MPE and Partial Inversion in Lifted Probabilistic Variable Elimination

Rodrigo de Salvo Braz and **Eyal Amir** and **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
{braz, eyal, danr}@cs.uiuc.edu

Abstract

It is often convenient to represent probabilistic models in a first-order fashion, using logical atoms such as $partners(X, Y)$ as random variables parameterized by logical variables. (de Salvo Braz, Amir, & Roth 2005), following (Poole 2003), give a *lifted* variable elimination algorithm (FOVE) for computing marginal probabilities from first-order probabilistic models (belief assessment, or BA). FOVE is lifted because it works directly at the first-order level, eliminating all the instantiations of a set of atoms in a single step, in some cases independently of the number of these instantiations. Previous work could treat only restricted potential functions. There, atoms' instantiations cannot constrain each other: predicates can appear at most once, or logical variables must not interact across atoms. In this paper, we present two contributions. The first one is a significantly more general lifted variable elimination algorithm, FOVE-P, that covers many cases where atoms share logical variables. The second contribution is to use FOVE-P for solving the Most Probable Explanation (MPE) problem, which consists of calculating the most probable assignment of the random variables in a model. The transition from BA to MPE is straightforward in propositional models, but the lifted first-order case is harder. We introduce the notion of *lifted assignments*, a *distribution* of values to a set of random variables rather than to each individual one. Lifted assignments are cheaper to compute while being as useful as regular assignments over that group. Both contributions advance the theoretical understanding of lifted probabilistic inference.

1 Introduction

Probabilistic models (Pearl 1988) offer a natural way to model domains with noise or incomplete information. In undirected graphical models, one can provide factors, or potential functions ϕ_i, over sets of random variables describing these facts. When the same dependency applies across multiple objects of the same type, it is natural to use *parameterized factors*, or *parfactors*, described over parameterized random variables. Parameterized random variables are represented as logical atoms whose parameters are logical variables, such as $retail(C)$, for whether each possible company C is a retailer. Parfactors stand for all factors obtained from instantiating its logical variables, and can be annotated by a constraint indicating which instantiations are valid. Consider a domain in which we model partnerships of companies around a certain product and whether they are retailers. An example of two parfactors is:

$\phi_1(partners(P, C_1, C_2)), C_1 \neq C_2.$
$\phi_2(partners(P, C_1, C_2), retail(C_1), retail(C_2)), C_1 \neq C_2.$

While more compact and clear than its propositional counterpart, parameterized models still need to be propositionalized for solving with a regular probabilistic inference algorithm. Propositionalized models are much larger and do not contain the original higher-level structure that can potentially be used for faster inference. Consequently, inference takes time that grows exponentially with domain size. This has been the approach in much of the work in the area (Ngo & Haddawy 1995; Ng & Subrahmanian 1992; Jaeger 1997; Kersting & De Raedt 2000; Koller & Pfeffer 1998; Poole 1993).

(de Salvo Braz, Amir, & Roth 2005), following (Poole 1993), describe FOVE (First-Order Variable Elimination), a *lifted* algorithm for calculating marginal probabilities (belief assessment, or BA) from a first-order probabilistic model. The algorithm is *lifted* in the sense that inference is carried on the first-order level, taking computational advantage of the model's structure and carrying similar calculations across groups of random variables.

FOVE uses two different elimination operations that are restricted in different ways. *Inversion elimination* operates in time independent from domain size, but can eliminate an atom only if it contains all logical variables in its parfactor, and its grounding is disjoint from any other atom's in that parfactor. *Counting elimination* can deal with atoms whose grounding is the same as some other atom in the parfactor, but logical variables in one atom cannot be constrained by those of others.

The first contribution of this paper is to present FOVE-P, a more general algorithm that can solve many cases where atoms share logical variables. Suppose a model describes the influence of a company doing business with two other companies and the occurrence of a conflict, by having a parfactor $\phi(business(X, Y), business(X, Z), conflict)$, for $X \neq Y \neq Z \neq X$, where $conflict$ is a 0-arity predicate indicating the occurrence of any conflict. FOVE cannot marginalize on $conflict$, since the $business$ atoms have

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

the same grounding (preventing inversion elimination) and share logical variable X (preventing counting elimination). This can however be solved by FOVE-P through a new operation called *partial inversion*.

