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

MANFRED JAEGER
Aalborg Universitet, Institut for Datalogi
Fredrik Bajers Vej 7, 9220 Aalborg Ø, Denmark
jaeger@cs.auc.dk

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 a representation framework for probability distributions, especially 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. They have also been adopted for the representation of real-valued functions in general (Fujita, McGeer & Yang 1997, Lai & Sastry 1992), and probability distributions in particular (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 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 introduce a generalization of Bozga and Maler's (1999) Probabilistic Decision Graphs, and then investigate the following questions: how are the probabilistic (or statistical) models encoded by the structure of a probabilistic decision graph characterized in terms of independence relations (section 3)? How can one perform basic probabilistic inference tasks based on this representation, and what is the complexity (section 4)? In answer to the last question we find that basic inference problems have linear time complexity. In order to evaluate the performance of probabilistic decision graphs in comparison to Bayesian networks, we therefore investigate in section 5 how large a representation of a distribution as a probabilistic decision graph will be in comparison to a representation as a junction tree. 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.

Throughout the paper we motivate our results by comparisons and analogies with Bayesian networks. Basic concepts relating to Bayesian networks are used without further explanations; for these the reader is referred to (Jensen 2001) and (Cowell, Dawid, Lauritzen & Spiegelhalter 1999). Readers not familiar with Bayesian networks will still be able to follow most of the technical material on probabilistic decision graphs, but may find it harder to understand the motivation of the particular questions here investigated.

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. An element $\boldsymbol{x} \in W$ is sometimes called an instantiation of the variables $\boldsymbol{X}$. If $\boldsymbol{Y} \subseteq \boldsymbol{X}$ then $W[\boldsymbol{Y}]$ denotes the factor $\times_{X \in \boldsymbol{Y}} R(X)$ of W, and $\boldsymbol{x}[\boldsymbol{Y}]$ denotes the projection of $\boldsymbol{x} \in W$ onto $W[\boldsymbol{Y}]$ (in other words, $\boldsymbol{x}[\boldsymbol{Y}]$ is the instantiation of the subset $\boldsymbol{Y}$ of variables according to the full instantiation $\boldsymbol{x}$).

Our primary interest is in representations for the joint distribution of $\boldsymbol{X}$, i.e. probability distributions on W. The following definition provides a somewhat more general framework for the representation of arbitrary real-valued functions on W. Figure 1 illustrates the definitions.

Definition 2.1 Let $F = \{T_1, \ldots, T_k\}$ be a forest over $\boldsymbol{X}$, i.e. each T_j is a rooted, directed tree whose nodes are a subset of $\boldsymbol{X}$, and the union of all nodes in the T_j is $\boldsymbol{X}$. Let E_F denote the edge relation in F. A *real function graph structure* for $\boldsymbol{X}$ with respect to the forest F is a rooted directed acyclic graph $G = (V, E)$, such that

- each node $\nu \in V$ is labeled with a variable $X_i \in \boldsymbol{X}$.
- For each node ν labeled with X_i, each $x_{i,h} \in R(X_i)$, and each $X_j \in \boldsymbol{X}$ with $(X_i, X_j) \in E_F$ 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.

A real function graph structure is turned into a *real function graph (RFG)* if

- each node ν labeled with X_i also is labeled with a *value vector* vector $\boldsymbol{p}^\nu = (p^\nu_1, \ldots, p^\nu_{k_i}) \in \mathbb{R}^{k_i}$ $(i = 1, \ldots, n)$.

We denote the resulting RFG with $G = (V, E, \boldsymbol{p})$. A RFG G is called a *probabilistic decision graph (PDG)* if for all nodes ν with label X_i: $p^\nu_h \in [0, 1]$ and $\sum_{h=1}^{k_i} p^\nu_h = 1$.

Figure 1 shows at the top a forest for variables $X_1, \ldots, X_6$. All variables here are binary, i.e. $R(X_i) = \{0, 1\}$. Below, a PDG with respect to F is shown. The labeling of the nodes $\nu_1, \ldots, \nu_{11}$ with the variables $\boldsymbol{X}$ is shown by indicating the sets V_i of nodes labeled with X_i. The labeling of the edges with the elements of $R(X_i)$ is indicated by dotted lines for label 0, and solid lines for label 1.

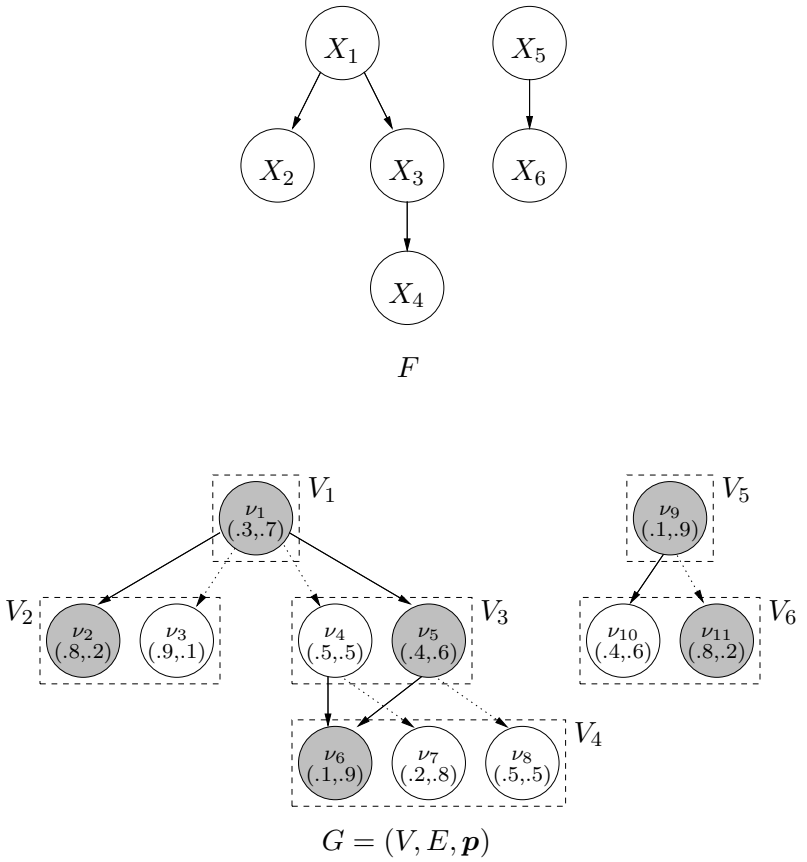

Figure 1: Probabilistic Decision Graph with underlying forest and nodes reached by $\boldsymbol{x} = (1, 0, 1, 1, 0, 0)$

A note on terminology here may be useful: the term probabilistic decision graph as we use it is adopted from (Bozga & Maler 1999). The same term had previously been used by Koenig and Simmons (1994) for a graphical representation of a sequential decision process. Rather than trying to resolve this clash of terminology by introducing yet another term, we here simply emphasize that these two usages are independent, and that our PDGs represent probability distributions, not decision models.

