Probabilistic Decision Graphs – Combining Verification and AI Techniques for Probabilistic Inference

Manfred Jaeger
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Abstract

We adopt probabilistic decision graphs developed in the field of automated verification as a tool for probabilistic model representation and inference. We show that probabilistic inference has linear time complexity in the size of the probabilistic decision graph, that the smallest probabilistic decision graph for a given distribution is at most as large as the smallest junction tree for the same distribution, and that in some cases it can in fact be much smaller. Behind these very promising features of probabilistic decision graphs lies the fact that they integrate into a single coherent framework a number of representational and algorithmic optimizations developed for Bayesian networks (use of hidden variables, context-specific independence, structured representation of conditional probability tables).

1 Introduction

Over the past 15 years Bayesian networks have been developed in AI as the main representation tool for probability distributions on finite state spaces. In about the same period of time, ordered binary decision diagrams (OBDDs) (Bryant 1986) have emerged in automated verification as the primary representation tool for Boolean functions, and have also been adopted for the representation of arbitrary real-valued functions (Fujita, McGeer & Yang 1997, Lai & Sastry 1992), and, in particular, probability distributions (Bozga & Maler 1999).

Bayesian networks and OBDD-based representation frameworks of probability distributions are developed with similar goals: to obtain compact representations of probability distributions on which certain basic operations can be performed efficiently. The types of operation one is interested in, however, are somewhat different. A basic problem motivating the whole OBDD approach, for instance, is equality testing, i.e. checking whether two OBDDs (or probabilistic versions thereof) represent the same Boolean function (probability distribution). This question has not attracted much attention in AI. The probabilistic inference problems considered in AI, on the other hand, play no prominent role in the verification literature.

In spite of some references to the use of OBDDs for some specialized subtasks in probabilistic representation and inference (Boutilier, Friedman, Goldszmidt & Koller 1996, Nielsen, Wuillemin, Jensen & Kjærulff 2000), there thus has not been any rigorous appraisal of the merits of (probabilistic) OBDD technology from the AI point of view. This paper is meant to provide such an appraisal. To this end we first introduce a generalization of Bozga and Maler's (1999) Probabilistic Decision Graphs, and show how to perform basic probabilistic inference problems on this representation. As it will turn out that these inference problems have linear time complexity in the size the representation, a crucial question will be how large a representation of a distribution as a probabilistic decision graph will be in comparison to a representation as a junction-tree. This question is answered in section 4. We also discuss the problem of learning probabilistic decision graphs from data, and argue that there can be substantial benefits in learning a decision graph, rather than a Bayesian network.

2 Definitions

Throughout the remainder of this paper $\boldsymbol{X} = X_1, \ldots, X_n$ denotes a set of random variables. The range (set of possible values) of X_i is $R(X_i) = \{x_{i,1}, \ldots, x_{i,k_i}\}$. The product set $R(X_1) \times \ldots \times R(X_n)$ is denoted by W. We are interested in representations for the joint distribution of $\boldsymbol{X}$ on W.

Definition 2.1 Let T be a tree with nodes $\{X_1, \ldots, X_n\}$ and direct successor relation E_T. A *real function graph (RFG)* for $\boldsymbol{X}$ with respect to the tree order T is a rooted directed acyclic graph $G = (V, E)$, such that

- each node $\nu \in V$ is labeled with a variable $X_i \in \boldsymbol{X}$. A node labeled with X_i also is labeled with a vector $(p_1^\nu, \ldots, p_{k_i}^\nu) \in \mathbb{R}^{k_i}$.
- For each node ν labeled with X_i, each $x_{i,h} \in R(X_i)$, and each $X_j \in \boldsymbol{X}$ with $(X_i, X_j) \in E_T$ there exists exactly one edge e (labeled with $x_{i,h}$) in E leading from ν to a node $\nu' \in V$ labeled with X_j.

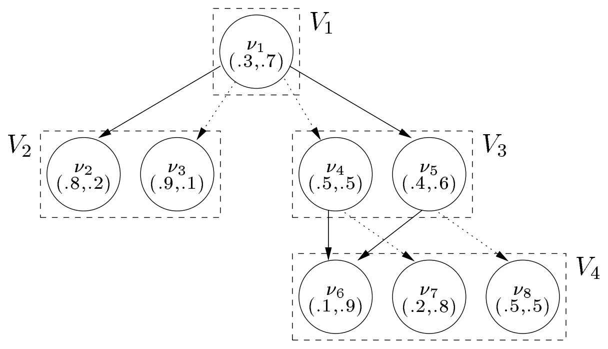

Figure 1: Probabilistic Decision Graph

A RFG G is called a *probabilistic decision graph (PDG)* if for all nodes ν with label X_i it holds that $p^\nu_h \in [0,1]$ and $\sum_{h=1}^{k_i} p^\nu_h = 1$.

We also introduce the following notation: with respect to the tree T we define for $X_i \in \boldsymbol{X}$: $\mathit{succ}_T(X_i) := \{X_j \in \boldsymbol{X} \mid (X_i, X_j) \in E_T\}$, $\mathit{succ}^*_T(X_i) := \{X_j \in \boldsymbol{X} \mid (X_i, X_j) \in E^*_T\}$, where E^*_T is the reflexive and transitive closure of E_T. Also we denote with $\mathit{pred}_T(X_i)$ the unique immediate predecessor of X_i in T ($\mathit{pred}_T(X_i) = \emptyset$ if X_i is the root). With respect to a RFG G we let V_i denote the set of all nodes in V labeled with X_i. When $X_j \in \mathit{succ}_T(X_i)$, $\nu \in V_i$, and $1 \leq h \leq k_i$, then we denote with $\mathit{succ}(\nu, X_j, x_{i,h})$ the node $\nu' \in V_j$ that is connected to ν via the (unique) edge $(\nu, \nu') \in E$ labeled with $x_{i,h}$. Finally, we use ρ to denote the root of G.