The second contribution of this paper is showing how to use FOVE-P to obtain a lifted solution for Most Probable Explanation (MPE). MPE is an important probabilistic inference problem which consists of calculating an assignment with maximum probability to a model's random variables. While the transition from BA to MPE in the propositional case is straightforward and obtained by replacing sum operations by maximizing operations (Dechter 1999), in the lifted case one must deal with two main issues. First, there is now a notion of *lifted assignment*, that is, an assignment over groups of indistinguishable random variables that does not specify assignments on an individual basis, but describes a distribution of values to the group as a whole. Second, assignments must now be represented in a more explicit way than in the propositional case, since their descriptions are manipulated by the algorithm.

The notion of lifted assignments gives rise to queries on the number of objects in a domain with certain properties. This is not something usually done in the propositional case, and useful in domains in which one deals with large populations. In the example above, an instance of MPE with lifted assignments would be the calculation of the number of retail companies with maximum probability, given the domain model.

2 The FOVE Algorithm

This section recalls First-Order Probabilistic Inference (FOPI) definitions and the FOVE algorithm presented in (de Salvo Braz, Amir, & Roth 2005).

The FOVE and FOVE-P algorithms are developed under the framework of undirected graphical models such as Markov networks (Pearl 1988). These models are specified by a set of potential functions (or *factors*) defined over sets of random variables. Following (Poole 2003), the notion of factor is generalized to that of *parameterized* factor (or *parfactor*). A joint probability on the random variables is defined as being proportional to the product of factors. A parfactor describes a potential function on a set of random variables, but the random variables are now represented by logical atoms, possibly parameterized by logical variables. The logical variables are typed and each type is a discrete domain. One can therefore represent a potential function $\phi(friend(X,Y), friend(Y,X)), X \neq Y$ which stands for all factors instantiated from it by substitutions of X and Y that satisfy the constraint $X \neq Y$. In general, a parfactor is a triple $g = (\phi_g, A_g, C_g)$ of a potential function, a set of atoms, and a constraint on its logical variables. The constraints are equational formulas, that is, formulas where the only predicate is equality, with the unique name assumption. The algorithm makes use of a constraint solver for such formulas.

A FOPI probabilistic model is defined by a set of parfactors G and the associated logical variable domains (the population). Because each instantiation of a parfactor g is a factor in itself, the joint distribution defined by G on its ground random variables, denoted $RV(G)$, is the product of all instantiations of all parfactors:

$$P(RV(G)) \propto \prod_{g \in G} \prod_{\theta \in [C_g]} \phi_g(A_g\theta)$$

where $[C_g]$ is the set of substitutions satisfying C_g. We denote the second product above as $\Phi(g)$ and the entire right-hand side as $\Phi(G)$.

2.1 Lifted Belief Assessment

The Lifted Belief Assessment inference problem is that of calculating the marginal probability on a set of ground random variables Q:

$$P(Q) \propto \sum_{RV(G)\setminus Q} \Phi(G)$$

FOVE assumes that G is *shattered*, that is, given any two atoms in it, their groundings are either identical or disjoint. The process of shattering a model is described in (de Salvo Braz, Amir, & Roth 2005). Note that distinct atoms can have the same groundings, as it is the case with $p(X)$ and $p(Y)$ for X, Y with same domain. The algorithm works by selecting a set of atoms E to be eliminated at each step:

$$\sum_{RV(G)\setminus Q} \prod_{g \in G} \Phi(g) = \sum_{RV(G)\setminus RV(E)\setminus Q} \sum_{RV(E)} \prod_{g \in G} \Phi(g)$$

(by defining G_E the parfactors in G depending on $RV(E)$ and $G_{\neg E} = G \setminus G_E$)

$$= \sum_{RV(G)\setminus RV(E)\setminus Q} \prod_{g \in G_{\neg E}} \Phi(g) \sum_{RV(E)} \prod_{h \in G_E} \Phi(h)$$