We have to introduce some additional notation for graph-theoretical concepts that are important for RFGs. With respect to the forest F we define for $X_i \in \boldsymbol{X}$: $succ_F(X_i) := \{X_j \in \boldsymbol{X} \mid (X_i, X_j) \in E_F\}$, $succ^*_F(X_i) := \{X_j \in \boldsymbol{X} \mid (X_i, X_j) \in E^*_F\}$, where E^*_F is the reflexive and transitive closure of E_F.

Also we denote with $pred_F(X_i)$ the unique immediate predecessor of X_i in F ($pred_F(X_i) = \emptyset$ if X_i is a root), and with $pred^*_F(X_i)$ the set of variables on the path from the root of the tree containing X_i to X_i (excluding X_i).

When $X_j \in succ_F(X_i)$, $\nu \in V_i$, and $1 \leq h \leq k_i$, then we denote with $succ(\nu, X_j, x_{i,h})$ the node $\nu' \in V_j$ that is connected to ν via the (unique) edge $(\nu, \nu') \in E$ labeled with $x_{i,h}$. In figure 1, for example, $succ(\nu_1, X_2, 0) = \nu_3$, and $succ(\nu_4, X_4, 1) = succ(\nu_5, X_4, 1) = \nu_6$.

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. F, $\nu \in V_i$. Let $succ_F(X_i) = \{Y_1, \ldots, Y_l\} \subseteq \boldsymbol{X}$. A real-valued function f^ν_G is defined on $W[succ^*_F(X_i)]$ by

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

($x_{i,h} \in R(X_i), \boldsymbol{z}_j \in W[succ^*_F(Y_j)]$). From the functions defined at roots of (G, E), we obtain a function on W:

$$f_G := \prod_{\nu:\ \nu \text{ root}} f^\nu_G \tag{2}$$

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, $f^{\nu_5}_G(X_3 = 1, X_4 = 1) = 0.6 \cdot 0.9$, $f^{\nu_1}_G(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 1) = 0.7 \cdot 0.8 \cdot 0.6 \cdot 0.9 = 0.3024$, $f^{\nu_9}_G(X_5 = 0, X_6 = 0) = 0.1 \cdot 0.8 = 0.08$, and thus $f_G((1, 0, 1, 1, 0, 0)) = 0.3024 \cdot 0.08$.

In the remainder of this section we provide some alternative characterizations and basic properties of the function f_G defined by G. A central notion in the investigation of RFGs is that of a *path*, which we develop in the following two definitions.

Definition 2.3 Let $\boldsymbol{x} \in W$, $\nu \in V$. We define inductively: $\boldsymbol{x}$ *reaches* ν if

- ν is a root, or
- $\nu \in V_i$, $X_i \in succ_F(X_j)$, $\boldsymbol{x}$ reaches $\nu' \in V_j$, and $\nu = succ(\nu', X_i, \boldsymbol{x}[X_j])$.

In figure 1 the nodes reached by $\boldsymbol{x} = (1, 0, 1, 1, 0, 0)$ are shaded. Whether $\boldsymbol{x} \in W$ reaches some $\nu \in V_i$ only depends on $\boldsymbol{x}[\boldsymbol{Y}]$, where $\boldsymbol{Y}$ is any subset of $\boldsymbol{X}$ containing $pred_F^*(X_i)$. For any $\boldsymbol{y} \in W[\boldsymbol{Y}]$ we can therefore say that $\boldsymbol{y}$ reaches (or does not reach) ν, and define:

Definition 2.4 Let $\nu \in V_i$, $pred_F^*(X_i) \subseteq \boldsymbol{Y} \subseteq \boldsymbol{X}$. Then

$$Path(\nu, \boldsymbol{Y}) := \{\boldsymbol{y} \in W[\boldsymbol{Y}] \mid \boldsymbol{y} \text{ reaches } \nu\}. \tag{3}$$

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

We can now also characterize the function f_G as follows.

Proposition 2.5 *(A)* For all $\boldsymbol{x} \in W$ and all $i \in 1, \ldots, n$ there exists exactly one $\nu_i(\boldsymbol{x}) \in V_i$ that is reached by $\boldsymbol{x}$, and

$$f_G(\boldsymbol{x}) = \prod_{i=1}^{n} p_{ind(i,\boldsymbol{x})}^{\nu_i(\boldsymbol{x})}, \tag{4}$$

where $ind(i, \boldsymbol{x})$ is the index in $R(X_i)$ of $\boldsymbol{x}[X_i]$.

(B) Let $G \setminus X_i$ denote the RFG obtained from G by removing all nodes labeled with some $X_j \in succ_F^*(X_i)$. For any $\nu \in V_i$, and any $\boldsymbol{x} \in Path(\nu, \boldsymbol{X})$ then

$$f_G(\boldsymbol{x}) = f_{G \setminus X_i}(\boldsymbol{x}[\boldsymbol{X} \setminus succ_F^*(X_i)]) f_G^{\nu}(\boldsymbol{x}[succ_F^*(X_i)]). \tag{5}$$

3 Independence Structure and Canonicity

The structure of a PDG encodes certain (conditional) independence relations in the distribution P_G. In this section we characterize these independence relations as conditional independencies of distributions given partitions of the state space W. This is a generalization of the type of conditional independence relations encoded in a Bayesian network structure, which can be characterized by conditional independencies given sets of variables. We will then show that for a fixed forest F, every probability distribution P that can be represented by a PDG over F has a canonical normal form PDG-representation. From this we can then derive that different PDG structures (for a given F) encode different probabilistic models, i.e. the sets of probability distributions that can be represented over the structures are different.

We begin with briefly reviewing the relevant definitions about conditional distributions given partitions of the state space. For details the reader is referred to (Billingsley 1986, Section 33).

Definition 3.1 Let $\mathscr{A} = \{A_1, \ldots, A_k\}$ be a partition of W, $i \in \{1, \ldots, n\}$, and P a probability distribution on W. The conditional probability distribution of X_i given A_j $(j = 1, \ldots, k)$ is the function on $R(X_i)$ defined by

$$P(X_i \mid A_j)(x_{i,h}) := P(X_i = x_{i,h} \mid A_j). \tag{6}$$

The conditional probability distribution of X_i given $\mathscr{A}$ is the function on $R(X_i) \times W$ defined by

$$P(X_i \mid \mathscr{A})(x_{i,h}, \boldsymbol{x}) := \sum_{j=1}^{k} P(X_i = x_{i,h} \mid A_j) 1_{A_j}(\boldsymbol{x}), \tag{7}$$

where 1_{A_j} is the indicator of A_j, i.e. $1_{A_j}(\boldsymbol{x}) = 1$ iff $\boldsymbol{x} \in A_j$, and $1_{A_j}(\boldsymbol{x}) = 0$ otherwise.

A subset $\boldsymbol{Y} \subseteq \boldsymbol{X}$ of variables generates the partition

$$\mathscr{A}(\boldsymbol{Y}) := \{\{\boldsymbol{x} \in W \mid \boldsymbol{x}[\boldsymbol{Y}] = \boldsymbol{y}\} \mid \boldsymbol{y} \in W[\boldsymbol{Y}]\}. \tag{8}$$

Then $P(X_i \mid \mathscr{A}(\boldsymbol{Y}))$ is the usual conditional distribution of X_i given $\boldsymbol{Y}$, for which we also use the simpler notation $P(X_i \mid \boldsymbol{Y})$.

