{1,1}, x_{5,0}) = P(x_{1,1})\ P(x_{5,0}) = 0.413.$$

Variables in the same subset are pairwise dependent, e.g.,

$$P(x_{2,1}, x_{3,0}) = 0.1 \neq P(x_{2,1})\ P(x_{3,0}) = 0.12.$$

The three variables in the submodel are collectively dependent, e.g.,

$$P(x_{1,1}|x_{2,0}, x_{3,1}) = 0.55$$

and

$$P(x_{1,1}|x_{2,0}) = P(x_{1,1}|x_{3,1}) = 0.7.$$

However, X_4 is independent of other variables given X_1, and X_5 is independent of other variables given X_3, e.g.,

$$P(x_{5,1}|x_{2,0}, x_{3,0}, x_{4,0}) = P(x_{5,1}|x_{3,0}) = 0.9.$$

4 Graphical representation of PI models

4.1 Terminologies

We review terminologies for graphical representation of PDMs used in the rest of the paper. Readers are referred to [22, 17, 4, 27, 1] for more in-depth discussion.

We denote a graph by $G = (N, E)$ where N is a set of nodes and E is a set of links connecting nodes in N. In this paper, we consider undirected graphs only. A graph is *complete* if each pair of nodes is connected by a link. A graph is *empty* if $E = \phi$. A graph is *connected* if there is a path between every pair of nodes. Otherwise, the graph is *disconnected*. A graph is *chordal* if every cycle of length > 3 has a link connecting two nonadjacent nodes in the cycle. The *adjacency set* or *boundary* of a node is the set of all nodes that are adjacent to it. A *clique* of a graph is a maximal set of nodes that is complete. A *component* of a graph is a maximal subgraph that is connected. For disjoint subsets A, B and C of nodes in a graph, we use $< A|C|B >$ to denote that nodes in C intercept all paths between A and B.

A *junction tree* (JT) T of a connected chordal graph 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. Without confusion, we shall refer to a node in a JT as a clique and the intersection of a pair of adjacent cliques as a *sepset*. A connected graph has a JT iff (if and only if) the graph is chordal [6]. A *junction forest* (JF) F of a chordal graph G is a set of JTs each of which is a JT of one component of G.

Graphical representations of PDMs (Bayesian and Markov networks) play an important role in probabilistic inference in artificial intelligence systems [22, 16, 12, 29]. A graph G is an *I-map* [22] of a PDM over N if there is an one-to-one correspondence between nodes of G and variables in N such that for all disjoint subsets A, B and C of N, $< A|C|B >$ implies A and B are conditionally independent given C. G is a *minimal* I-map of a PDM if no link in G can be removed such that the resultant graph is still an I-map.

In this paper, we shall restrict to decomposable Markov networks (DMNs). They are closely related to Bayesian networks but are simpler to study for the purpose of this paper [1]. A DMN is an ordered pair (G, P) where G is a

[1]See [15], for example, for how testing d-separation [22] in a Bayesian network B can

chordal graph and P is a probability distribution

$$P(N) = (\prod_C P(C))/(\prod_S P(S)),$$

where C is a clique in a JF of G and S is a sepset in the JF. We refers to G as the *structure* of the DMN and P as its distribution. We will use a DMN as the target representation and learning outcome of an unknown PDM. As the DMN will be an approximation of the PDM, we do not require the structure of a DMN to be a minimal I-map of a PDM as required in some literature (e.g., in [22]).

4.2 Colored I-maps

Since variables in a PI submodel are collectively dependent, in a minimal I-map [22] of the PDM, the submodel is complete. The marginal independence between subsets in the submodel is thus unrepresented. We extend the minimal I-maps into *colored* I-maps. The marginal independence between subsets are highlighted in a colored I-map by coloring the corresponding links.

Definition 5 *An undirected graph G is a* `colored I-map` *of a PDM M over N if (1) G is a minimal I-map of M, and (2) for each PI submodel m, links between each pair of nodes from distinct marginally independent subsets in m are colored.*

Figure 1 shows the colored I-maps for PI models in Table 1 through 3. Marginally independent subsets are connected by grey links.

A PDM may contain more than one embedded PI submodels. When this is the case, N is partitioned into $\{A_1, A_2, \ldots\}$ such that each pair of variables from distinct A_i's are marginally independent. Consider the following example:

Three balls are drawn each from a different urn. Urn 1 has 20% white balls and the rest of balls are black. Urn 2 and urn 3 have 60% and 50% of white balls, respectively. A music box plays if all three balls are white or exactly one is white. A dog barks if two random lights are both on or both off. John complains if it's too quiet (neither the box plays nor the dog barks) or too noisy (both the box plays and the dog barks).

be performed in a simpler way in an undirected graph converted from B.

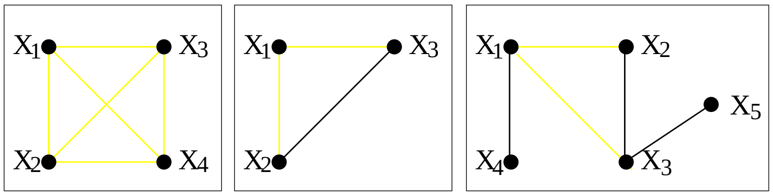
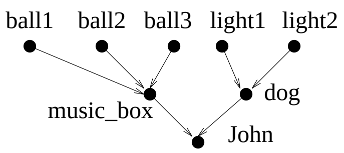


Figure 1: The colored I-maps for PI models in Table 1 (left), 2 (middle) and 3 (right).

The eight variables form $2^8 = 256$ tuples. To simplify the presentation, we specify the PDM as a Bayesian network in Figure 2. Abbreviations w, b, p, q, b, c are used for *white*, *black*, *play*, *quiet*, *bark*, *complain*, respectively.



dog ε {bark,quiet}
music_box ε {play,quiet}
John ε{complain,satisfied}

ball ε {white,black}
light ε {on,off}

P(light1=on)=0.5
P(light2=on)=0.7

P(ball1=w)=0.2
P(ball2=w)=0.6
P(ball3=w)=0.5

P(John=c|music_box=p,dog=b)=1
P(John=c|music_box=q,dog=b)=0
P(John=c|music_box=p,dog=q)=0
P(John=c|music_box=q,dog=q)=1

P(dog=b|light1=on,light2=on)=1
P(dog=b|light1=on,light2=off)=0
P(dog=b|light1=off,light2=on)=0
P(dog=b|light1=off,light2=off)=1

P(music_box=p|all balls=w)=1
P(music_box=p|one ball=w)=1
P(music_box=p|all balls=b)=0
P(music_box=p|one ball=b)=0

Figure 2: Specification of the music-box-dog-John model.

The domain is partitioned into six marginally independent subsets. Figure 3 shows the colored I-map.

The PDM contains five embedded PI submodels over

$$N_1 = \{ball1, ball3, music\_box\}, \;\; N_2 = \{ball2, ball3, music\_box\},$$

$$N_3 = \{ball1, ball2, ball3, music\_box\}, \;\; N_4 = \{light1, light2, dog\},$$

$$N_5 = \{music\_box, dog, John\}.$$

Note that the first two PI submodels are embedded in the third PI submodel whose distribution is shown in Table 4.