Figure 1 illustrates the definitions given so far. It shows a PDG for four binary random variables $X_1, \ldots, X_4$. The underlying tree T is given by the edges $E_T = \{(X_1, X_2), (X_1, X_3), (X_3, X_4)\}$. The labels of the edges in E are represented by their style: dotted for $x_{i,1} = 0$, solid for $x_{i,2} = 1$. The labeling of the nodes with elements from $\boldsymbol{X}$ is represented by the sets V_i. In this example $\mathit{succ}(\nu_1, X_2, 0) = \nu_3$, and $\mathit{succ}(\nu_4, X_4, 1) = \mathit{succ}(\nu_5, X_4, 1) = \nu_6$.

We now turn to the semantics of a RFG, which essentially consists of a real-valued function on W. This is defined recursively via a definition of a function f^ν_G for each node ν.

Definition 2.2 Let $G = (V, E)$ be a RFG w.r.t. T, $\nu \in V_i$. Let $\mathit{succ}_T(X_i) = \{Y_1, \ldots, Y_l\} \subseteq \boldsymbol{X}$. A real-valued function f^ν_G is defined on $\times_{X_j \in \mathit{succ}^*_T(X_i)} R(X_j)$ by

$$f^\nu_G(x_{i,h}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) := p^\nu_h \prod_{\lambda=1}^{l} f_G^{\mathit{succ}(\nu, Y_\lambda, x_{i,h})}(\boldsymbol{z}_\lambda) \tag{1}$$

($x_{i,h} \in R(X_i), \boldsymbol{z}_\lambda \in \times_{Z \in \mathit{succ}^*_T(Y_\lambda)} R(Z)$). The function f^ρ_G on W defined for the root ρ of G is simply denoted by f_G. When G is a PDG, then f_G defines a probability distribution on W, which we denote with P_G.

Note that for leaf nodes ν the right hand side of (1) reduces to p^ν_h, so that the recursive definition is well-founded.

For the PDG in Figure 1 we obtain, for instance, $f^{\nu_5}(X_3 = 1, X_4 = 0) = 0.6 \cdot 0.1$, and $f^{\nu_1}(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = 0.7 \cdot 0.8 \cdot 0.6 \cdot 0.1$.

Like Bayesian networks, PDGs encode by their structure certain conditional independence relations. These independence relations, however, are not expressible by equations of the form $P(X_i \mid X_1, \ldots, X_{i-1}) = P(X_i \mid \mathit{Pa}(X_i))$, where $\mathit{Pa}(X_i) \subseteq \{X_1, \ldots, X_{i-1}\}$ is a set of random variables, but in terms of equalities between conditional distributions given partitions of the state space: $P(X_i \mid X_1, \ldots, X_{i-1}) = P(X_i \mid \mathscr{A}_i)$, where $\mathscr{A}_i$ is a partition of W that is definable by $X_1, \ldots, X_{i-1}$ (see (Billingsley 1986, Section 33) for the basic notion of a conditional distribution given a partition of the state space). In the PDG of Figure 1, for example, we have $P(X_4 \mid X_1, X_2, X_3) = P(X_4 \mid \mathscr{A}_4)$, where $\mathscr{A}_4 = \{\{X_3 = 1\}, \{X_1 = 1, X_3 = 0\}, \{X_1 = 0, X_3 = 0\}\}$. The general form of the factorization by a PDG structure then is

$$P_G = \prod_{i=1}^{n} P_G(X_i \mid \mathscr{A}_i), \tag{2}$$

where $|\mathscr{A}_i| = |V_i|$.

From this basic factorization further conditional independence relations in P_G can be deduced by a PDG-analogon of the d-separation condition. These derived conditional independencies, again, are in terms of partitions rather than variables, and take the general form $P_G(X_i \mid \mathscr{B}, \mathscr{C}) = P_G(X_i \mid \mathscr{B})$ for certain partitions $\mathscr{B}, \mathscr{C}$. Independence structures captured by Bayesian networks and by PDGs are different: conditional independence relations expressed by PDGs in general cannot be expressed by Bayesian network structures, and vice versa. We defer a more detailed analysis of this topic to an extended version of this paper.

3 Probabilistic Inference

We now turn to the question of how to compute posterior probabilities from PDG representations. Central to the solution of this problem are the concepts of the *in-flow* and *out-flow* of a node.

Definition 3.1 Let $G = (V, E)$ be an RFG for $\boldsymbol{X}$ with respect to T. Let $\nu \in V_i$. The out-flow of ν is defined as

$$\mathit{ofl}(\nu) := \sum_{\boldsymbol{z} \in \times_{X_j \in \mathit{succ}^*_T(X_i)} R(X_j)} f^\nu_G(\boldsymbol{z}).$$

Thus, the outflow of ν is just the sum of all values of f^{ν}_G.