Given two partitions $\mathscr{B}, \mathscr{C}$ we denote with $\mathscr{A}(\mathscr{B}, \mathscr{C})$ the partition generated by $\mathscr{B}, \mathscr{C}$, i.e. the partition consisting of the sets of the form $B \cap C$ with $B \in \mathscr{B}$ and $C \in \mathscr{C}$. We say that X_i is *conditionally independent from $\mathscr{C}$ given $\mathscr{B}$* if

$$P(X_i \mid \mathscr{A}(\mathscr{B}, \mathscr{C})) = P(X_i \mid \mathscr{B}). \tag{9}$$

Every node ν in a PDG defines a two-way partition of W into $Path(\nu, \boldsymbol{X})$ and its complement. We can now properly identify the value vectors $\boldsymbol{p}^\nu$ and the local functions P_G^ν as conditional probability distributions:

Proposition 3.2 Let $\nu \in V_i$, $\boldsymbol{Y} = pred_F^*(X_i)$. Then for all $\boldsymbol{y} \in Path(\nu, \boldsymbol{Y})$:

$$\boldsymbol{p}^\nu = P_G(X_i \mid \boldsymbol{Y} = \boldsymbol{y}) = P_G(X_i \mid Path(\nu, \boldsymbol{Y})) \tag{10}$$

$$P_G^\nu = P_G(succ_F^*(X_i) \mid \boldsymbol{Y} = \boldsymbol{y}) = P_G(succ_F^*(X_i) \mid Path(\nu, \boldsymbol{Y})). \tag{11}$$

A subset $U \subseteq V$ defines the partition generated by all the two-way partitions given by the $\nu \in U$. We denote this partition with $\mathscr{A}(U)$. Of particular interest is the case $U = V_i$. The partition $\mathscr{A}(V_i)$ simply is the partition of W into the sets $\{\boldsymbol{x} \mid \boldsymbol{x}$ reaches $\nu\}$ $(\nu \in V_i)$. Using these partitions, we can express the representation of a probability distribution by a PDG as a factorization in a manner that parallels the factorized representation $P = P(X_i \mid Pa(X_i))$ of a distribution in a Bayesian network (see e.g. (Jensen 2001, Section 1.4)).

Proposition 3.3 The probability distribution P_G represented by a PDG G satisfies the conditional independence relations

$$P_G(X_i \mid \boldsymbol{X} \setminus succ_F^*(X_i)) = P_G(X_i \mid pred_F^*(X_i)) = P_G(X_i \mid \mathscr{A}(V_i)), \quad (12)$$

and therefore satisfies the factorization

$$P_G(\boldsymbol{x}) = \prod_{i=1}^{n} P_G(X_i \mid \mathscr{A}(V_i))(\boldsymbol{x}[X_i], \boldsymbol{x}). \quad (13)$$

We also obtain the following converse statement:

Proposition 3.4 A probability distribution P can be represented as a PDG over a given PDG structure (V, E) if P satisfies the conditional independence relations (12) for $i = 1, \ldots, n$.

For the PDG in figure 1 we obtain as an instance of (12) that $P_G(X_4 \mid X_1, X_3) = P_G(X_4 \mid \mathscr{A}(V_4))$, which expresses equality of the conditional distributions given the two partitions $\mathscr{B} := \{\{X_3 = 0, X_1 = 0\}, \{X_3 = 0, X_1 = 1\}, \{X_3 = 1, X_1 = 0\}, \{X_3 = 1, X_1 = 1\}\}$, and $\mathscr{C} := \{\{X_3 = 1\}, \{X_3 = 0, X_1 = 0\}, \{X_3 = 0, X_1 = 1\}\}$. This, in turn, can be written as the "context specific" independence

$$P_G(X_4 \mid X_3 = 1, X_1) = P_G(X_4 \mid X_3 = 1). \quad (14)$$

The independence relation (14) can not be represented by the structure of a Bayesian network. On the other hand, there are conditional independence relations between sets of random variables that can be expressed in a Bayesian network structure, but cannot be captured in a PDG. For example, one can show (by an exhaustive search over candidate PDG structures) that no PDG can capture the conditional independence relations between variables X_1, X_2, X_3, X_4 that are encoded in a "diamond" Bayesian network structure with edges (X_1, X_2), (X_1, X_3), (X_2, X_4), (X_3, X_4). The intuitive (albeit not fully correct) explanation for this is that PDG structures encode 'variable-level' independence relations only through certain separation properties in the underlying forest structure, and that the class of separation relations on forests is less rich than the class of d-separation relations (Pearl 1988) on directed acyclic graphs. It follows that PDGs and Bayesian network are incomparable with respect to their expressiveness for conditional independence relations.

Apart from the basic conditional independence relations (12) a PDG structure will usually encode many further conditional independence relations. In the case of Bayesian networks, the d-separation criterion provides a

complete characterization of conditional independence relations implied by a network structure. It is an open problem whether for PDGs one can also find suitable (graph-theoretic) characterizations of all implied conditional independencies. The problem is made more difficult by the fact that in the context of PDGs one considers conditional independencies of the form (9) expressed in terms of partitions. As we have seen, there are various ways to define partitions (by sets of variables, sets of PDG-nodes,...), and before one can derive criteria for the derivation of independencies (9) one has to establish a representation language for partitions. It may very well turn out, therefore, that one can derive for certain restricted representation languages for partitions complete characterizations for conditional independencies in PDGs, but no single such characterization for a language that can represent every partition.

One of the main motivations behind the development of OBDDs is their canonicity: given a boolean function f and a fixed order on the variables, there is a unique canonical representation of f by a (reduced) OBDD (Bryant 1986). A related question has received some attention in the Bayesian network literature: given an independence model (i.e. a certain set of conditional independencies), is there a unique Bayesian network structure representing this model? The answer usually is no. This has given rise to the study of equivalence classes of Bayesian network structures and their representation by *essential graphs* (Verma & Pearl 1991, Andersson, Madigan & Perlman 1997).

We will now investigate the corresponding questions for PDGs. First it will be shown that, unsurprisingly, the canonicity property of OBDDs carries over to PDGs: for a fixed forest F there is a unique canonical representation of each distribution P that can be represented over F. We will then show that this implies that different PDG-structures for the same forest F encode distinct independence models.

To simplify matters, we only consider in the following representations of *strictly positive* probability distributions P, i.e. $P(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in W$. Distributions that do not have this property permit PDG-representations with arbitrary subgraphs rooted at nodes that are not reachable by any $\boldsymbol{x}$ of positive probability. To obtain normal forms for their representation one therefore has to use suitable additional conventions for the representation of such subgraphs.

The following definition is standard for OBDD-related frameworks.