(by *fusion* (figure 1), we obtain a single parfactor g_E such that $\Phi(G_E) = \Phi(g_E)$)

$$= \sum_{RV(G)\setminus RV(E)\setminus Q} \prod_{g \in G_{\neg E}} \Phi(g) \sum_{RV(E)} \Phi(g_E)$$

(by using the elimination operations described later, we obtain g' such that $\Phi(g')$ is equal to the last sum)

$$= \sum_{RV(G)\setminus RV(E)\setminus Q} \prod_{g \in G_{\neg E}} \Phi(g)\Phi(g')$$

(by defining $G' = G_{\neg E} \bigcup \{g'\}$)

$$= \sum_{RV(G')\setminus Q} \prod_{g \in G'} \Phi(g),$$

which reduces our problem to an instance with strictly less random variables.

However, not all choices of E are valid. The algorithm must pick E so that it satisfies the conditions for either of FOVE's two elimination operations: inversion elimination and counting elimination. We now recall them both.

Inversion Elimination Inversion elimination requires that there is one atom in E whose grounding is disjoint from all other atoms in g_E and which contains all its logical variables (the reason for this is provided below). This is the case for $q(X,Y)$ in a parfactor with atoms $p(X), q(X,Y)$, so it can be eliminated as follows:

$$\sum_{RV(q(X,Y))} \prod_{XY} \phi(p(X), q(X,Y))$$

(let $\{o_1, \ldots, o_n\}$ be the domain of both X and Y, and $\phi(i,j)$ shorthand for $\phi(p(o_i), q(o_i, o_j))$)

$$= \sum_{q(o_1,o_1)} \sum_{q(o_1,o_2)} \cdots \sum_{q(o_n,o_n)} \phi(1,1)\phi(1,2)\ldots\phi(n,n)$$

(by observing that $\phi(i,j)$ depends on $q(o_i,o_j)$ only)

$$= \sum_{q(o_1,o_1)} \phi(1,1) \cdots \sum_{q(o_n,o_n)} \phi(n,n)$$

(by observing that only $\phi(i,j)$ depends on $q(o_i,o_j)$)

$$= \Big(\sum_{q(o_1,o_1)} \phi(1,1) \Big) \ldots \Big(\sum_{q(o_n,o_n)} \phi(n,n) \Big)$$
$$= \prod_{XY} \sum_{q(X,Y)} \phi(p(X), q(X,Y))$$
$$= \prod_{XY} \phi'(p(X)) = \prod_{X} \phi'(p(X))^{|Y|} = \prod_{X} \phi''(p(X)).$$

However one cannot eliminate $q(X,Y)$ from parfactors with atoms $(p(X,Z), q(X,Y))$ or $(q(X,Y), q(Z,W))$. The inverted atom (the one under the sum) needs to contain all logical variables because its expansion would not be one-to-one with the expansion of the products otherwise.

Note that inversion elimination reduces a sum over all assignments to the *set* of random variables $RV(q(X,Y))$ to one over the assignments to a single one, and its cost is independent from $|RV(q(X,Y))|$.

Counting Elimination Counting elimination requires that E be all atoms with any logical variables in g_E, and that the logical variables in any two atoms of E do not constrain each other under C_g. Therefore, it can eliminate $p(X)$ from $(p(X), q(Y))$ but not from $(p(X), q(Y))$ with $X \neq Y$ since the choice of X limits the choice of Y. In FOVE, cases not covered by these operations have to be directly calculated.

Before we present the theorem on which counting elimination is based, some preliminary definitions are necessary.