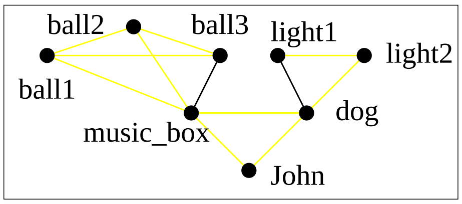


Figure 3: The colored I-map for the music-box-dog-John example.

$(ball1, ball2, ball3, music\_box)$	$P(.)$	$(ball1, ball2, ball3, music\_box)$	$P(.)$
(w, w, w, p)	0.06	(b, w, w, p)	0
(w, w, w, q)	0	(b, w, w, q)	0.24
(w, w, b, p)	0	(b, w, b, p)	0.24
(w, w, b, q)	0.06	(b, w, b, q)	0
(w, b, w, p)	0	(b, b, w, p)	0.16
(w, b, w, q)	0.04	(b, b, w, q)	0
(w, b, b, p)	0.04	(b, b, b, p)	0
(w, b, b, q)	0	(b, b, b, q)	0.16

Table 4: A PI submodel in the music-box-dog-John model.

To see that N_1 forms a PI submodel, first we have

$$P(ball1 = w|ball3) = P(ball1 = w|music\_box) = P(ball1 = w) = 0.2.$$

Namely, $ball1$ is marginally independent of $ball3$ and $music\_box$. We also have

$$P(ball1 = w|ball3 = w, music\_box = p) = 0.2727 \neq P(ball1 = w|music\_box = p) = 0.2,$$

$$P(ball1 = w|music\_box = p, ball3 = w) = 0.2727 \neq P(ball1 = w|ball3 = w) = 0.2,$$

$$P(ball3 = w|music\_box = p, ball1 = w) = 0.6 \neq P(ball3 = w|ball1 = w) = 0.5.$$

Namely, the three variables are collectively dependent.

This example illustrates that in general, a PI model can contain one or more PI submodels, and this embedding can occur recursively for any finite number of times.

The example also illustrates that the partition of a PI model into marginally independent subsets is not unique. The above PI submodel over N_3 can be partitioned into $\{\{ball1\}, \{ball2\}, \{ball3, music\_box\}\}$. It can also be partitioned into $\{\{ball1\}, \{ball2, ball3, music\_box\}\}$. In either case, variables in each subset is generally dependent as required by S1'.

5 Why bother learning PI models?

Given a PDM that contains an embedded PI submodel and a dataset generated from the domain, what is the consequence if our learning algorithm fails to recognize the collective dependence within the embedded submodel?

Consider the PI model given in Table 3. Suppose a learning algorithm considers the subsets $\{X_1, X_4\}$ and $\{X_2, X_3, X_5\}$ to be marginally independent. If the learned model is used in inference and we want to know the probability of $X_1 = 1$ given the observation $X_2 = 0$ and $X_3 = 0$, the model will tell us

$$P(x_{1,1}|x_{2,0}, x_{3,0}) = P(x_{1,1}) = 0.7.$$

However, the collective dependence within the embedded PI submodel determines

$$P(x_{1,1}|x_{2,0}, x_{3,0}) = 1.$$

This difference in the posteriors may warrant totally different actions.

It has been shown [30] that several commonly used algorithms (based on the entropy, the minimum description length, the Bayesian and the conditional independence metrics) for learning belief networks fail to learn PI models. The common reason is that they rely on detection of local dependence using a single link search. In order to design better learning algorithms to copy with PI models, an in-depth understanding of these models is necessary.

6 Parameterization of PI models

A general discrete probability distribution has $k^n - 1$ independent parameters (values) where n is the number of variables and k is the number of possible values each variable can take. A PI model is more constrained and has less parameters. We study how a PI model is composed of these parameters.

Such an understanding can provide hints for how to learn these models, can provide a direct method to simulate these models which can then be used to test our learning algorithms, and can guide us in determining what problem domains may or may not contain a PI model.

Since the number of parameters of a PDM that contains an embedded PI submodel may be calculated from the number of parameters of the PI submodel and the number of parameters in the rest of the model, we focus on partial PI models. We present a result on full PI models and then discuss its implication on partial PI models.

Theorem 6 *The total number of parameters of a full PI model is*

$$W = W_1 + W_2.$$

The number W_1 is the count of marginal parameters (marginals),

$$W_1 = \sum_{i=1}^{n} (D_i - 1),$$

where n is the total number of variables and $D_i \geq 2$ is the number of values that the ith variable can take. The number W_2 is the count of joint probability parameters (joints),

$$W_2 = 1 + \sum_{i=1}^{n} \sum_{j=1}^{C(n,i)} \prod_{k=1}^{i} (D_k - 2),$$

where j ranges over each combination taking i variables out of n each time, k ranges over each variable in a given combination, and D_k is the number of values that the variable can take.

Proof:

In a full PI model, each variable may have a different marginal distribution, which implies W_1.

To derive W_2, we assume that the W_1 marginals have been specified and we then determine the value for each joint. We group the joints according to the number of variables taking non-zero values. For example, the group GP_0 contains a single joint

$$P(x_{1,0}, \ldots, x_{n,0}),$$

and the group GP_1 contains joints

$$P(x_{1,l_1}, x_{2,0}, \ldots, x_{n,0}),\ P(x_{1,0}, x_{2,l_2}, x_{3,0}, \ldots, x_{n,0}),\ \ldots$$

where $l > 0$. We determine the values of joints group by group in ascending order of the group index.

The single joint of GP_0 is not uniquely determined by the W_1 marginals. We can specify this joint subject to the constraint

$$P(x_{1,0}, \ldots, x_{n,0}) \neq \prod_{i=1}^{n} P(x_{i,0})$$

due to S2. This gives the first term, 1, in the formula for W_2.

For each joint in GP_1, only a single variable may take non-zero values. This leads to $C(n,1) = n$ choices. Once a variable X_k with D_k possible values is selected, there are $D_k - 1$ joints associated with the variable in GP_1 that need to be determined. The first $D_k - 2$ such joints are not uniquely determined by the W_1 marginals, $P(x_{1,0}, \ldots, x_{n,0})$, S1 and S2. Nor are they uniquely determined by the joints in GP_1 associated with other variables through S1 and S2. Once we have specified the first $D_k - 2$ joints, however, the $D_k - 1$'th joint is uniquely determined by S1:

$$\sum_{j=0}^{D_k-1} P(x_{1,0}, \ldots, x_{k,j}, \ldots, x_{n,0}) = \prod_{i=1, i\neq k}^{n} P(x_{i,0}).$$

Hence the contribution of GP_1 to W_2 is

$$\sum_{j=1}^{C(n,1)} (D_k - 2) = \sum_{j=1}^{C(n,1)} \prod_{k=1}^{1} (D_k - 2).$$