Definition 3.5 Let $G = (V, E, \boldsymbol{p})$ be a PDG, $i \in \{1, \ldots, n\}$. Two nodes $\nu, \nu' \in V_i$ are *equivalent*, denoted $\nu \sim \nu'$, if $\boldsymbol{p}^{\nu} = \boldsymbol{p}^{\nu'}$, and $succ(\nu, X_j, x_{i,h}) \sim$

$succ(\nu', X_j, x_{i,h})$ for all $X_j \in succ_F(X_i)$ and $h = 1, \ldots, k_i$.

The definition of $\sim$ gives rise to a simple recursive decision procedure for $\sim$. We note in the following proposition that $\sim$ also has a very natural semantic interpretation.

Proposition 3.6 Let P_G be strictly positive. Then for all $i = 1, \ldots, n$; $\nu, \nu' \in V_i$: $\nu \sim \nu'$ iff $P_G^{\nu} = P_G^{\nu'}$.

By recursively merging nodes ν, ν' with $\nu \sim \nu'$ (starting with leaf nodes, where $\nu \sim \nu'$ simply means $\boldsymbol{p}^{\nu} = \boldsymbol{p}^{\nu'}$) one can transform in quadratic time a given PDG into an equivalent one that is reduced in the following sense (again a standard definition for OBDD-related frameworks):

Definition 3.7 A PDG is called *reduced* if there do not exist distinct nodes ν, ν' with $\nu \sim \nu'$.

Theorem 3.8 Let F be a given forest. Let P be a strictly positive probability distribution that is representable by a PDG over F. Then there exists exactly one reduced PDG over F representing P.

Proof: To simplify notation, we assume that all random variables are binary, and that F consists of a single tree that is the linear order $X_1, \ldots, X_n$ (the general case then follows by applying this case to all linear branches in F). Let P be given, and let $i \in 1, \ldots, n$. Define an equivalence relation $\approx$ for $\boldsymbol{y}, \boldsymbol{y}' \in W[X_1, \ldots, X_{i-1}]$ by: $\boldsymbol{y} \approx \boldsymbol{y}'$ iff

$$P(X_i, \ldots, X_n \mid (X_1, \ldots, X_{i-1}) = \boldsymbol{y}) = P(X_i, \ldots, X_n \mid (X_1, \ldots, X_{i-1}) = \boldsymbol{y}'). \tag{15}$$

From the strict positivity of P it follows that the two conditional probability distributions in (15) are well-defined, and hence the relation $\approx$ is well-defined. Let $[\boldsymbol{y}]$ denote the equivalence class of $\boldsymbol{y}$.

Now consider a reduced PDG $G = (V, E, \boldsymbol{p})$ for P. From (15),(11) and proposition 3.6 it follows that there is a 1-to-1 correspondence between the nodes in V_i and the equivalence classes $[\boldsymbol{y}]$: for all $\nu \in V_i$ we have $Path(\nu, \{X_1, \ldots, X_{i-1}\}) = [\boldsymbol{y}]$ for some $\boldsymbol{y}$. The equivalence relation $\sim$ on $W[X_1, \ldots, X_{i-1}]$ therefore determines the set V_i, and the partition of $W[X_1, \ldots, X_{i-1}]$ into the sets $\{Path(\nu, \{X_1, \ldots, X_{i-1}\}) \mid \nu \in V_i\}$. These partitions for $i = 1, \ldots, n$, in turn completely determine the structure (V, E). Finally, for any $\nu \in V_i$ the vector $\boldsymbol{p}^{\nu}$ is uniquely determined by P through (10). $\square$

As a consequence we obtain:

Theorem 3.9 Let $G = (V, E), G' = (V', E')$ be two distinct PDG structures over the same forest F. Let $\mathcal{P}_G$ and $\mathcal{P}_{G'}$ denote the set of probability distributions representable over G and G', respectively. Then $\mathcal{P}_G \neq \mathcal{P}_{G'}$.

Proof: Assume without loss of generality that $|\,V\,| \geq |\,V'\,|$. Let $\boldsymbol{p} = \{\boldsymbol{p}^\nu \mid \nu \in V\}$ be a set of value vectors for G, such that all components p_h^ν are distinct ($\nu \in V_i,\ 1 \leq h \leq k_i, i = 1, \ldots, n$). The PDG $(V, E, \boldsymbol{p})$ then is reduced. Let P_G be the probability distribution represented by $(V, E, \boldsymbol{p})$, and assume that P_G also is representable over (V', E') with parameters $\boldsymbol{p'}$. From theorem 3.8 it follows that $(V', E', \boldsymbol{p'})$ is not reduced. Turning $(V', E', \boldsymbol{p'})$ into a reduced PDG by merging states yields a reduced PDG $(V'', E'', \boldsymbol{p''})$ for P_G with $|\,V''\,| < |\,V\,|$, in contradiction to theorem 3.8. □

Theorem 3.9 does not extend to PDG-structures over different forests. For instance, two different linear orders of $\boldsymbol{X}$ are underlying forests for two distinct complete trees for $\boldsymbol{X}$, each of which can represent every probability distribution.

4 Probabilistic Inference

We now turn to the question of how to compute conditional probability distributions of a variable $X \in \boldsymbol{X}$ given the values $\boldsymbol{y}$ of a subset of observed variables $\boldsymbol{Y} \subset \boldsymbol{X}$. Central to the solution of this problem are the concepts of the *in-flow* and *out-flow* of a node.

Definition 4.1 Let $G = (V, E, \boldsymbol{p})$ be an RFG for $\boldsymbol{X}$ with respect to F. Let $\nu \in V_i$. The out-flow of ν is defined as

$$\mathit{ofl}(\nu) := \sum_{\boldsymbol{z} \in W[\mathit{succ}_F^*(X_i)]} f_G^\nu(\boldsymbol{z}). \tag{16}$$

Thus, the outflow of ν is just the sum of all values of f_G^ν.

Definition 4.2 Let G and F be as in Definition 4.1. Let $\nu \in V_i$, and let $G \setminus X_i$ as in proposition 2.5 (B). The in-flow of ν is defined as

$$\mathit{ifl}(\nu) := \sum_{\boldsymbol{y} \in \mathit{Path}(\nu, \boldsymbol{X} \setminus \mathit{succ}_F^*(X_i))} f_{G \setminus X_i}(\boldsymbol{y}). \tag{17}$$

If $\boldsymbol{X} \setminus \mathit{succ}_F^*(X_i)$ is empty (which happens exactly when F consists of a single tree with root X_i) define $\mathit{ifl}(\nu) := 1$.