First, we define the notion of independent atoms given a constraint. Intuitively, this happens when choosing a substitution for one atom from the constraint's solutions does not change the possible choices of substitutions for the other atom. Let α be any object, $LV(\alpha)$ be the logical variables in α, and $C_{|L}$ be the projection of a constraint C to a set of logical variables L. Let $\bar{X}_1$ and $\bar{X}_2$ be two sets of logical variables and C a constraint such that $\bar{X}_1 \cup \bar{X}_2 \subseteq LV(C)$. $\bar{X}_1$ is *independent from* $\bar{X}_2$ *given* C if, for any substitution $\theta_2 \in [C_{|\bar{X}_2}]$, $C_{|\bar{X}_1} \Leftrightarrow (C\theta_2)_{|\bar{X}_1}$. $\bar{X}_1$ and $\bar{X}_2$ are *independent given* C if $\bar{X}_1$ is independent from $\bar{X}_2$ given C and vice-versa. Two *atoms* $p_1(\bar{X}_1)$ and $p_2(\bar{X}_2)$ are *independent given* C if $\bar{X}_1$ and $\bar{X}_2$ are independent given C.

The second notion is that of *just-different* atoms, a notion present only implicitly in (de Salvo Braz, Amir, & Roth 2005). This means that choosing a substitution for the first atom restricts the choices for the second one in exactly one substitution. This is defined only for atoms with the same predicate. Two atoms $p(\bar{X}_1)$ and $p(\bar{X}_2)$ are *just-different given* C if $C_{|\bar{X}_1 \cup \bar{X}_2} \Leftrightarrow C_{|\bar{X}_1} \wedge C_{|\bar{X}_2} \wedge \bar{X}_1 \neq \bar{X}_2$. This formula declares that $\bar{X}_1 \cup \bar{X}_2$ is allowed, by C, to assume any values allowed to each individual set, as far as they are different, but with no other restrictions.

Finally, we define multinomial counters. Let a be an atom (with an associated constraint system) with domain D_a. Then the *multinomial counter of* a $\vec{N}_a$ is a vector where $\vec{N}_{a,j}$ indicates how many groundings of a are assigned the j-th value in D_a. The multinomial coefficient $\vec{N}_a!$ is defined as $(\vec{N}_{a,1}, \ldots, \vec{N}_{a,|D_a|})! = \frac{(\vec{N}_{a,1}+\cdots+\vec{N}_{a,|D_a|})!}{\vec{N}_{a,1}!\ldots\vec{N}_{a,|D_a|}!}$ (Abramowitz & Stegun 1972).

The set of multinomial counters for a *set* of atoms A is denoted $\vec{N}_A$, and the product $\prod_{a \in A} \vec{N}_a!$ of their multinomial coefficients is denoted $\vec{N}_A!$.

Theorem 1. Let g be a shattered parfactor and $E = \{E_1, \ldots, E_k\}$ be a subset of A_g such that $RV(E)$ is disjoint from $RV(A_g \setminus E)$, $A' = A_g \setminus E$ are all ground, and where each pair of atoms of different predicates are independent given C_g and each pair of atoms of same predicate are just-different given C_g. Then

$$\sum_{RV(E)} \Phi(g) = \sum_{RV(E)} \prod_{\theta \in \Theta_g} \phi(A_g\theta)$$
$$= \sum_{\vec{N}_E} \vec{N}_E! \prod_{v \in D_E} \phi(v, A')^{\#(v, \vec{N}_E)}$$
$$\#(v, \vec{N}_E) = \prod_{i=1}^{k} (\vec{N}_{E_i, v_i} - excluded(i))$$

$excluded(i) = |\{j < i : E_i, E_j \text{ are just-different given } C_g\}|$.

The idea behind the proof is that there is a limited number of values v in which E can be instantiated inside ϕ. We therefore calculate $\phi(v)$ for each of them and exponentiate it by the $\#(v, \vec{N}_E)$ different ways E can be instantiated into v, given the counters $\vec{N}_E$. This number is calculated by choosing, for each v_i, from the $\vec{N}_{E_i,v_i}$ random variables in $RV(E_i)$ assigned v_i, minus the ones already taken from previous just-different atoms. Once potentials are expressed in terms of counters, we can group assignments on $RV(E)$ by their counters $\vec{N}_E$, taking into account that there are $\vec{N}_E!$ of them.

In particular, we cannot apply counting elimination to parfactors where atoms share logical variables, since this means that their logical variables constrain each other (by equality). Since atoms in the same parfactor will usually be related in that way, this restriction is a strong one.