Next, we consider GP_i $(i > 1)$ where each joint has i variables with non-zero values. There are $C(n,i)$ ways to choose the i variables. Suppose a combination is chosen as $X_1, \ldots, X_i$. Denote the set of associated joints in GP_i by J. For each joint in J, each X_k $(k \leq i)$ may take any value from $\{1, \ldots, D_k - 1\}$, and hence

$$|J| = \prod_{k=1}^{i} (D_k - 1).$$

If we restrict the value range to $\{1, \ldots, D_k - 2\}$, there are exactly

$$\prod_{k=1}^{i}(D_k - 2)$$

distinct such joints in J. Denote this subset of joints by J'. None of the joint in J' can be uniquely determined by the other joints in J' plus the W_1 marginals, the joints in $GP_0, \ldots, GP_{i-1}$, S1 and S2. However, once we have specified joints in J', each joint in $J \setminus J'$ is uniquely determined through S1, e.g.,

$$\sum_{j=0}^{D_i-1} P(x_{1,l_1}, \ldots, x_{i-1,l_{i-1}}, x_{i,j}, x_{i+1,0}, \ldots, x_{n,0}) = (\prod_{k=1}^{i-1} P(x_{k,l_k}))(\prod_{k=i+1}^{n} P(x_{k,0})),$$

where $l > 0$. Hence the contribution of GP_i to W_2 is

$$\sum_{j=1}^{C(n,i)} \prod_{k=1}^{i}(D_k - 2).$$

The formula for W_2 now follows. □

For a full PI model over four trinary variables,

$$W_1 = 8, \ W_2 = 16, \ W = 24.$$

If one of the variables is binary, then

$$W_1 = 7, \ W_2 = 8, \ W = 15.$$

A binary full PI model of n variables has a very small number of parameters. Since $W_1 = n$ and $W_2 = 1$, we have $W = n + 1$. It differs from a marginally independent model of the same number of variables by just one parameter.

A general partial PI model will have more parameters than a full PI model with the same set of variables. This is because a proper subset of variables in a general partial PI model may be dependent on each other. Additional parameters are needed to determine exactly how they are dependent on each other within the subset.

Note that although W parameters can be *non-uniquely* specified in a full PI model, they *cannot* be specified *independently*, but rather must follow S1 and S2. Propositions 7, 8 and their combination Theorem 9 show how S1 and S2 constrain the W parameters in the binary case.

Proposition 7 *The jpd of a full PI model over $n \geq 3$ binary variables has the following form:*

$$P(x_{1,0},..,x_{n,0}) = P(x_{n,0}|x_{1,0},..,x_{n-1,0}) \prod_{i=1}^{n-1} P(x_{i,0}), \tag{1}$$

$$P(x_{l_1,1},..,x_{l_w,1},x_{l_{w+1},0},..,x_{l_n,0}) =$$

$$\frac{\prod_{i=1}^{n} P(x_{i,0})}{\prod_{i=1}^{w} P(x_{l_i,0})} \sum_{i=1}^{w} (-1)^{i+1} (\prod_{j=1}^{w-i} P(x_{l_j,1}))(\prod_{k=w-i+2}^{w} P(x_{l_k,0})) + (-1)^w P(x_{1,0},..,x_{n,0}) \tag{2}$$

where $w = 1,..,n$,

$$P(x_{n,0}) = \max_i P(X_i), \tag{3}$$

$$P(x_{n,0}|x_{1,0},..,x_{n-1,0}) \neq P(x_{n,0}), \tag{4}$$

and the largest w ($w \geq 2$) probabilities $P(X_{l_i})$ ($i = 1,..,w$) satisfy

$$\sum_{i=1}^{w} (-1)^{i+1} (\prod_{j=1}^{w-i} (1 - P(X_{l_j})))(\prod_{k=w-i+2}^{w} P(X_{l_k})) + (-1)^w \frac{P(x_{n,0}|x_{1,0},..,x_{n-1,0})}{P(x_{n,0})} \prod_{i=1}^{w} P(X_{l_i}) \geq 0. \tag{5}$$

Proof:

Since we are interested in PDMs, we assume that all marginals are in $(0,1)$ instead of [0,1]. To simplify the notation, we denote

$$P(x_{1,0},..,x_{i-1,0},x_{i,1},x_{i+1,0},..,x_{j-1,0},x_{j+1,0},..,x_{n,0})$$

by $R(i|j)$ where the vertical bar is not to be confused with the conditioning bar in $P(X_i|X_j)$. That is, the tuple has $X_i = 1$, X_j marginalized out, and $X_k = 0$ for all $k \neq i,j$. We denote $P(x_{1,0},..,x_{n,0})$ by $R(.)$ and denote $P(x_{1,0},..,x_{j-1,0},x_{j+1,0},..,x_{n,0})$ by $R(|j)$. We group $P(X_1,..,X_n)$ into $GP_0,..,GP_n$ where GP_i contains probabilities of tuples in which exactly i variables take value 1.

Suppose the PDM is a full PI model, then S1 and S2 hold. S1 implies that for each $n-1$-tuple, its probability is the product of the corresponding marginals. That is, the PDM satisfies n such constraints one for each subset of $n-1$ variables.

Let p denote $P(x_{n,0}|x_{1,0},..,x_{n-1,0}) \in [0,1]$. Due to S1, we have

$$R(.) = R(|n)\ p = P(x_{1,0})..P(x_{n-1,0})\ p.$$

Due to S2, we have

$$R(.) = P(x_{1,0})..P(x_{n-1,0})\ p \neq P(x_{1,0})..P(x_{n,0}).$$

Hence we have equation (1) and condition (4), and the single probability $R(.)$ in GP_0 is derived.

Next we consider the probabilities in GP_w ($1 \leq w \leq n$). Without losing generality, we derive $R(1,..,w)$.

$$\begin{aligned}
& R(1,..,w) \\
= \ & R(1,..,w-1|w) - R(1,..,w-1) \\
= \ & R(1,..,w-1|w) - R(1,..,w-2|w-1) + R(1,..,w-2) \\
= \ & \sum_{i=1}^{w}(-1)^{i+1}\ R(1,..,w-i|w-i+1) + (-1)^w\ R(.) \\
= \ & \frac{\prod_{i=1}^{n} P(x_{i,0})}{\prod_{i=1}^{w} P(x_{i,0})} \sum_{i=1}^{w}(-1)^{i+1}(\prod_{j=1}^{w-i} P(x_{j,1}))(\prod_{k=w-i+2}^{w} P(x_{k,0})) + (-1)^w P(x_{1,0},..,x_{n,0})
\end{aligned}$$

This is equation (2). Next, we use induction to derive the condition under which $R(1,..,w)$ is a valid probability. We consider first $R(j)$ in GP_1. We have

$$R(j) = (\prod_{i=1}^{n} P(x_{i,0}))/P(x_{j,0}) - (\prod_{i=1}^{n-1} P(x_{i,0}))\ p = (\prod_{i=1}^{n-1} P(x_{i,0}))\ (P(x_{n,0}) - P(x_{j,0})\ p)/P(x_{j,0}).$$

It is a valid probability iff

$$P(x_{n,0}) \geq P(x_{j,0})\ p.$$

This is always satisfied if

$$P(x_{n,0}) = \max_i P(X_i)$$

which is condition (3).