From proposition 2.5 (B) it follows that

$$\mathit{ifl}(\nu)\mathit{ofl}(\nu) = \sum_{\boldsymbol{x}\in Path(\nu,\boldsymbol{X})} f_G(\boldsymbol{x}). \tag{18}$$

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

Lemma 4.3 (a) Let $\nu \in V_i$. Then

$$\mathit{ofl}(\nu) = \sum_{h=1}^{k_i} p_h^\nu \prod_{Y\in succ_F(X_i)} \mathit{ofl}(succ(\nu, Y, x_{i,h})). \tag{19}$$

(b) Let $\nu \in V_i$, where X_i is a root of F. Then

$$\mathit{ifl}(\nu) = \prod_{\nu'\neq\nu:\ \nu' \text{ root}} \mathit{ofl}(\nu') \tag{20}$$

(c) Let $\nu \in V_i$, where X_i is not a root of F. Assume that $pred_F(X_i) = X_j$. Then

$$\mathit{ifl}(\nu) = \sum_{h=1}^{k_j} \sum_{\substack{\nu'\in V_j:\\ \nu=succ(\nu',X_i,x_{j,h})}} [\mathit{ifl}(\nu')p_h^{\nu'} \prod_{Y\in succ_F(X_j)\setminus X_i} \mathit{ofl}(succ(\nu', Y, x_{j,h}))]. \tag{21}$$

Proof: (a) follows directly by summation of (1) over all $x_{i,h}$ and $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l$.

(b): For X_i a root we have $Path(\nu, \boldsymbol{X} \setminus succ_F^*(X_i)) = W[\boldsymbol{X} \setminus succ_F^*(X_i)]$, and therefore (20) follows from (2) and (16).

For (c), let ν, V_i be given. Let $\boldsymbol{Y} := \boldsymbol{X} \setminus succ_F^*(X_i)$. We partition $Path(\nu, \boldsymbol{Y})$ into disjoint subsets according to the X_j-component of $\boldsymbol{y} \in Path(\nu, \boldsymbol{Y})$, and according to the nodes reached by $\boldsymbol{y}$ in V_j:

$$Path(\nu, \boldsymbol{Y}) = \bigcup_{h=1}^{k_j} \bigcup_{\substack{\nu'\in V_j\\ \nu=succ(\nu',X_i,x_{j,h})}} Path(\nu', \boldsymbol{Y} \setminus X_j) \times \{x_{j,h}\}. \tag{22}$$

Denote $\boldsymbol{Z} := succ_F^*(X_j) \setminus \{succ_F^*(X_i) \cup \{X_j\}\}$. With

$$Path(\nu', \boldsymbol{Y} \setminus X_j) = Path(\nu', \boldsymbol{Y} \setminus succ_F^*(X_j)) \times W[\boldsymbol{Z}] \tag{23}$$

and (from (5)) for $\boldsymbol{y} \in Path(\nu', \boldsymbol{Y} \setminus X_j)$:

$$f_{G\setminus X_i}(\boldsymbol{y}, x_{j,h}) = f_{(G\setminus X_i)\setminus X_j}(\boldsymbol{y}[\boldsymbol{Y} \setminus succ_F^*(X_j)]) f^{\nu'}_{G\setminus X_i}(\boldsymbol{y}[\boldsymbol{Z}], x_{j,h}) \qquad (24)$$

we obtain

$$\begin{aligned}
&\sum_{\boldsymbol{y}\in Path(\nu',\boldsymbol{Y}\setminus X_j)} f_{G\setminus X_i}(\boldsymbol{y}, x_{j,h}) \\
&\quad = \sum_{\boldsymbol{u}\in Path(\nu',\boldsymbol{Y}\setminus succ_F^*(X_j))} f_{(G\setminus X_i)\setminus X_j}(\boldsymbol{u}) \sum_{\boldsymbol{w}\in W[\boldsymbol{Z}]} f^{\nu'}_{G\setminus X_i}(\boldsymbol{w}, x_{j,h}) \\
&\quad = ifl(\nu') \sum_{\boldsymbol{w}\in W[\boldsymbol{Z}]} p_h^{\nu'} \prod_{Y\in succ_F(X_j)\setminus X_i} f_G^{succ(\nu',Y,x_{j,h})}(\boldsymbol{w}[succ_F^*(Y)]) \\
&\quad = ifl(\nu') p_h^{\nu'} \prod_{Y\in succ_F(X_j)\setminus X_i} ofl(succ(\nu', Y, x_{j,h})). \qquad (25)
\end{aligned}$$

The last equality is obtained by splitting the summation over $\boldsymbol{w} \in W[\boldsymbol{Z}]$ into individual summations over the factors $W[succ_F^*(Y)]$ ($Y \in succ_F(X_j) \setminus X_i$), and distributing the summations into the product. Together with (22) this proves (c). □

Equations (19) - (21) give rise to a simple procedure for computing $ifl(\nu)$ and $ofl(\nu)$ for all nodes ν in time linear in the size of G. We give a time bound in terms of the size $|\,G\,| := \sum_{i=1}^{n} |\,V_i\,|\; k_i$ of G. Note that by this definition we identify the size of G with the number of parameters in G (which also is an upper bound for the number of edges in E). In particular, we assume a fixed representation size for the parameters, so that considerations about complexity dependence on the length of the numerical representations do not enter the picture.

Theorem 4.4 Given a RFG $G = (V, E, \boldsymbol{p})$ one can compute in $O(|\,G\,|)$ time the values $ofl(\nu)$ and $ifl(\nu)$ for all $\nu \in V$.

Proof: One first computes $ofl(\nu)$ for all ν. This is done recursively starting with the leaf nodes ν, for which $ofl(\nu) = \sum_{h=1}^{k_i} p_h^{\nu}$. Given that $ofl(\nu')$ is already computed for all successors ν' of an interior node ν, $ofl(\nu)$ can be computed using (19). This computation is linear in the number of outgoing edges of ν, and so the computation for all ν is linear in the size of G. Apart from the ofl values we also store at each node the values $\pi_h(\nu) := \prod_{Y\in succ_F(X_i)} ofl(succ(\nu, Y, x_{i,h}))$ ($h = 1, \ldots, k_i$). This comes at no extra cost, as the π_h are just intermediate results in the computation of ofl.

In a second pass, $ifl(\nu)$ is computed for all ν. This computation starts at root nodes ν, for which $ifl(\nu)$ is computed with (20). Given that $ifl(\nu')$ has been computed for all predecessors ν' of an interior node ν, $ifl(\nu)$ is computed using (21). The double summation of (21) amounts to a summation over all incoming edges of ν, so the overall complexity will be linear, provided each term in the sum can be computed in constant time. This is done using the auxiliary values π_h computed in the first phase, which allow us to rewrite the terms in the sum (21) as $ifl(\nu')p_h^{\nu'}\pi_h(\nu')/ofl(\nu)$.