The definition of the in-flow of a node is slightly more involved, and we first have to provide some terminology for paths in G. Let $\boldsymbol{Y}$ be a subset of $\boldsymbol{X}$ that is upward closed in T, i.e. $X_i \in \boldsymbol{Y}$ and $(X_j, X_i) \in E_T$ implies $X_j \in \boldsymbol{Y}$. Let $\boldsymbol{Y}^+ := \boldsymbol{Y} \cup \{X_i \mid \mathit{pred}_T(X_i) \in \boldsymbol{Y}\}$. Any instantiation $\boldsymbol{y} \in \times_{Y \in \boldsymbol{Y}} R(Y)$ defines a subtree of G with root ρ that contains exactly one node from each V_i with $X_i \in \boldsymbol{Y}^+$. We refer to such an instantiation as a *path* in G. In Figure 1, for instance, for $\boldsymbol{Y} = \{X_1\}$ we obtain $\boldsymbol{Y}^+ = \{X_1, X_2, X_3\}$, and the instantiation, or path, $X_1 = 1$ defines the subtree consisting of the nodes ν_1, ν_2, ν_5.

Now let $\boldsymbol{Y} \subseteq \boldsymbol{X}$ be upward closed and $X_i \in \boldsymbol{Y}^+$. Each path $\boldsymbol{y} \in \times_{Y \in \boldsymbol{Y}} R(Y)$ then defines the unique $\nu \in V_i$ contained in the subtree defined by $\boldsymbol{y}$. We say that ν is *reached by* $\boldsymbol{y}$, and define

$$\mathit{Path}(\nu, \boldsymbol{Y}) := \{\boldsymbol{y} \in \times_{Y \in \boldsymbol{Y}} R(Y) \mid \nu \text{ reached by } \boldsymbol{y}\}.$$

For the PDG of Figure 1 we have, for example: $\mathit{Path}(\nu_6, (X_1, X_3)) = \{(0,1), (1,1)\}$, and $\mathit{Path}(\nu_6, (X_1, X_2, X_3)) = \{(0,0,1), (0,1,1), (1,0,1), (1,1,1)\}$.

Definition 3.2 Let G and T be as in Definition 3.1. Let $\nu \in V_i$, $\nu \neq \rho$, and let $G \setminus X_i$ denote the RFG over $\boldsymbol{X} \setminus \mathit{succ}^*_T(X_i)$ obtained by removing all nodes labeled with some $X_j \in \mathit{succ}^*_T(X_i)$ from G. The in-flow of ν is defined as

$$\mathit{ifl}(\nu) := \sum_{\boldsymbol{y} \in \mathit{Path}(\nu, \boldsymbol{X} \setminus \mathit{succ}^*_T(X_i))} f_{G \setminus X_i}(\boldsymbol{y}). \tag{3}$$

For the root we define $\mathit{ifl}(\rho) := 1$.

To provide a better intuition for the quantities *ifl* and *ofl*, we note that $\mathit{ifl}(\nu) \cdot \mathit{ofl}(\nu)$ is equal to $\sum_{\boldsymbol{x} \in \mathit{Path}(\nu, \boldsymbol{X})} f_G(\boldsymbol{x})$.

The following lemma is the basis for an efficient computation of *ifl* and *ofl* for all nodes in a RFG.

Lemma 3.3 (a) Let $\nu \in V_i$. Then

$$\mathit{ofl}(\nu) = \sum_{h=1}^{k_i} p^{\nu}_h \prod_{Y \in \mathit{succ}_T(X_i)} \mathit{ofl}(\mathit{succ}(\nu, Y, x_{i,h})). \tag{4}$$

(b) Let $\nu \in V_i$, $\nu \neq \rho$. Assume that $\mathit{pred}_T(X_i) = X_j$. Then

$$\mathit{ifl}(\nu) = \sum_{h=1}^{k_j} \sum_{\substack{\nu' \in V_j: \\ \nu = \mathit{succ}(\nu', X_i, x_{j,h})}} [\mathit{ifl}(\nu') p^{\nu'}_h \prod_{Y \in \mathit{succ}_T(X_j) \setminus X_i} \mathit{ofl}(\mathit{succ}(\nu', Y, x_{j,h}))]. \tag{5}$$

Equations (4) and (5) give rise to a simple procedure for computing $\mathit{ifl}(\nu)$ and $\mathit{ofl}(\nu)$ for all nodes ν in time linear in the size of G: first $\mathit{ofl}(\nu)$ is computed in bottom up pass through G starting with the leaves, then $\mathit{ifl}(\nu)$ is computed in a top down pass starting with the root.

For a PDG G the computations of *ifl* and *ofl* serve to answer simple probabilistic queries. The most basic probabilistic query one needs to solve is the computation of the probability $P_G(\boldsymbol{Y} = \boldsymbol{y})$ of an instantiation of a subset of variables $\boldsymbol{Y} \subseteq \boldsymbol{X}$ to the values $\boldsymbol{y} \in \times_{Y \in \boldsymbol{Y}} R(Y)$. This can be done by transforming G into a RFG $G_{\boldsymbol{Y}=\boldsymbol{y}}$ as follows: for all $X_j \in \boldsymbol{Y}$ and all $\nu \in V_j$ change p^{ν}_h to 0 if $x_{j,h}$ is not the value to which X_j is instantiated in $\boldsymbol{y}$ (otherwise leave p^{ν}_h unchanged). For $\boldsymbol{x} \in W$ we then have

$$f_{G_{\boldsymbol{Y}=\boldsymbol{y}}}(\boldsymbol{x}) = \begin{cases} f_G(\boldsymbol{x}) & \text{if } \boldsymbol{x} \text{ satisfies } \boldsymbol{Y} = \boldsymbol{y} \\ 0 & \text{else.} \end{cases}$$

It follows that the root in $G_{\boldsymbol{Y}=\boldsymbol{y}}$ has out-flow equal to $P_G(\boldsymbol{Y} = \boldsymbol{y})$. As $G_{\boldsymbol{Y}=\boldsymbol{y}}$ and the out-flow of its root can be computed in time linear in the size of G, we see that probabilities of events of the form $\boldsymbol{Y} = \boldsymbol{y}$ can be computed in linear time based on a PDG representation.