Without losing generality, we then consider $R(1, j)$ $(1 < j \leq n)$:

$$R(1,j) = \frac{\prod_{i=1}^{n} P(x_{i,0})}{P(x_{1,0})\ P(x_{j,0})}\ (P(x_{1,1}) - P(x_{j,0}) + \frac{P(x_{1,0})\ P(x_{j,0})\ p}{P(x_{n,0})}).$$

For it to be a valid probability, we must have

$$P(x_{1,1}) - P(x_{j,0}) + \frac{P(x_{1,0})\ P(x_{j,0})\ p}{P(x_{n,0})} \geq 0.$$

We shall view the left-hand side as a function $f(P(x_{1,0}))$ of $P(x_{1,0})$ with other parameters held constant. Note that

$$P(x_{i,0}) = 1 - P(x_{i,1}).$$

The first order derivative of $f(P(x_{1,0}))$ is

$$f'(P(x_{1,0})) = -1 + P(x_{j,0})\ p/P(x_{n,0}).$$

We have

$$f'(P(x_{1,0})) \leq 0\ \textit{since}\ P(x_{n,0}) \geq P(x_{j,0})\ p.$$

This implies that $f(P(x_{1,0}))$ is non-increasing with $P(x_{1,0})$, i.e., if $f(P(x_{1,0})) \geq 0$ then for any $\alpha < P(x_{1,0})$, we have $f(\alpha) \geq 0$. Since $P(x_{1,0}), P(x_{j,0})$ and all marginals are symmetric, we conclude that if

$$1 - P(X_i) - P(X_j) + \frac{P(X_i)\ P(X_j)\ p}{P(x_{n,0})} \geq 0$$

holds for the largest marginals $P(X_i)$ and $P(X_j)$, then each value in GP_2 defined by equation (2) is a valid probability. We have thus verified condition (5) for $w = 2$.

We now make the inductive assumption that each value in GP_{w-1} $(w > 2)$ is a valid probability if condition (5) holds. We show that each value in GP_w is a valid probability if condition (5) holds. Without losing generality, we consider $R(1, .., w)$. For $R(1, .., w)$ to be a valid probability, it must be the case

$$\sum_{i=1}^{w}(-1)^{i+1}(\prod_{j=1}^{w-i} P(x_{j,1}))(\prod_{k=w-i+2}^{w} P(x_{k,0})) + (-1)^{w}\frac{p}{P(x_{n,0})}\ \prod_{i=1}^{w} P(x_{i,0}) \geq 0.$$

We shall view the left-hand side as a function $f(P(x_{1,0}))$ of $P(x_{1,0})$ with other parameters held constant. The first order derivative of $f(P(x_{1,0}))$ is

$$f'(P(x_{1,0})) = \sum_{i=1}^{w-1}(-1)^i(\prod_{j=2}^{w-i} P(x_{j,1}))(\prod_{k=w-i+2}^{w} P(x_{k,0}))+(-1)^w\frac{p}{P(x_{n,0})}\prod_{i=2}^{w} P(x_{i,0}).$$

From the inductive assumption on GP_{w-1}, we know

$$-[\sum_{i=1}^{w-1}(-1)^i(\prod_{j=2}^{w-i} P(x_{j,1}))(\prod_{k=w-i+2}^{w} P(x_{k,0})) + (-1)^w\frac{p}{P(x_{n,0})}\prod_{i=2}^{w} P(x_{i,0})] \geq 0.$$

Thus $f'(P(x_{1,0}) \leq 0$, i.e., $f(P(x_{1,0}))$ is non-increasing with $P(x_{1,0})$. This implies that if $f(P(x_{1,0})) \geq 0$, then for any $\alpha < P(x_{1,0})$, we have $f(\alpha) \geq 0$. Since $P(x_{1,0}), .., P(x_{w,0})$ and all marginals are symmetric, we conclude that if condition (5) holds for the largest w marginals, then each value in GP_w is a valid probability.

The proposition is proven. □

Proposition 8 *A PDM over $n \geq 3$ binary variables is a full PI model if its jpd satisfies equations (1) and (2) subject to conditions (3) through (5).*

Proof: Since equation (1) and condition (4) hold, S2 is true. To show S1 holds, it suffices to show

$$R(.) + R(j) = \prod_{i=1}^{j-1} P(x_{i,0}) \prod_{i=j+1}^{n} P(x_{i,0})$$

and

$$R(1, .., w) + R(1, .., w+1) = \prod_{i=1}^{w} P(x_{i,1}) \prod_{i=w+2}^{n} P(x_{i,0}).$$

From equations (1) and (2) with $w = 1$, we obtain

$$R(.) + R(j) = (\prod_{i=1}^{n} P(x_{i,0}))/P(x_{j,0}).$$

From equation (2), we obtain

$$\begin{aligned}
& R(1,..,w) + R(1,..,w+1) \\
= \; & \frac{\prod_{i=1}^{n} P(x_{i,0})}{\prod_{i=1}^{w} P(x_{i,0})} \sum_{i=1}^{w} (-1)^{i+1} (\prod_{j=1}^{w-i} P(x_{j,1}))(\prod_{k=w-i+2}^{w} P(x_{k,0})) + \\
& \frac{\prod_{i=1}^{n} P(x_{i,0})}{\prod_{i=1}^{w+1} P(x_{i,0})} \sum_{i=1}^{w+1} (-1)^{i+1} (\prod_{j=1}^{w-i+1} P(x_{j,1}))(\prod_{k=w-i+3}^{w+1} P(x_{k,0})) \\
= \; & \frac{\prod_{i=1}^{n} P(x_{i,0})}{\prod_{i=1}^{w+1} P(x_{i,0})} [\sum_{i=1}^{w} (-1)^{i+1} (\prod_{j=1}^{w-i} P(x_{j,1}))(\prod_{k=w-i+2}^{w+1} P(x_{k,0})) + \\
& \sum_{i=1}^{w+1} (-1)^{i+1} (\prod_{j=1}^{w-i+1} P(x_{j,1}))(\prod_{k=w-i+3}^{w+1} P(x_{k,0}))] \\
= \; & \frac{\prod_{i=1}^{n} P(x_{i,0})}{\prod_{i=1}^{w+1} P(x_{i,0})} \prod_{j=1}^{w} P(x_{j,1}).
\end{aligned}$$

□

Combining Propositions 7 and 8, we obtain the following theorem.

Theorem 9 *A PDM over $n \geq 3$ binary variables is a full PI model iff its jpd satisfies equations (1) and (2) subject to conditions (3) through (5).*

Equations (1) and (2) describe how each joint probability is composed of the $n+1$ parameters $P(x_{i,0})$ $(i = 1, \ldots, n)$ and $P(x_{n,0}|x_{1,0}, \ldots, x_{n-1,0})$. Conditions (3) through (5) describe the constraints that they must observe.

Theorem 9 shows how easy (or how difficult) it is to form a binary full PI model. Suppose all marginals of a truly marginally independent PDM are in the right range dictated by conditions (3) and (5). To turn this PDM into a PI model, all it takes is to change one conditional probability according to condition (4). This analysis shows that suspicion against the existence of PI models in practice (and their challenge to our learning algorithms) is unjustified. We discuss below some special PI models which provide further evidence for the existence of PI models in practice.

7 Special PI models

7.1 Parity problems

It is well known that parity problems cause failure for many machine learning algorithms, see for example [21, 13, 26]. A parity problem can be described as follows: A set of marginally independent input variables $\{X_1, .., X_{n-1}\}$ each take the value 0 or 1 with an equal chance. An output variable X_n takes 0 or 1 such that the total number of 1's is even (for even parity).