3 Partial Inversion

In (de Salvo Braz, Amir, & Roth 2005), counting elimination was restricted to cases when eliminated atoms were independent from each other, or at least just-different. This prevents its application to some important cases. Suppose a parfactor establishes the potential for conflict (c) when a company does business (b) with more than one other company:

$$\sum_{b(X,Y)} \prod_{\neq(X,Y,Z)} \phi(b(X,Y), b(X,Z), c)$$

In the above, $\neq (X,Y,Z)$ stands for pairwise difference between X, Y, Z and the $RV(b(X,Y))$ under the summation is written as $b(X,Y)$ for simplicity. Note that we could have used $RV(b(X,Z))$ under the sum just as well. Then the shared logical variable X between atoms prevents counting elimination from being used. Inversion elimination is not possible either, because no atom contains all logical variables in the parfactor.

Partial inversion is a technique in which a sum-product inversion on *some* of the logical variables is performed on the parfactor, binding them. This binding is in many cases enough to reduce the problem to a solvable one. In our example, assuming the domain is $o_1, \ldots, o_n$, we can invert X by rewriting the above as

$$\begin{aligned}
&\sum_{b(o_1,Y)} \cdots \sum_{b(o_n,Y)} \\
&\quad \prod_{Y,Z,\neq(o_1,Y,Z)} \phi(b(o_1,Y), b(o_1,Z), c) \ldots \\
&\quad \prod_{Y,Z,\neq(o_n,Y,Z)} \phi(b(o_n,Y), b(o_n,Z), c) \\
&= \Big(\sum_{b(o_1,Y)} \prod_{Y,Z,\neq(o_1,Y,Z)} \phi(b(o_1,Y), b(o_1,Z), c) \Big) \cdots \\
&\quad \Big(\sum_{b(o_n,Y)} \prod_{Y,Z,\neq(o_n,Y,Z)} \phi(b(o_n,Y), b(o_n,Z), c) \Big) \\
&= \prod_{X} \sum_{b(x,Y)} \prod_{Y,Z,\neq(x,Y,Z)} \phi(b(x,Y), b(x,Z), c)
\end{aligned}$$

where X is bound and written as x for emphasis. Because of that, the summation can be solved with regular counting elimination, yielding a new potential function $\phi'(c)$. Note that $\phi'(c)$ does not depend on the particular value of x; we can therefore calculate it with any arbitrary value in the domain of X. We are left with

$$\prod_{X} \phi'(c) = \phi'^{|X|} = \phi''(c).$$

We call the atoms whose corresponding summations are inverted *inverted atoms*.

3.1 Groundings of inverted atoms by distinct substitutions must be disjoint

Sometimes we cannot apply partial inversion. Consider the expansion of the case in which the second b atom is $b(Z,X)$:

$$\begin{aligned}
&\sum_{b(o_1,Y)} \cdots \sum_{b(o_n,Y)} \\
&\quad \prod_{Y,Z,\neq(o_1,Y,Z)} \phi(b(o_1,Y), b(Z,o_1), c) \ldots \\
&\quad \prod_{Y,Z,\neq(o_n,Y,Z)} \phi(b(o_n,Y), b(Z,o_n), c)
\end{aligned}$$

This does not yield the same type of factoring, since every product involves some random variable from each of the summations. This motivates the following.

Condition 3.1. Partial inversion of logical variables L in a parfactor g is applicable only if the groundings $RV(A_L\theta_1)$ and $RV(A_L\theta_2)$ are disjoint, where A_L is $\{a \in A_g : L \subseteq LV(a)\}$ and θ_1 and θ_2 are any two distinct substitutions of L:

$$\forall \theta', \theta'' \in [C_{g|L}]\ \theta' \neq \theta'' \Rightarrow RV(A_L\theta') \cap RV(A_L\theta'') = \emptyset.$$

Such a formula can be decided by using the constraint solver, but we omit the details here.