A slightly more complicated probabilistic inference problem is the computation of a posterior distribution $P_G(X_j \mid \boldsymbol{Y} = \boldsymbol{y})$ for a variable $X_j \notin \boldsymbol{Y}$ given (the "evidence") $\boldsymbol{Y} = \boldsymbol{y}$. Of course, this can be reduced to a number of computations of $P_G(X_j = x_{j,h}, \boldsymbol{Y} = \boldsymbol{y})$ by the method already described. However, the posterior distribution of X_j can also be read off the RFG $G_{\boldsymbol{Y}=\boldsymbol{y}}$ directly, because

$$P_G(X_j = x_{j,h} \mid \boldsymbol{Y} = \boldsymbol{y}) = \frac{1}{\mathit{ofl}(\rho)} \sum_{\nu \in V_j} \mathit{ifl}(\nu) p^{\nu}_h \prod_{Y \in \mathit{succ}_T(X_j)} \mathit{ofl}(\mathit{succ}(\nu, Y, x_{j,h})) \tag{6}$$

(where *ifl* and *ofl* are computed in $G_{\boldsymbol{Y}=\boldsymbol{y}}$).

Instead of computing the posterior distribution of a variable X_j given an instantiation $\boldsymbol{Y} = \boldsymbol{y}$ as evidence, one may also be interested in computing the posterior distribution of X_j given evidence of a more general form, e.g. disjunctive evidence like $X_3 = x_{3,2} \vee X_3 = x_{3,7}$. We now show that the approach used to compute $P_G(X_i \mid \boldsymbol{Y} = \boldsymbol{y})$ can be extended in a very coherent way to compute posterior distributions $P_G(X_i \mid \mathcal{E})$, where $\mathcal{E}$ can, in principle, be any subset of W that is given as evidence. As to be expected, the linear time complexity of the computation of $P_G(X_i \mid \boldsymbol{Y} = \boldsymbol{y})$ can not be maintained for general evidence sets $\mathcal{E}$.

The function $f_{G_{\boldsymbol{Y}=\boldsymbol{y}}}$ can also be written as the product $f_G \cdot 1_{\boldsymbol{Y}=\boldsymbol{y}}$, where $1_{\boldsymbol{Y}=\boldsymbol{y}}$ is the indicator function of $\boldsymbol{Y} = \boldsymbol{y}$, i.e. $1_{\boldsymbol{Y}=\boldsymbol{y}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x}$ satisfies $\boldsymbol{Y} = \boldsymbol{y}$, and $1_{\boldsymbol{Y}=\boldsymbol{y}}(\boldsymbol{x}) = 0$ else. To generalize our approach to the computation of posterior distributions, it is sufficient to

```
**Algorithm:** multiply-rfg($\rho_G, \rho_H$: roots of RFGs w.r.t. $T$)
$i$:= index of variable at the root of $T$
$\rho$ := new node labeled with $X_i$;
for $h = 1, \ldots, k_i$ do
    $p_h^{\rho} := p_h^{\rho_G} \cdot p_h^{\rho_H}$;
    for all $Y \in succ_T(X_i)$ do
        $succ(\rho, Y, x_{i,h}) :=$
        $multiply\text{-}rfg(succ(\rho_G, Y, x_{i,h}), succ(\rho_H, Y, x_{i,h}))$;
    end
end
return $\rho$.
```

Table 1: Multiplication algorithm

show how to compute RFGs representing functions of the form $f_G \cdot 1_{\mathcal{E}}$ with $\mathcal{E} \subseteq W$ in general.

We first observe that indicator functions $1_{\mathcal{E}}$ also can be represented by RFGs:

Definition 3.4 A RFG H for $\boldsymbol{X}$ is called an *indicator graph*, iff $p_h^{\nu} \in \{0,1\}$ for all ν, h. Then $f_H(\boldsymbol{x}) \in \{0,1\}$ for all $\boldsymbol{x} \in W$, and $E_H := \{\boldsymbol{x} \mid f_G(\boldsymbol{x}) = 1\}$ is the event defined by H.

Indicator graphs, of course, are closely related to OBDDs, which are essentially the subclass of indicator graphs for which the underlying T is a linear order.

The computation of a RFG representation of $f_G \cdot 1_{\mathcal{E}}$, thus, becomes a special case of the general problem of computing for two RFGs G and H a RFG $G \cdot H$ representing the product $f_G \cdot f_H$. We now show how this can be accomplished in general, in essence by extending Bryant's (1986) method for performing Boolean operations on OBDDs.

Bryant's algorithm requires OBDD's with respect to the same variable order as input. Similarly, our basic multiplication algorithm requires that the two factors of the multiplication are RFGs with respect to the same underlying tree structure T.

A high-level description of the algorithm is given in table 1. In its formulation an RFG is identified with its root node, and nodes here are also understood to store the information on their successor nodes *succ*$(\nu, \cdot, \cdot)$. To make the algorithm efficient, it is necessary to do some bookkeeping in order to avoid evaluating recursive calls with identical arguments more than once. This can be done as for Boolean operations on OBDDs, which leads to the following complexity result.