The following proposition shows that parity problems form a degeneration of equations (1) and (2) and thus form a special case of binary full PI models.

Proposition 10 *A parity problem over n variables is a full PI model with $P(X_i) = 0.5$ $(i = 1, .., n)$. For even parity,*

$$P(x_{n,0}|x_{1,0}, .., x_{n-1,0}) = 1,$$

and for odd parity,

$$P(x_{n,0}|x_{1,0}, .., x_{n-1,0}) = 0.$$

Proof: It suffices to show the even parity case due to the symmetry. From the problem description, clearly

$$P(X_i) = 0.5 \quad (i = 1, .., n-1)$$

and

$$P(x_{n,0}|x_{1,0}, .., x_{n-1,0}) = 1.$$

We first show $P(X_n) = 0.5$. For the set of $n-1$ input variables, there are 2^{n-1} $n-1$-tuples each occurring with probability 0.5^{n-1}. Exactly half of them contain an even number of 0's. Denote this group of 2^{n-2} $n-1$-tuples by G_e and the other group of 2^{n-2} $n-1$-tuples by G_o.

Each $n-1$-tuple extends into two n-tuples when X_n is added. If a $n-1$-tuple is in G_e, the n-tuple with $X_n = 0$ has non-zero probability. If a $n-1$-tuple is in G_o, the n-tuple with $X_n = 0$ has zero probability. Hence we have $P(x_{n,0}) = 2^{n-2} * 0.5^{n-1} = 0.5$, the probability by which a $n-1$-tuple belongs to G_e.

Next, we show that conditions (3) through (5) are satisfied. Conditions (3) and (4) are trivially true. Condition (5) becomes

$$\sum_{i=1}^{w}(-1)^{i+1}0.5^{w-1} + (-1)^{w}0.5^{w-1} = 0.5^{w-1}[\sum_{i=1}^{w}(-1)^{i+1} + (-1)^{w}].$$

When w is even, the sum is 0.5^{w-1}. When w is odd, the sum is 0. □

Note that the distinction between a unique output variable X_n and the other input variables in a parity problem is unnecessary. Once the n variables behave according to a particular parity, we would not be able to tell which one is the output variable since each variable has the same marginals and each may assume the role of the output.

7.2 Modulus addition problems

Recently, it was shown [26] that parity problems are special cases of modulus addition problems. The latter display similar properties of parity problems and cause failure of ID3-like algorithms. A modulus addition problem can be described as follows: A problem domain consists of a set of marginally independent and uniformly distributed input variables $\{X_1, .., X_{n-1}\}$ and an output variable X_n. Each X_i $(i = 1, ..., n)$ has the domain $\{0, 1, \ldots, D_i - 1\}$ where $D_i \geq 2$ such that for each $i < n$,

$$D_i = k_i \ D_n,$$

where k_i is a positive integer. X_n is the sum of $X_1, .., X_{n-1}$ modulo D_n.

Proposition 11 shows that modulus addition problems are also special cases of full PI models.

Proposition 11 *A modulus addition problem is a full PI model.*

Proof:

It suffices to show that S1 and S2 holds. That S2 holds is trivially true. To show that S1 holds, we need only to show

$$P(X_n|X_1, .., X_{n-2}) = P(X_n).$$

First,

$$P(X_n|X_1,..,X_{n-2}) = \sum_{X_{n-1}} P(X_nX_{n-1}|X_1,..,X_{n-2})$$

$$= \sum_{X_{n-1}} P(X_n|X_1,..,X_{n-1})P(X_{n-1}|X_1,..,X_{n-2})$$

$$= \sum_{X_{n-1}} P(X_n|X_1,..,X_{n-1})P(X_{n-1}) = (1/D_{n-1}) \sum_{X_{n-1}} P(X_n|X_1,..,X_{n-1})$$

Since X_n is determined given $X_1,..,X_{n-1}$, $P(X_n|X_1,..,X_{n-1})$ is either 0 or 1. Given the values of $X_1,..,X_{n-2}$, as X_{n-1} takes values $0,..,D_{n-1}-1$ in sequence, X_n will go through each value in its domain exactly k_{n-1} times. Hence the summation in the above equation equals to k_{n-1} and we obtain

$$P(X_n|X_1,..,X_{n-2}) = k_{n-1}/D_{n-1} = 1/D_n.$$

Due to the symmetry of inputs, the above equation holds for any $n-2$ inputs. That is, the conditioning is irrelevant (removable), which implies

$$P(X_n|X_1,..,X_{n-2}) = P(X_n) = 1/D_n.$$

□

7.3 Comparison of two learning paradigms

It is known that parity problems [21] and modulus addition problems [26] cause failure of traditional machine learning algorithms that exploit probabilistic dependence. It is also known that their superclass, PI models, causes failure of several common algorithms for learning belief networks [30]. It is interesting to compare the two failures.

In traditional machine learning, the objective is to extract from a dataset a compact set of general rules (including decision trees) for determining the value of a target variable (e.g. class). Probabilistic dependence is used to guide many learning algorithms. As shown in [26], when the problem is the parity or modulus addition and the dataset contains all possible tuples, ID3 fails to generalize and a decision tree with one leaf node per tuple is returned. We argue that this is not a failure of the learning algorithm but a failure of the target representation. There exists no more compact decision tree, other than the one returned by ID3, that can represent the concept better. The solution of the failure lies on better representations.

The objective of learning belief networks is to extract a close-to-minimal I-map that captures dependence between domain variables. As shown in [30], when the PDM is a PI model, several common algorithms will fail to recognize the collective dependence between variables in a PI submodel. However, this is *not* a failure of the belief network representation, but rather a failure of the single link search procedure used in the learning algorithm. On the other hand, failing to learn a more compact set of rules than the dataset [26] is a matter of efficiency and generality. But treating a subset of collectively dependent variables as truly marginally independent is a matter of correctness. Therefore we can and must improve our learning algorithm to handle such PDMs better. This is addressed in Section 9.

8 PI models and single link search

We consider learning an I-map of a PDM from a dataset. The PDM may or may not be a PI model. We assume that we have no prior knowledge whether it is a PI model.

Suppose an algorithm LIM for `Learning I-Maps` is equipped with a test whether $P(A|B,C) = P(A|C)$ holds for three disjoint subsets of variables A, B and C. Clearly, if we allow such test to be performed for arbitrary A, B and C, then LIM will be able to learn an I-map of any PDM. Unfortunately, the complexity of LIM will be exponential. We therefore restrict LIM such that the test is only performed based on the current learned graph in the following manner:

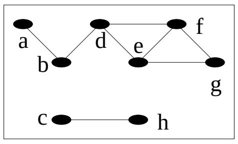


Figure 4: A chordal graph for illustration of single link search by LIM.

LIM starts with an empty graph G (with no links). It systematically selects a link (X, Y) not contained in G such that either X and Y are contained in different components of G or their minimal cutset C is complete. In

Figure 4, the missing link (b, c) belongs to the first case, and (a, d) and (d, g) belong to the second case. In the first case, LIM tests if $P(X|Y) = P(X)$. In the second case, LIM tests if $P(X|Y, C) = P(X|C)$. If the test is negative, then the link is added to the current graph. LIM repeats the above until no link can be added.