□

Based on computations of *ifl* and *ofl* probabilistic queries can be answered. 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 $\boldsymbol{y} \in W[\boldsymbol{Y}]$. 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_h^ν to 0 if $\boldsymbol{y}[X_j] \neq x_{j,h}$. For $\boldsymbol{x} \in W$ we then have

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

It follows that $P_G(\boldsymbol{Y} = \boldsymbol{y})$ is equal to $\prod ofl(\nu')$, where the product is over roots ν' of $G_{\boldsymbol{Y}=\boldsymbol{y}}$. As $G_{\boldsymbol{Y}=\boldsymbol{y}}$ and the out-flow of its roots 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

$$P_G(X_j = x_{j,h} \mid \boldsymbol{Y} = \boldsymbol{y}) = \frac{1}{P_G(\boldsymbol{Y} = \boldsymbol{y})} \sum_{\nu \in V_j} ifl(\nu) p_h^\nu \prod_{Y \in succ_F(X_j)} ofl(succ(\nu, Y, x_{j,h})) \tag{26}$$

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

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 $P_G(X_j \mid \boldsymbol{Y} = \boldsymbol{y})$ can be extended in a very coherent way to compute posterior distributions $P_G(X_j \mid \mathcal{E})$, 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 $P_G(X_j \mid \boldsymbol{Y} = \boldsymbol{y})$ can not always 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 $\boldsymbol{x}[\boldsymbol{Y}] = \boldsymbol{y}$, and $1_{\boldsymbol{Y}=\boldsymbol{y}}(\boldsymbol{x}) = 0$ else. To generalize our approach to the computation of posterior distributions, it is sufficient to 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 the subclass of RFGs that essentially corresponds to classical OBDDs:

Definition 4.5 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.

The computation of a RFG representation of $f_G \cdot 1_{\mathcal{E}}$, thus, is a special case of the general problem of computing for two RFGs G and H a RFG $G{\cdot}H$ representing the product $f_G \cdot f_H$. When G and H are given w.r.t. the same underlying forest F, then this problem can be solved by a simple adaption of Bryant's (1986) method for performing Boolean operations on OBDDs. A high-level description of the resulting algorithm is given in table 1. It is stated for the case that F consists of a single tree. For general F one simply applies this procedure to all pairs of corresponding trees of G. To make this 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 4.6 Let $G = (V, E_G)$ and $H = (U, E_H)$ be two RFGs for $\boldsymbol{X}$ w.r.t. the same forest F. 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 forest of Figure 1 and the instantiation $X_2 = 1, X_3 = 0, X_6 = 0$. As an indicator graph for an instantiation can be given for any forest, and such that $|V_i| = 1$ for all i, we find that the general complexity result of Theorem 4.6 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

Algorithm: *multiply-rfg*(ρ_G, ρ_H: roots of RFGs w.r.t. single-tree forest F)

i:= index of variable at the root of F
ρ := 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 $Y \in succ_F(X_i)$ **do**
        $succ(\rho, Y, x_{i,h}) :=$ *multiply-rfg*$(succ(\rho_G, Y, x_{i,h}), succ(\rho_H, Y, x_{i,h}))$;
    end
end
return ρ.

Table 1: Multiplication algorithm

reduces to a traversal of G and the multiplication with 0 of some parameters p_h^ν).

Not every set $\mathcal{E} \subseteq W$ can be represented with an indicator graph that has the same underlying forest F as a given PDG. To compute the conditional distribution of P_G given arbitrary evidence $\mathcal{E}$, one therefore may first have to determine a forest F^*, such that both P_G and $1_\mathcal{E}$ are representable over F^*, and then transform the given PDG and indicator graph into RFGs over F^*. While algorithmically not very difficult, this procedure can cause an exponential blowup in the size of the PDG and/or the indicator graph. One thus sees that conditional distributions $P_G(X_i \mid \mathcal{E})$ can be computed efficiently (in quadratic time) for evidence $\mathcal{E}$ representable over the forest F underlying G, but not for all $\mathcal{E}$. The situation, thus, is quite similar to what one finds in Bayesian networks, where one can condition on $\mathcal{E}$ efficiently if $\mathcal{E}$ is expressed as a condition on variables in a single clique of the junction tree (or a conjunction of such conditions), but $P(X_i \mid \mathcal{E})$ also cannot be computed efficiently for arbitrary $\mathcal{E}$.

5 PDGs vs. Junction Trees

In this section we compare the efficiency of probabilistic inference for PDGs with the efficiency of Bayesian network inference. For PDG based inference we have found linear time complexity in the size of the PDG. The time complexity of Bayesian network based inference is linear in the size of the junction tree into which the Bayesian network is compiled for infer-

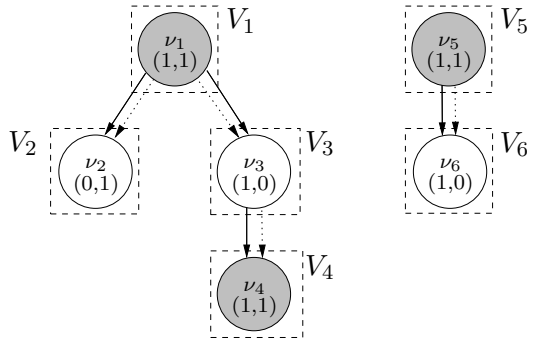

Figure 2: Indicator graph for partial instantiation

ence (inference methods that do not use junction trees have essentially the same complexity; see e.g. (Dechter 1996)). The pertinent comparison we have to make, therefore, is between the sizes of PDGs and junction tree representations of a probability distribution.

Theorem 5.1 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: We give a fairly detailed proof of this theorem here. In spite of the somewhat technical nature of the formal proof, the basic construction is very simple and fully illustrated by the example shown in figure 3.

Let J be a junction tree with nodes (cliques) $C_1, \ldots, C_K$. Thus, each C_i is a subset of $\boldsymbol{X}$, and $\cup_{i=1}^{K} C_i = \boldsymbol{X}$. Let P be the distribution of $\boldsymbol{X}$ encoded by the J.

The underlying forest of the PDG we construct consists of one tree for each connected component of J. To simplify notation we here assume that J only has one connected component.

We turn J into a directed tree by choosing an arbitrary clique C_r as the root, and directing all edges away from C_r. With every node C of J we can then associate the set $new(C) \subseteq C$ of variables from $\boldsymbol{X}$ that are not contained in the parent node $pred_J(C)$ of C. The forest F for G is now obtained from this directed junction tree by substituting for each node C_i a linear sequence of nodes, one for each $X_h \in new(C_i)$. The predecessor of the first node in the sequence is the last node of the sequence substituted for $pred_J(C_i)$. Figure 3 (i) and (ii) show a directed junction tree J, and the forest F constructed for J.

By induction on K one now shows that one can construct a PDG G w.r.t. F, 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:

$$\sum_{i:X_i \in new(C)} |V_i||R(X_i)| \;\leq\; 2 \prod_{X \in var(C)} |R(X)| \,. \tag{27}$$

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.

Let $K = 1$, i.e. J consists of a single clique C_1 with $new(C_1) = \boldsymbol{X}$. Then F is a linear order, which we may choose just to be the order $X_1, \ldots, X_n$. Define the PDG structure (V, E) to be the complete tree for the order F. Then every probability distribution can be represented over (V, E), and we have $|V_i| = \prod_{j=1}^{i-1} |R(X_j)|$, and

$$\sum_{i=1}^{n} |V_i||R(X_i)| = \sum_{i=1}^{n} \prod_{j=1}^{i} |R(X_i)| \leq\; 2 \prod_{j=1}^{n} |R(X_i)| \,. \tag{28}$$

In the complete tree (V, E) the leaf nodes $\nu \in V_n$ are in 1-to-1 correspondence with the elements of $W[X_1, \ldots, X_{n-1}]$. We generalize this property to the following invariant that will be maintained in the induction:

(I) For all $i = 1, \ldots, n$: if $X_i \in new(C)$ is the last element in the sequence substituted for C in the construction of F, then

$$V_i = \{\nu_{\boldsymbol{y}} \mid \boldsymbol{y} \in W[C \setminus X_i]\} \tag{29}$$

where $\nu_{\boldsymbol{y}}$ is such that that $Path(\nu_{\boldsymbol{y}}, \boldsymbol{X}) = \{\boldsymbol{x} \in W \mid \boldsymbol{x}[C \setminus X_i] = \boldsymbol{y}\}$.