Theorem 3.5 Let $G = (V, E_G)$ and $H = (U, E_H)$ be two RFGs for $\boldsymbol{X}$ w.r.t. the same tree order T. Then an RFG $G \cdot H$ representing $f_G \cdot f_H$ can be computed in time $\mathcal{O}(\sum_{i=1}^{n} |V_i| \cdot |U_i| \cdot k_i)$.

Figure 2 shows an indicator graph for the tree structure of Figure 1 and the instantiation $X_2 = 1, X_3 = 0$. As an indicator graph for an instantiation can be given for any tree structure, and satisfies $|V_i| = 1$ for all i, we find that the general complexity result of Theorem 3.5 gives the same linear bound on the computation of a representation for $G \cdot 1_{\boldsymbol{Y}=\boldsymbol{y}}$ as we found before (and, in fact, the computation of $G \cdot 1_{\boldsymbol{Y}=\boldsymbol{y}}$ via the general algorithm of Table 1 reduces to a traversal of G and the setting to 0 of parameters p_h^{ν} when $\nu \in V_i$, and $x_{i,h}$ is inconsistent with $\boldsymbol{Y} = \boldsymbol{y}$).

Not every set $\mathcal{E} \subseteq W$ can be represented with an indicator graph that has the same tree structure T as a given PDG. However, it is always possible to represent $\mathcal{E}$ with a tree structure T' that is a refinement of T in the sense that for all $X_i, X_j : X_j \in succ_T^*(X_i) \Rightarrow X_j \in succ_{T'}^*(X_i)$. A possible choice for T' is a total order consistent with the partial order of T, because every indicator function $1_{\mathcal{E}}$ can be represented on the basis of such T'. For multiplication, the PDG G then first has to be transformed into a PDG G' with tree structure T'. While algorithmically not difficult, this transformation can cause an exponential blowup in the size of the PDG.

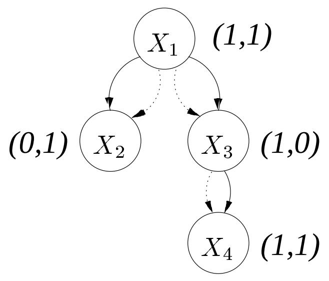

Figure 2: Indicator graph for partial instantiation

4 PDGs vs. Junction Trees

As probabilistic inference is linear in the size of a PDG, but NP-hard in the size of a Bayesian network representation (Cooper 1990), the respective sizes of PDG and Bayesian network representations are not an appropriate measure to compare the efficiency of these two frameworks.

The most popular approach for probabilistic inference from Bayesian networks is to transform a given Bayesian network into a *junction tree* representing the same distribution, and then compute probabilities on the junction tree representation. The transformation of a Bayesian network into a junction tree can cause an exponential blowup, but inference on junction trees then is linear in the size of the junction tree (see e.g. (Jensen 2001, Cowell, Dawid, Lauritzen & Spiegelhalter 1999)). Other methods for probabilistic inference from Bayesian networks have the same complexity characteristics as the junction tree approach.

The pertinent comparison we have to make, therefore, is between the sizes of PDGs and junction tree representations of a probability distribution.

Theorem 4.1 There exists an effective transformation that takes a junction tree J as input, and returns a PDG G representing the same distribution as J. The size of G is linear in the size of J.

Proof: (Sketch) Let J be a junction tree. For a node C of J denote with $\mathit{var}(C)$ the set of variables from $\boldsymbol{X}$ that appear in C. Turn J into a directed tree by taking an arbitrary node as the root and directing all edges away from the root. With every node C of J we can then associate the set $\mathit{new}(C) \subseteq \mathit{var}(C)$ of variables from $\boldsymbol{X}$ that are not contained in the parent node $\mathit{pred}_J(C)$ of C. The tree order T for G is now obtained from this directed junction tree by "expanding" each node C into a linear sequence of nodes, one for each $X_i \in \mathit{new}(C)$. By induction on the number of nodes in J one now shows that one can construct a PDG G w.r.t. T, which represents the same distribution as J, and for each node C of the original junction tree the following holds for the cardinalities of the node sets V_i:

$$\sum_{i: X_i \in \mathit{new}(C)} |V_i||R(X_i)| \leq \prod_{X \in \mathit{var}(C)} |R(X)|.$$

The theorem then follows, because the total sizes of G, respectively J, are given (up to linear factors) by summing the left, respectively right, side of this inequality over all nodes C of J. $\square$

Theorem 4.1 shows that PDG representations are always as efficient as Bayesian network representations. The following example shows that in some cases they are more efficient.

Example 4.2 Let $X_1, \ldots, X_{n-1}$ be independent binary random variables with $P(X_i = 1) = 1/2$ ($i = 1, \ldots, n-1$), and X_n a random variable with

$$P(X_n = 1 \mid X_1 = e_1, \ldots, X_{n-1} = e_{n-1}) \\ = \begin{cases} 0 & \text{if } \sum_{j=1}^{n-1} e_j \bmod 2 = 0 \\ 1 & \text{if } \sum_{j=1}^{n-1} e_j \bmod 2 = 1. \end{cases}$$

The joint distribution of $X_1, \ldots, X_n$ then models the generation of an $n-1$ bit random number with an added parity check bit. A Bayesian network representation of this distribution is shown in Figure 3 (a). The junction-tree constructed from this network (as well as any other junction-tree for P) is of exponential size in n. Figure 3 (b) shows the structure of a PDG representation of P, which is linear in n.