LIM can be viewed as a generalization of several commonly used algorithms for learning belief network structures [9, 14, 24, 3, 28, 8]. The algorithm PC [24] uses just the above test, except that PC starts with a complete graph and removes a link when the test is positive. A similar test was used by Rebane and Pearl [23]. It has been shown [31] that the cross entropy based test is equivalent to the above test.

Commonly used learning algorithms are usually able to differentiate between a strong dependence from a weak one in the dataset such that a link corresponding to a weak true dependence or a false dependence due to sampling or other sources may be rejected. These algorithms commonly select the link to add that corresponds to the strongest dependence among alternatives (greedy search) such that the learned structure is as close to the minimal I-map or as sparse as possible. We have chosen to abstract these capabilities out from LIM. Namely, LIM cannot detect a strong dependence from a weak one, cannot reject any noise, nor does LIM try to minimize the links added. It on average will not learn an I-map that is close to minimal and it may sometime learn a trivial I-map (complete graph). These are not our concern here. The important features left in LIM are its single-link search and use of a restricted independence test. Theorem 12 therefore shows clearly that PI models define a dividing line between success and failure of a class of learning algorithms that use a single link search.

Theorem 12 *LIM can learn an I-map of a PDM M iff M is not a PI model.*

Proof:

We prove the necessity by contraposition. Assume that M is a PI model. According to definitions of PI models, the domain variables can be partitioned into marginally independent subsets. Consider two such subsets A and B. Since LIM starts with an empty graph, A and B are initially disconnected. Since the test "$P(X|Y) = P(X)$?" will succeed for each $X \in A$ and each $Y \in B$, A and B will never be connected by LIM. Hence LIM cannot return an I-map of M.

Next, we prove the sufficiency by induction on the number of variables in M. Assume that M over N is not a PI model. For the base case, let $N = \{X, Y, Z\}$, i.e., $|N| = 3$. A PI model over N will be either a full PI model satisfying

$$P(X|Y) = P(X), \quad P(X|Z) = P(X), \quad P(Y|Z) = P(Y),$$

or a partial PI model satisfying, say,

$$P(X|Y) = P(X), \quad P(X|Z) = P(X), \quad P(Y|Z) \neq P(Y).$$

Since M is not a PI model, we will have at least, say,

$$P(X|Z) \neq P(X), \quad P(Y|Z) \neq P(Y).$$

If the minimal I-map contains only links (X, Z) and (Y, Z), then LIM will return it. Otherwise, the minimal I-map must be complete. In this case, LIM wll first learn (X, Z) and (Y, Z), then find

$$P(X|Y, Z) \neq P(X|Z),$$

and finally add (X, Y).

We now assume that for all $|N| = n \geq 3$, if M is not a PI model, then LIM will return an I-map. Suppose $|N| = n + 1$. Let the minimal I-map of M be G_M.

Suppose G_M is not complete. Then by Dirac's lemma (see [6]), there exists nonadjacent X_1 and X_2 each with a complete boundary in G_M. Let M_i be the PDM obtained from M by marginalizing X_i out ($i = 1, 2$). Since M is not a PI model, M_i ($i = 1, 2$) cannot be a PI model. Hence LIM can return an I-map of M_i by assumption. Since learning links in G_M with X_1 as one endpoint does not involve X_2, all such links will be added by LIM when learning M. From the symmetry of M_i, we conclude that LIM can return an I-map of M.

Now suppose G_M is complete. For any X_i ($i = 1, 2$), let M_i be the PDM obtained from M by marginalizing X_i out ($i = 1, 2$). Since M is not PI, M_i cannot be a PI model. Note that since G_M is complete, M_i is also complete. Since LIM can learn the I-map for M_i ($i = 1, 2$) by assumption, it can learn G_M with only one link (X_1, X_2) missing. Afterwards, it will add the link since

$$P(X_1|X_2, N \setminus \{X_1, X_2\}) \neq P(X_1|N \setminus \{X_1, X_2\}).$$

$\square$

9 Managing complexity in learning PI models

The single link search used in LIM is used by several common algorithms in learning belief networks [9, 14, 24, 3, 28, 8]. The precise reason is the search efficiency: only $O(|N|^2)$ links need to be tested in a systematic search. Theorem 12 shows that we must increase the sophistication of learning algorithms if we do not want to miss PI models. A PI submodel over N' is partitioned into k marginally independent subsets. If we try multiple links at each search step such that the submodel is complete by the new links, and test

$$\text{``}P(X|Y, N' \setminus \{X, Y\}) = P(X|, N' \setminus \{X, Y\})?''$$

where (X, Y) is one of the new links, we will get a negative result which prompts the completion of N'. We shall call such a strategy a *multi-link search*.

We shall refer to a systematic search that examines one link, two links, ..., up to $i > 1$ links at each step as an i-link search. An i-link search is more expensive: $O(|N|^{2i})$ sets of links need to be tested. As the complexity increases exponentially with the number i, an i-link search must be performed cautiously. We propose three strategies in the following three subsections to manage the computational complexity.

9.1 Performing single link search first

The potential of a single link search should be exploited fully before an i-link search is attempted. The objective is to reduce the search space for potential embedded PI submodels to the maximum extent. A single link search achieves this objective in three ways. The following proposition justifies the first.

Theorem 13 *If LIM returns a connected graph, the underlying PDM is not a PI model.*

Proof:

We prove by contraposition. Suppose the PDM is a PI model. By Theorem 12, LIM succeeds iff the PDM is not PI and hence LIM will fail, i.e., it will return a disconnected graph. □

Therefore, a multi-link search is *not* needed when a single link search learns a connected graph. The search space for embedded PI models has been reduced to nil, and we are confident that the outcome is correct. However, if the learned graph is disconnected, we are not certain if this is due to a PI model or the domain consists of truly marginally independent subsets. In this case, a multi-link search is needed.

The second way by which a single link search reduces the search space for PI submodels is justified by Corollary 14.

Corollary 14 *Let G be a (disconnected) graph returned by LIM. For each component of G, it does not contains any PI submodel.*

Proof:

Direct result from Theorem 13. □

Hence, a multi-link search can be safely focused on cross-component links only. For example, if G has only two components and one of them has a single variable, then a focused i-link search needs to examine only $O(N^i)$ instead of $O(N^{2i})$ links.

Third, variables in a marginally independent subset of an embedded PI submodel may be pairwise dependent (if the subset does not form an embedded PI submodel). A single link search will connect them. A less number of links are then needed at each multi-link search step in order to detect the submodel. For example, a submodel of four variables takes six links to complete. However, if it partitions into two subsets with two variables each, then a four-link search will detect the submodel.

9.2 Increasing number of links to search one-by-one

After a single link search is terminated with a disconnected graph, perform a 2-link search followed by a 3-link search, and so on. We shall denote an modified version of LIM extended with this strategy by LIM-ML. This strategy is based on Theorem 15.

Theorem 15 *If LIM-ML terminates a single link search or an i-link search with a connected graph G, then G is an I-map of the underlying PDM.*

Proof:

Let M be the PDM. If M is not a PI model, then an I-map G will be returned at the end of the single link search by Theorem 13. According to Theorem 12, if M is a PI model, then there exists a minimal I-map G_M of M such that links in each subgraph of G_M induced by a marginally independent subset are learned except some links involved in PI submodels may be missing, and all these subgraphs are disconnected from each other.