Now let $K > 1$. Without loss of generality, assume that C_K is a leaf of J, that C_{K-1} is its parent, and that for some $1 \leq k < l \leq m < n$:

$$C_{K-1} = \{X_k, X_{k+1}, \ldots, X_m\} \quad C_K = \{X_l, X_{l+1}, \ldots, X_n\}. \tag{30}$$

Then $new(C_K) = \{X_{m+1}, \ldots, X_n\}$. Let J' be the junction tree obtained from J by removing C_K, and let G' be the PDG over a (single-tree) forest F' constructed for J' according to the induction hypothesis.

Let X_f be the last element in the linear sequence of nodes $new(C_{K-1})$ that was substituted for C_{K-1} in the construction of the forest F'. The forest

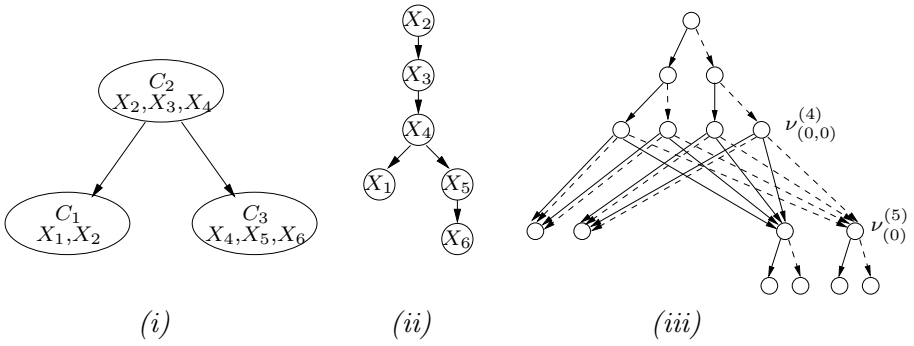

Figure 3: Construction of PDG from Junction Tree

F for J now is obtained by appending the linear sequence $X_{m+1}, \ldots, X_n$ to X_f in F'.

To define G, first define the set V_{m+1} as follows:

$$V_{m+1} := \{\nu_{\boldsymbol{y}}^{(m+1)} \mid \boldsymbol{y} \in W[X_l, \ldots, X_m]\}. \tag{31}$$

We connect each $\nu_{\boldsymbol{y}}^{(m+1)}$ to the nodes in V_f, so that

$$Path(\nu_{\boldsymbol{y}}^{(m+1)}, \boldsymbol{X}) = \{\boldsymbol{x} \in W \mid \boldsymbol{x}[X_l, \ldots, X_m] = \boldsymbol{y}\}. \tag{32}$$

For this, first observe that according to (I) there exists for all $\boldsymbol{z} \in W[\{X_k, \ldots, X_m\} \setminus X_f]$ a node $\nu_{\boldsymbol{z}}^{(f)} \in V_f$ with $Path(\nu_{\boldsymbol{z}}^{(f)}, \boldsymbol{X}) = \{\boldsymbol{x} \in W \mid \boldsymbol{x}[\{X_k, \ldots, X_m\} \setminus X_f] = \boldsymbol{z}\}$. If $f \notin \{l, \ldots, m\}$ define

$$succ(\nu_{\boldsymbol{z}}^{(f)}, X_{m+1}, x_{f,h}) := \nu_{\boldsymbol{z}[X_l,\ldots,X_m]}^{(m+1)} \quad (h = 1, \ldots, k_f). \tag{33}$$

If $f \in \{l, \ldots, m\}$ define

$$succ(\nu_{\boldsymbol{z}}^{(f)}, X_{m+1}, x_{f,h}) := \nu_{\boldsymbol{z}[X_l,\ldots,X_m],x_{f,h}}^{(m+1)} \quad (h = 1, \ldots, k_f). \tag{34}$$

Definition (33) is illustrated in figure 3 (iii) by the connections between V_4 and V_1, whereas definition (34) is illustrated in the connections between V_4 and V_5. Now (32) holds for all $\nu_{\boldsymbol{y}}^{(m+1)}$. The construction of G is completed by appending to each node in V_{m+1} a complete tree for the variables $X_{m+2}, \ldots, X_n$.

A leaf node $\nu \in V_n$ then has a unique predecessor $\nu_{\boldsymbol{y}}^{(m+1)}$ in V_{m+1} from which it is reached by a unique $\boldsymbol{z} \in W[X_{m+1}, \ldots, X_{n-1}]$. Thus

$$Path(\nu, \boldsymbol{X}) = \{\boldsymbol{x} \mid \boldsymbol{x}[X_l, \ldots, X_{n-1}] = (\boldsymbol{y}, \boldsymbol{z})\}. \tag{35}$$

Conversely, for every pair $(\boldsymbol{y}, \boldsymbol{z})$ as in (35) there exists a $\nu \in V_n$ with (35). This shows (I) for the extended graph.

The verification of the bound (27) for the extended graph is similar as in the base case $K = 1$. It thus remains to show that the distribution P represented by J can be represented over the PDG structure (V, E). By induction hypothesis, the marginal distribution $P \mid X_1, \ldots, X_m$ is representable over (G', E') with parameters $\boldsymbol{p}'$. By the semantics of join trees, $P(X_{m+1}, \ldots, X_n \mid X_1, \ldots, X_m) = P(X_{m+1}, \ldots, X_n \mid X_l, \ldots, X_m)$. For each $\boldsymbol{y} \in W[X_l, \ldots, X_m]$ one can parameterize the subtree rooted at $\nu_{\boldsymbol{y}}^{(m+1)}$ so that

$$P_G^{\nu_{\boldsymbol{y}}^{(m+1)}} = P(X_{m+1}, \ldots, X_n \mid (X_l, \ldots, X_m) = \boldsymbol{y}). \tag{36}$$

In the PDG $G = (V, E, \boldsymbol{p})$ obtained by extending the parameterization $\boldsymbol{p}'$ with these parameters then $P_G = P_G(X_{m+1}, \ldots, X_n \mid X_1, \ldots, X_m) P_G(X_1, \ldots, X_m)$ $= P_G(X_{m+1}, \ldots, X_n \mid X_l, \ldots, X_m) P_{G'}(X_1, \ldots, X_m) = P(X_{m+1}, \ldots, X_n \mid X_l, \ldots, X_m) P(X_1, \ldots, X_m) = P$.