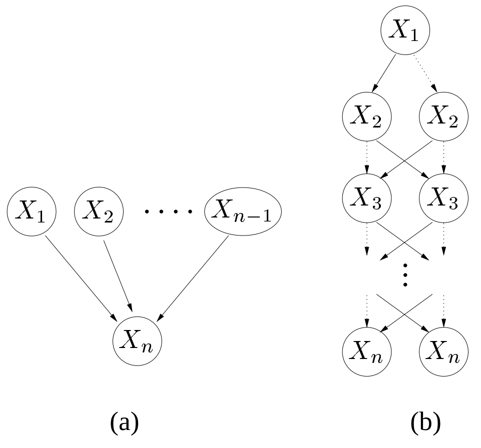

Figure 3: Bayesian network and PDG representations

In the preceding example any direct Bayesian network representation of P is of exponential size. However, one can reduce the size of the representation by introducing suitable auxiliary ("hidden") variables and represent the joint distribution of the X_i and the new variables: while X_n depends on all of $X_1, \ldots, X_{n-1}$, it becomes independent from $X_1, \ldots, X_{n-2}$ given $\sum_{j=1}^{n-2} X_j \bmod 2$. This can be utilized by introducing a new random variable $Y := \sum_{j=1}^{n-2} X_j \bmod 2$, and decompose the network by introducing Y as an intermediary variable between $X_1, \ldots, X_{n-2}$ and X_n. This process can be iterated, thus effectively replacing the exponentially large conditional probability table at X_n with a network of hidden variables. Both the size of the resulting Bayesian network and its junction-tree then are linear in n. The following theorem shows that hidden variables provide a general method for making Bayesian network representations as efficient as PDGs.

Theorem 4.3 There exists an effective transformation that takes a PDG G as input, and returns a Bayesian network B with the following properties: when $\boldsymbol{X} = X_1, \ldots, X_n$ are the random variables of G, then B has nodes $\boldsymbol{X} \cup \{N_i \mid i = 1, \ldots, n\}$; the marginal distribution defined by B on $\boldsymbol{X}$ is equal to the distribution defined by G, and there exists a junction tree J for B whose size is quadratic in the size of G.

Proof: (Sketch) Let $G = (V, E)$ with tree structure T be given. Let N_i be a random variable with range $R(N_i) := V_i$ ($i = 1, \ldots, n$). The network structure of B is defined by $\mathit{pred}_B(X_i) := N_i$ for all i, and $\mathit{pred}(N_i) := \{X_j, N_j\}$ when $X_i = \mathit{succ}_T(X_j)$. With this network structure P_G can be encoded. The cliques of the junction tree obtained from B by the usual construction are sets of the form $\{N_j, X_j, N_i\}$ and thus are labeled with tables of size $|V_j||R(X_j)||V_i|$. □

A more detailed analysis furthermore shows that the size of J obtained in this construction is only linear in the size of G when one does not explicitly represent rows with value 0 in the tables of J.

5 Construction of PDGs

Our results so far demonstrate that PDG representations of probability distributions provide a basis for probabilistic inference with several potential advantages over Bayesian network or junction-tree representations. The question then is, how do we obtain a PDG representation? In this section we briefly discuss three possible approaches: direct specification, compilation of a Bayesian network, and learning from data.

Direct Specification

Like a Bayesian network, a PDG may be specified directly by a domain expert. In some cases this can be straightforward (as in Example 4.2) and rather more natural than the specification of a Bayesian network for the same distribution. In other cases the potentially large number of nodes allocated to each variable X_i will make a direct specification of a large scale PDG rather cumbersome.

One reason why the specification of a large probability distribution by a Bayesian network is feasible is that the specification task is decomposed into the problem of specifying a number of relatively small conditional distributions. It turns out that a similar compositional specification is possible for PDGs. We illustrate the general approach by an example.

Figure 4 (a) shows a PDG for three binary random variables A, B, C (according to which B and C are independent given A). Figure 4 (b) shows a RFG that specifies the conditional distribution of another variable D given B and C. Note that according to this RFG D is conditionally independent from C given $B = 1$. A joint distribution of A, B, C, D now is the product of the functions defined by (a) and (b). A PDG representing this product can be computed with the multiplication algorithm of Section 3. This means that first for the variables A, B, C, D a tree structure T has to be determined which is a refinement of the tree structures of the individual factors. In this example, the only solution is the linear order A, B, C, D. Transformations of the two factors (a) and (b) into RFGs for this linear order are shown as (c) and (d). Note that we also introduced "dummy" nodes for variables that originally did not appear in the individual factors. Finally, (c) and (d) are multiplied, yielding the final PDG (e). The size of the final PDG will depend on the choice of the tree structure T. The problem of finding a tree structure that minimizes the size of the product is somewhat analogous to the problem of finding an optimal triangulation for a Bayesian network in order to minimize the size of the induced junction tree.

Compilation

The preceding discussion also points to a method for automatically compiling a Bayesian network into a PDG: one can first rewrite the conditional probability tables of the network as RFGs similar in form to Figure 4 (b), and then compute the product. A second approach is provided by the results of Section 4: one can first construct a junction tree from the Bayesian network and then turn the junction tree into a PDG, as described in the proof of Theorem 4.1. This latter approach has the advantage that it can utilize all the techniques that have been developed for the construction of small junction trees. The first approach, on the other hand, can utilize at an early stage in the construction process potential conditional independencies, which can already greatly reduce the size of the RFGs representing the individual conditional probability tables.