For each subsequent multi-link search, the subgraphs involved in a PI submodel will be connected if all variables contained in the submodel can be completed by the new links plus links learned earlier. Hence, all submodels will be completed if an i-link search terminates with a connected graph G. At this time, all links in G_M have been included in G. □

Based on Theorem 15, after a connected graph is learned in an i-link search, the learning can be terminated. This parallels the reduction of search space to nil in the single link search when the PDM is not a PI model (Section 9.1). The other two ways for reducing search space discussed in Section 9.1 are also paralleled by LIM-ML: Multi-link search for smaller i values are performed first before larger ones. This will complete smaller PI submodels first and allows search for larger PI submodels more focused. Some PI submodels may have a subset that itself is a PI submodel. This latter submodel will be completed first and allows i-link search with a smaller i value to complete the PI submodel at the next higher level.

9.3 Making learning inference-oriented

Similarly as discussed in Section 9.1, when LIM-ML terminates an i-link search with a disconnected graph, it is uncertain whether the PDM is a PI model or consists of truly marginally independent submodels. In the latter case, we are only certain when the multi-link search with $i = |N|$ has halted with a disconnected graph, which has an exponential complexity. What can we do if LIM-ML does not halt with a connected graph after multi-link search for a reasonably large i value has been performed?

A belief network whose structure is an I-map of the PDM can be used to answer any probabilistic queries. However, for a particular application, it may be the case that not every conceivable query will be used. If we can restrict the range of potential queries, then we can trade the generality of the learned structure with the computational complexity of learning. We discuss two possible restrictions of potential queries that can be exploited.

First, each of the potential queries may contain no more than k variables including both evidence variables (whose values are observed) and query variables (whose posterior distribution is requested), where k is a positive integer. We shall call such a query *k-query*. Formally, a k-query is of the form

$$\text{“}what\ is\ the\ value\ of\ P(A|B)?\text{''}$$

where $|A \cup B| \leq k$. If k is reasonably small, then we need only to make sure that all the PI submodels of a size no larger than k are learned. This is justified by Theorem 16.

Theorem 16 *Let M be a PDM that contains PI submodels and G be a structure returned by LIM-ML after $k(k-1)/2$-link search. Then a belief network T with the structure G answers all k-queries correctly.*

Proof:

After i-link searches where $i = 1, 2, \ldots, k(k-1)/2$, there exists a minimal I-map G_M of M such that G misses no links of G_M except marginally independent subsets of variables corresponding to PI submodels of more than k variables may not be completed. Any k-query that does not involve these PI submodels will be answered correctly using T since no dependence between variables involved in the query is missing from G. Any k-query that does involve these PI submodels will also be answered correctly. This is because for any PI submodel of n ($n > k$) variables, any $i \leq k$ variables in the submodel, partitioned into marginally independent subsets as in G, do not constrain each other (beyond the subset) in M any way. $\square$

A related restriction is obtained when potential queries can be classified into two groups: One group contains k-queries for some reasonably small k. The other group of queries involve more than k variables, but the entire set A ($|A| > k$) of variables involved in the group is not very large ($|A| << |N|$).

In this case, we can perform up to the $k(k-1)/2$-link search first, and then perform i-link searches for $i = 0.5k(k-1)+1, \ldots, 0.5|A|(|A|-1)$ focused only on the subdomain A. Each such focused search examines $O(|A|^{2i})$ sets of links. Compared with an unfocused i-link search which examines $O(|N|^{2i})$ sets of links, there is a significant reduction in complexity when $|A| << |N|$.

10 Experimental results

We demonstrate several results presented above using a specific implementation of LIM-ML. In Section 10.1, we describe features of our implementation besides the basic features of LIM-ML. In Section 10.2, we present our learning results. Our implementation uses a specific scoring metrics and learns a DMN. Since our main purpose is to demonstrate the search method used, the results are general to other related scoring metrics and learning Bayesian networks as well.

10.1 Algorithm

Our implementation is based on Algorithm 1. It is an enhancement of the algorithm in [31] by incorporating the results of Section 9.

The target representation is a decomposable Markov network (DMN). A brief discussion on learning Bayesian networks and DMNs is given in [31]. Algorithm 1 focuses on learning the structure of a DMN (lines 12 and 13). Once the chordal graph is obtained, marginal distributions on each clique can be estimated from the dataset. It is assumed that for each tuple in the dataset, there is no missing variables.

Algorithm 1

Input: A dataset D over a set N of variables, a maximum number κ of lookahead links, and a threshold δh.

begin
1 initialize a graph $G = (N, E = \phi)$;
2 $G' := G$;
3 for $i = 1$ to κ, do
4   repeat
5   initialize the entropy decrement $dh' := 0$;
6   for each set L of links ($|L| \leq i$, $L \cap E = \phi$), do
7   if $G^* = (N, E \cup L)$ is chordal and L is implied by a clique, then
8   compute the entropy decrement dh*;
9   if $dh^* > dh'$, then $dh' := dh^*$, $G' := G^*$;
10   if $dh' > \delta h$, then $G := G'$, $done := false$; else $done := true$;
11   until done = true;
12   if G is connected, then return G and halts;
13 return G and halts;
end

Instead of testing the conditional independence directly as used in LIM-ML, a test whether new links decrease the Kullback-Leibler cross entropy is performed (line 8). It has been shown [31] that (1) minimizing the K-L cross entropy between a dataset D and a DMN obtained from D is equivalent to minimizing the entropy of the DMN, and (2) a learning process starting from an empty DMN structure and driven by the minimization of its entropy is paralleled by the process of removing false independence (missing links relative to some minimal I-map) in the intermediate DMNs.

Since the entropy is a single real number, a threshold (δh) can be used to differentiate between a strong dependence and a weak one (may be due to noise). A greedy search can thus be applied (lines 6 through 10) to avoid adding unnecessary links and links due to weak dependence.

The single link search is performed first in Algorithm 1 ($i = 1$). One link is added after all links are examined. If no link decreases the entropy significantly, the single link search is terminated. If a connected chordal graph is obtained, Algorithm 1 halts based on Theorem 13. Otherwise, the multi-link search is performed next ($i > 1$). The algorithm may halt at the end of any i-link search if a connected graph is obtained. This is based on Theorem 15.

The maximum number κ of links to search can be specified by the user based on the potential query patterns as discussed in Section 9.3. Note that G may still be disconnected when the algorithm halts at line 13. The correctness of the future inference with anticipated queries is warranted even if some large PI submodels are still uncovered. This is justified by Theorem 16.

10.2 Experiments and results

An implementation of Algorithm 1 was tested with a series of experiments. The objective was to demonstrate the failure of single link search and the effectiveness of a cautious multi-link search as presented in Section 9. To this end, we need data from known PI models. We could generate models with a large number of variables embedded with smaller PI submodels. We have chosen not to do that for the following reasons:

A restricted version of Algorithm 1 [31] has been tested successfully using data generated from a non-PI model of 37 variables using the single link search. From the proof of Theorem 15, the effectiveness of a multi-link search is only affected by the sizes of embedded PI submodels, and is unaffected by the size of the PDM (except the efficiency). We have therefore used example PDMs given in this paper in our experiments.