□

Theorem 5.1 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 5.2 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 4 (a). The junction-tree constructed from this network (as well as any other junction-tree for P) is of exponential size in n. Figure 4 (b) shows the structure of a PDG representation of P, which is linear in n.

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

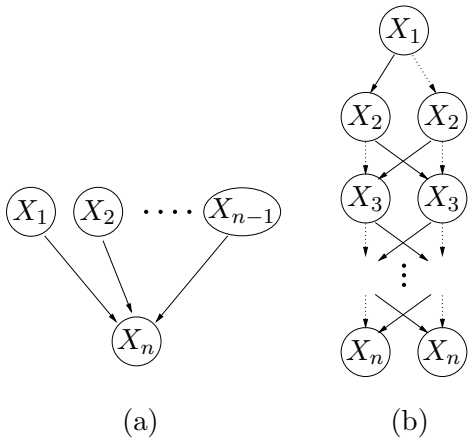

Figure 4: Bayesian network and PDG representations

$Y := \sum_{j=1}^{n-2} X_j \bmod 2$, 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 5.3 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 $G = (V, E)$, and assume that F consists of a single tree (otherwise the construction proceeds separately for each component of (V, E)). 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 $X_i = \mathit{succ}_F(X_j)$. With this network structure P_G can be encoded. The cliques of the junction tree obtained from B by the usual construction are the sets of the form $\{N_j, X_j, N_i\}$ and thus are labeled with tables of size $|\, V_j \,||\, R(X_j) \,||\, V_i \,|$. Summing these over i gives a

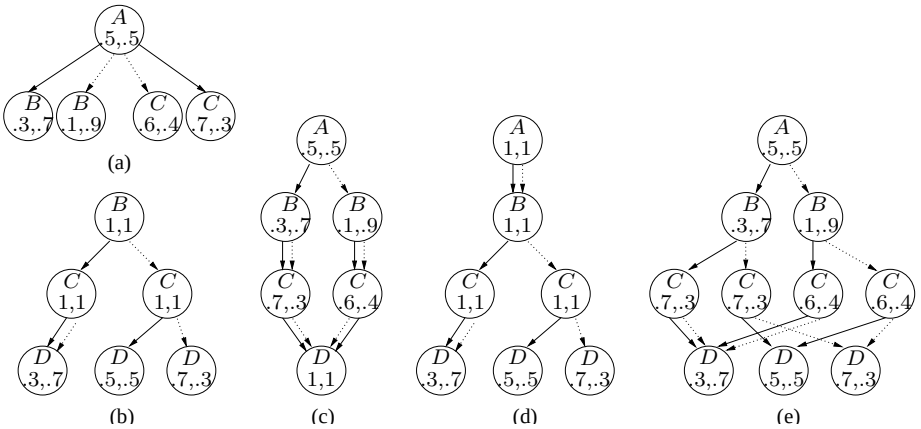

Figure 5: Compositional specification

quadratic bound in the size of G of the size of the junction tree. □

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.

6 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 5.2) 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 5 (a) shows a PDG for three binary random variables A, B, C (according to which B and C are independent given A). Figure 5 (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 4. This means that first for the variables A, B, C, D a forest F has to be determined which is a refinement of the forests 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 forest F. The problem of finding a forest 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 5 (b), and then compute the product. A second approach is provided by the results of Section 5: 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 5.1. 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 $P^{data}(X_i \mid \mathscr{A}(V_i))$ 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 iteration of the EM-algorithm has time complexity $O(N \mid G \mid)$, where N is the number of data items.

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 forests F, and, for each F, the possible refinements to a graph (V, E). This gives rise to hierarchical search strategies where a top-level search over F-structures is guided by approximate optimizations of the exact graph structure given the F-structure. For any given F, 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. A join operation reduces model complexity by merging nodes $\nu, \nu' \in V_i$ with $succ(\nu, Y, x_{i,h}) = succ(\nu', Y, x_{i,h})$ for all $Y \in succ_F(X_i), 1 \leq h \leq k_i$, and $(p_1^\nu, \ldots, p_{k_i}^\nu) \approx (p_1^{\nu'}, \ldots, p_{k_i}^{\nu'})$. Thus, join operations corresponds to the merging of states in the computation of a reduced PDG, only that now states are merged that are only approximately equivalent.

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 penalizes 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) that still differ widely with respect to complexity of probabilistic inference, because they give rise to junction trees of very different size. As eventually one 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. By limiting search to PDG-structures within a specified size bound, one moreover can force the learning procedure to produce models for which inference speed satisfies a specified performance requirement.

7 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. Most questions that Bozga and Maler then investigate in connection with these representations are quite distinct from the questions considered in this paper. An exception here is the issue of canonicity of representation, for which Bozga and Maler also present a result, which, however, only refers to representations over complete tree structures.

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: Boutilier 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 appears to be very difficult to specify a probability distribution directly as a circuit, or to learn a circuit representation from data. This makes this form of representation a secondary representation that will have to be obtained by compilation of some other representation like a Bayesian network (as proposed by Darwiche (2000)) – or a PDG.

8 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.

Our main goal in this paper was to develop the basic concepts of PDG representations (syntax, (independence-)semantics, algorithms and complexity of probabilistic inference), and to assess the respective efficiencies of

Bayesian network and PDG representations. Some other important questions here were only treated very superficially. Most important among these is the question of how to construct PDGs in practice, and especially how to learn PDGs from data. This is the subject of ongoing and future work.

Another aspect not considered here is the integration of PDGs into the Bayesian network paradigm by using them as a compact representation of conditional probability tables and/or clique potentials (as suggested by some of the previous approaches mentioned in section 7). The difficulty with this approach is that marginalization operations that need to be performed for probabilistic inference can cause an exponential blowup of the size of PDG representations, so that there are no guarantees that an initially compact representation of conditional probability tables will really make inference more efficient.

References

Andersson, S. A., Madigan, D. & Perlman, M. D. (1997), 'A characterization of markov equivalence classes for acyclic digraphs', *The Annals of Statistics* **25**(2), 505–541.

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 the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (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.

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 the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–2000)'.

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

Dechter, R. (1996), Bucket elimination: A unifying framework for probabilistic inference, *in* 'Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)', Portland, Oregon, pp. 211–219.

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.

Koenig, S. & Simmons, R. G. (1994), Risk-sensitive planning with probabilistic decision graphs, *in* 'Principles of Knowledge Representation and Reasoning, Proceedings of the Fourth International Conference (KR-94)', pp. 363–373.

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 the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–2000)'.

Pearl, J. (1988), *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, The Morgan Kaufmann series in representation and reasoning, rev. 2nd pr. edn, Morgan Kaufmann, San Mateo, CA.

Verma, T. & Pearl, J. (1991), Equivalence and synthesis of causal models, *in* P. Bonissone, M. Henrion, L. Kanal & J. Lemmer, eds, 'Uncertainty in Artificial Intelligence 6', Elsevier Science Publishers, pp. 255–268.

Zhang, N. L. & Poole, D. (1999), On the role of context-specific independence in probabilistic inference, *in* 'Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)', pp. 1288–1293.