Learning from data

PDGs can be learned from empirical data with essentially the same techniques as used for Bayesian networks. When the structure (V, E) of a PDG is given, then maximum likelihood estimates $\hat{p}_h^\nu$ for the parameters are obtained from the empirical conditional distributions $P^{data}(X_i \mid \mathscr{A}_i)$ in the data. In the case of incomplete data, the EM-algorithm can be used. Due to the linear time complexity of probabilistic inference, a single expectation step of the EM-algorithm has time complexity $O(NK)$, where N is the number of data items and $K = \sum_i |V_i| k_i$ is the size of the PDG (the computation of the expected conditional distributions given a single data item essentially consists of one outflow/inflow computation, as in probabilistic inference). The maximization step requires time $O(K)$, so that the overall complexity of a single iteration of the EM-algorithm is $O(NK)$.

Structure learning for PDGs has some interesting aspects. At first this might seem like an almost hopeless task, as the space of all PDG-structures is considerably larger than the space of all Bayesian network structures for the same set of variables (a very coarse lower bound for all PDG-structures for n variables is 2^{2^n}, whereas a coarse upper bound for all Bayesian network structures is 2^{n^2}). However, there are some mitigating factors: first, the space of PDG-structures has a hierarchical structure determined by the level of possible tree structures T, and, for each T, the possible refinements to a graph (V, E). This gives rise to hierarchical search strategies where a top-level search over T-structures is guided by approximate optimizations of the exact graph structure given

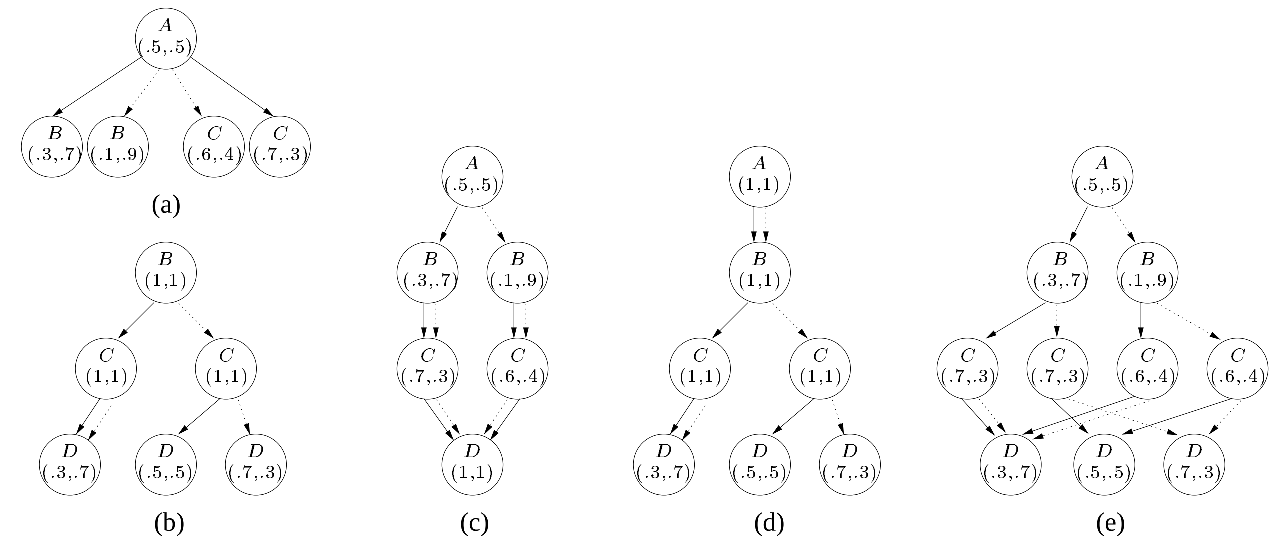

Figure 4: Compositional specification

the T-structure. For any given T-structure, the space of possible PDG-structures can be explored using elementary *split* and *join* operations: a split operation replaces a node ν with $l \geq 2$ predecessors with l distinct copies, each having one of ν's predecessors as a parent, and all having the same successors as ν. One can show that for every PDG-structure that does not realize the maximal likelihood score for PDG-structures over the same T-structure, there exists at least one split operation that will lead to a structure with higher likelihood score. A join operation reduces model complexity by merging nodes $\nu, \nu' \in V_i$ with $\mathit{succ}(\nu, Y, x_{i,h}) = \mathit{succ}(\nu', Y, x_{i,h})$ for all $Y \in \mathit{succ}_T(X_i), 1 \leq h \leq k_i$, and $(p_1^\nu, \ldots, p_{k_i}^\nu) \approx (p_1^{\nu'}, \ldots, p_{k_i}^{\nu'})$.

A major advantage of learning PDGs rather than Bayesian networks lies in the fact that here a score function like MDL-score that penalizes model complexity directly penalize the *inferential complexity* of a model, rather than merely its *representational complexity*. In contrast, MDL-score does not distinguish between two Bayesian networks of the same size (and having the same likelihood score), but with perhaps widely different behavior in terms of inference complexity. As eventually we will want to use the model for inference (not for compressing the observed data!), it is very desirable to guide the model search directly towards those models that will perform well on inference problems.

6 Related Work

As noted in Section 1, our definition of probabilistic decision graphs is based on a definition originally proposed by Bozga and Maler (1999). Probabilistic decision graphs in the sense of Bozga and Maler are essentially PDGs as introduced here with an underlying linear order T. The questions that Bozga and Maler then investigate in connection with these representations are quite distinct from the questions considered in this paper.