First, a dataset of 1000 cases was generated from the full PI model of Table 1. Using $\kappa = 5$ and $\delta h = 0.008$, an empty graph was learned. Note that this result is equivalent to the results of five runs with $\kappa = 1, 2, 3, 4, 5$, respectively. Search of up to 5 links have all failed. The next run with $\kappa = 6$ accepted the link set L that contained all six possible links, returning the minimal I-map given in Figure 1 (left).

For the partial PI model in Table 2, a dataset of 1000 cases was generated. Using $\kappa = 1$ and $\delta h = 0.008$, only the link set

$$L_1 = \{(X_2, X_3)\}$$

was added. A graph with the single link was returned. The next run using $\kappa = 2$ added the link set

$$L_2 = \{(X_1, X_2), (X_1, X_3)\},$$

yielding the minimal I-map given in Figure 1 (middle).

Next, we generated a dataset of 2000 cases from the embedded PI model in Table 3. Using $\kappa = 1$ and $\delta h = 0.002$, three sets of links

$$L_1 = \{(X_1, X_4)\},\ L_2 = \{(X_2, X_3)\},\ L_3 = \{(X_3, X_5)\},$$

were added. The algorithm failed to learn the colored links in Figure 1 (right). The next run using $\kappa = 2$, the link set

$$L_4 = \{(X_1, X_2), (X_1, X_3)\}$$

was learned, yielding the minimal I-map given in Figure 1 (right).

Finally, we generated a dataset of 1500 cases from the music-box-dog-John domain. The successful run used $\kappa = 3$ and $\delta h = 0.005$. During the search with $i = 1$, only two sets of links

$$L_1 = \{(light1, dog)\},\ L_2 = \{(ball3, music\_box)\}$$

were learned. The learned network structure is shown in Figure 5 (a). Note that using this learned network, any query of the form $P(X|Y)$ where X and Y are distinct variables can be answered correctly, e.g.,

$$P(music\_box = p|ball3 = w),$$

$$P(music\_box = p | ball1 = w),$$

$$P(John = c | light1 = on).$$

However, not all queries that involve more than two variables can be answered correctly, e.g.,

$$P(dog = b | light1 = on, light2 = on).$$

Next during the search with $i = 2$, four additional sets of links were learned in the following order:

$$L_3 = \{(light1, light2), (light2, dog)\}, L_4 = \{(ball2, ball3), (ball2, music\_box)\},$$

$$L_5 = \{(ball1, ball3), (ball1, music\_box)\}, L_6 = \{(ball1, ball2)\}.$$

Figure 5 (b) was obtained first and then (c). Finally, during the search with $i = 3$, the last link set

$$L_6 = \{(John, music\_box), (music\_box, dog), (dog, John)\}$$

was learned, yielding the outcome in Figure 5 (d). Note that Algorithm 1 would terminate anyway at this stage even if we had used $\kappa > 3$.

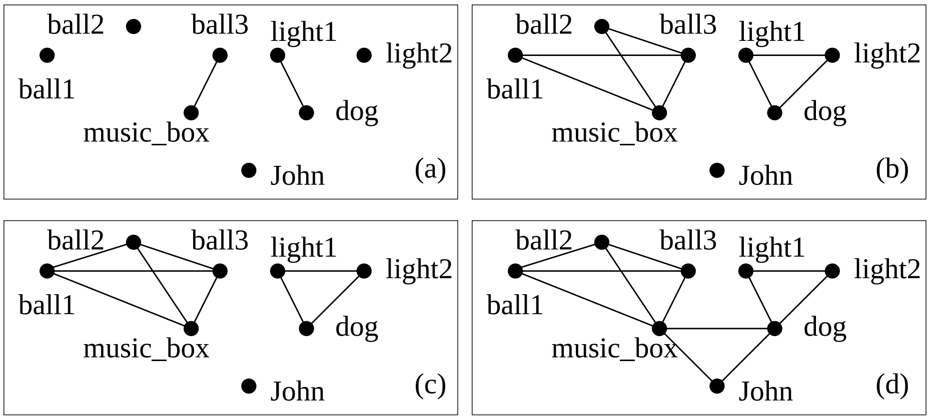


Figure 5: Progress in learning the music-box-dog-john model.

To demonstrate Theorem 13, we modified the music-box-dog-John slightly. We replaced $P(ball3 = w) = 0.5$ by 0.7 and $P(light1 = on) = 0.5$ by 0.8, which turned the PDM into a non-PI model. Algorithm 1 returned Figure 5 (d) after the single link search.

11 Remarks

In this paper, we proposed formal definitions of all three types of PI models, which forms a basis for the study of these models.

We provided a parameterization of PI models and showed that PI models form a superclass of the well known parity problems in machine learning as well as the modulus addition problems. This analysis provided evidence for the existence of PI models in practice.

We demonstrated that failure to learn such models may cause decision errors in subsequent inference, which motivated the need for learning these models.

Using an abstraction of a class of algorithms for learning belief networks, we have shown that PI models form the necessary and sufficient condition for the failure of algorithms based on a conditional independence test and a single link search. Coupled with a demonstration elsewhere [30] on the failure of several belief network learning algorithms in learning such models, our result suggests that PI models are hard to learn and more sophisticated search methods are needed for better management of PI models.

In order to tackle PI models in learning, we proposed several strategies to manage the increased computational complexity due to the need for multi-link search. The effectiveness of these strategies were demonstrated with experimental results.

Acknowledgement

This work is supported by Research Grant OGP0155425 from the Natural Sciences and Engineering Research Council (NSERC). The author is a member of the Institute for Robotics and Intelligent Systems (IRIS) and wishes 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] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, (2):159–225, 1994.

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

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

[4] 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.

[5] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The batmobile: towards a bayesian automated taxi. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence*, pages 1878–1885, Montreal, Canada, 1995.

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

[7] D. Heckerman, J.S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, pages 49–57, 1995.

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

[9] 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.

[10] A.L. Jensen and F.V. Jensen. MIDAS an influence diagram for management of mildew in winter wheat. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 349–356, 1996.

[11] F.V. Jensen. *An introduction to Bayesian networks.* UCL Press, 1996.

[12] 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.

[13] 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.

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

[15] S.L. Lauritzen, A.P. Dawid, B.N. Larsen, and H.G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.

[16] 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.

[17] S.L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, (17):31–57, 1989.

[18] 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.

[19] S.M. Mahoney and K.B. Laskey. Network engineering for complex belief networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 389–396, 1996.

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

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

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

[23] 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.

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

[25] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and Search.* Springer-Verlag, 1993.

[26] C. Thornton. Parity: the problem that won't go away. In G. McCalla, editor, *Advance in Artificial Intelligence*, pages 362–374. Springer, 1996.

[27] J. Whittaker. *Graphical models in applied multivariate statistics.* Wiley, 1990.

[28] 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.

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

[30] Y. Xiang, S.K.M. Wong, and N. Cercone. Critical remarks on single link search in learning belief networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 564–571, Portland, 1996.

[31] Y. Xiang, S.K.M. Wong, and N. Cercone. A 'microscopic' study of minimum entropy search in learning decomposable Markov networks. Technical Report CS-95-07, University of Regina, 1996. Modifed and submitted for publication.