Several proposals have been made to encode conditional probability tables of Bayesian networks with (decision-) trees that make use of "context-specific"-independence relations within the conditional distribution of a variable X_i given its parents *Pa*(X_i) (Boutilier et al. 1996, Zhang & Poole 1999, Cano, Moral & Salmeron 2000). The possible use of OBDDs instead of trees has also been mentioned (Boutilier et al. 1996). These approaches can be seen as hybrid frameworks combining elements of "pure" Bayesian network and PDG representations. A number of adaptations of standard Bayesian network inference techniques to such structured representations of conditional probability tables have been described: Boutelier et al. (1996) suggest to use the structure in the conditional distributions either to decompose the network by introducing auxiliary variables that reduce the overall network connectivity, or to obtain more efficient strategies for cutset conditioning. Zhang and Poole (1999) show how to perform variable elimination on tree representations of cpts. In none of these approaches has it been possible to quantify the gain in inferential tractability afforded by the more compact representations. This is not surprising, because the compactness of the new cpt-representations in general will not be preserved under the multiplication and marginalization operations occurring in the inference procedures.

In a somewhat different vein, Cano et al. (2000) use trees to represent the potentials in a join-tree representation, and show how to adapt the standard propagation algorithm to this representation. In order to make sure that trees generated during the inference process remain small, it is suggested to use trees that, where necessary, only approximate the true potentials, which makes this a framework for approximate inference.

Another method related to PDG representations are the polynomial representations of Darwiche (2000, 2002). In this framework the joint distribution of variables $X_1, \ldots, X_n$ is represented by a multilinear polynomial in indicator variables λ_{X_i} and numerical constants. This polynomial, in turn, can be represented by an arithmetic circuit whose size is linear in the size of a junction-tree representation for the distribution. Probabilistic inference now is linear in the size of the arithmetic circuit. Though these latter results bear some resemblance to our results on PDGs, there is a fundamental difference between Darwiche's approach and ours. Unlike PDGs, arithmetic circuits do not provide a primary representation language for probability distributions: there is no syntactic criterion that can be applied to tell whether any given arithmetic circuit represents a probability distribution or some other real-valued function. For this reason it is all but impossible to directly specify a probability distribution as a circuit, or to learn a circuit representation from data. This makes this form of representation a secondary representation that can only be obtained by compilation of some other representation like a Bayesian network (as proposed by Darwiche (2000)) – or a PDG.

7 Conclusion

PDGs are a promising alternative to Bayesian networks and/or junction trees for the representation of probability distributions. An attractive feature of PDGs is that they replace by one coherent, simple framework a number of modeling techniques (use of hidden variables, context-specific independence, structured representation of conditional probability tables) previously used to make Bayesian network representations more efficient. The algorithms for probabilistic inference on PDGs (computation of in- and out-flow, multiplication of RFGs) are very simple and extend the special inference methods proposed by Nielsen et al. (2000) for deterministic subnetworks to general probabilistic inference.

Different direct construction methods for PDGs are available, so that PDGs are a "stand-alone" representation framework that is not dependent on compilation from some other representation.

Eventually, a more comprehensive evaluation of the significance and applicability of PDGs will have to come out of a better understanding where the partition-based factorizations (2) encoded by PDGs are more natural and compact than the variable-based factorizations encoded by Bayesian networks.

References

Billingsley, P. (1986), *Probability and Measure*, Wiley.

Boutilier, C., Friedman, N., Goldszmidt, M. & Koller, D. (1996), Context-specific independence in Bayesian networks, *in* 'Proceedings of UAI–96', Portland, Oregon, pp. 115–123.

Bozga, M. & Maler, O. (1999), On the representation of probabilities over structured domains, *in* 'Proceedings of CAV-99', number 1633 *in* 'Lecture Notes in Computer Science'.

Bryant, R. E. (1986), 'Graph-based algorithms for boolean function manipulation', *IEEE Transactions on Computers* **35**(8), 677–691.

Cano, A., Moral, S. & Salmeron, A. (2000), 'Penniless propagation in join trees', *International Journal of Intelligent Systems* **15**(11), 1027–1059.

Cooper, G. F. (1990), 'The computational complexity of probabilistic inference using bayesian belief networks', *Artificial Intelligence* **42**, 393–405.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. & Spiegelhalter, D. J. (1999), *Probabilistic Networks and Expert Systems*, Springer.

Darwiche, A. (2000), A differential approach to inference in bayesian networks, *in* 'Proceedings of UAI–2000'.

Darwiche, A. (2002), A logical approach to factoring belief networks, *in* 'Proceedings of KR-2002'.

Fujita, M., McGeer, P. C. & Yang, J.-Y. (1997), 'Multi-terminal binary decision diagrams: an efficient data structure for matrix representation', *Formal Methods in System Design* **10**, 149–169.

Jensen, F. (2001), *Bayesian Networks and Decision Graphs*, Springer.

Lai, Y.-T. & Sastry, S. (1992), Edge-valued binary decision diagrams for multi-level hierarchical verification, *in* 'Proceedings of the 29th ACM/IEEE Design Automation Conference', pp. 608–613.

Nielsen, T. D., Wuillemin, P.-H., Jensen, F. V. & Kjærulff, U. (2000), Using ROBDDs for inference in Bayesian networks with troubleshooting as an example, *in* 'Proceedings of UAI–2000'.

Zhang, N. L. & Poole, D. (1999), On the role of context-specific independence in probabilistic inference, *in* 'Proceedings of IJCAI-99', pp. 1288–1293.