3.2 Uniform Solution Counting Partition (USCP)

Consider the inversion of Y resulting in the expression

$$\prod_{Y} \sum_{p(y,Z)} \prod_{Z\neq y, Z\neq a} \phi(p(y,Z)).$$

The summation cannot be calculated independently of y, since $|RV(p(y,Z))|$ will depend on whether $Y = a$. One needs to consider instead

$$\Big(\prod_{Y=a} \sum_{p(y,Z)} \prod_{Z\neq y} \phi(p(y,Z)) \Big) \Big(\prod_{Y\neq a} \sum_{p(y,Z)} \prod_{Z\neq y} \phi(p(y,Z)) \Big)$$

(by treating each part separately and having y' and y'' arbitrary solutions of $Y = a$ and $Y \neq a$ respectively)

$$\begin{aligned}
&= \Big(\sum_{p(y',Z)} \prod_{Z\neq y'} \phi(p(y',Z)) \Big)^{|[Y=a]|} \\
&\quad \Big(\sum_{p(y'',Z)} \prod_{Z\neq y''} \phi(p(y'',Z)) \Big)^{|[Y\neq a]|}
\end{aligned}$$

(by observing that $y' = a$)

$$= \Big(\sum_{p(a,Z)} \prod_{Z\neq a} \phi(p(a,Z)) \Big)^{1} \Big(\sum_{p(y,Z)} \prod_{Z\neq y} \phi(p(y,Z)) \Big)^{|[Y\neq a]|}$$

(by using inversion elimination twice)

$$= \phi'()^1 \phi''()^{|[Y\neq a]|} = \phi'''().$$

In general, one needs to consider the *uniform solution counting partition* (USCP) of the inverted logical variables. The USCP is a partition of inverted logical variables substitutions set into subsets that yield a uniform number of solutions for the *remaining* logical variables. It can be computed by using the constraint solver.

3.3 All atoms with inverted logical variables must be inverted

It is necessary to always invert all atoms of an inverted logical variable, as the following example illustrates. Consider the following valid inversion of Y in $q(X,Y)$:

$$\sum_{p(X,Y),q(Y,Z)} \prod_{XYZ} \phi(p(X,Y)q(Y,Z))$$
$$= \sum_{p(X,Y)} \sum_{q(Y,Z)} \prod_{Y} \prod_{XZ} \phi(p(X,Y)q(Y,Z))$$
$$= \sum_{p(X,Y)} \prod_{Y} \sum_{q(y,Z)} \prod_{XZ} \phi(p(X,y)q(y,Z))$$

Solving the inner summation provides a function on $\vec{N}_{p(X,y)}$ which is different for each Y. Instead, one should invert both $p(X,Y)$ and $q(Y,Z)$, obtaining

$$= \prod_{Y} \sum_{p(X,y)} \sum_{q(y,Z)} \prod_{XZ} \phi(p(X,y)q(y,Z))$$

whose sum is counting eliminable.

Finally, it is important to note that when inverting *all* logical variables of a parfactor, the inner sum will be a propositional one that does not involve counters. In this case we would not have the problem above, and one can invert only one of the atoms. After that, the atom is eliminated as in the propositional case since all its logical variables will be bound. This inversion and then propositional elimination amount to the inversion elimination of this atom as done by FOVE.

4 The FOVE-P algorithm

The FOVE-P algorithm presented in figure 1 is a modification of FOVE that incorporates partial inversion. Because partial inversion, combined with a propositional elimination, is equivalent to FOVE's inversion elimination, this operation is not present in FOVE-P.

FOVE-P works as follows: it selects the next group of atoms to be eliminated, E, according to the restrictions imposed by partial inversion and counting elimination. This is done by the procedure *FIND-ELIMINABLE-ATOMS*.

In selecting eliminable atoms, it is important to consider *all* parfactors involving their grounding. Even parfactors with *distinct* atoms from E need to be considered, if these distinct atoms still have the same groundings as the eliminable atoms. This is why the algorithm often uses G_e and G_E, the subsets of parfactors G which depend on $RV(e)$ and $RV(E)$, respectively.

After E is determined, the algorithm *fuses* the parfactors in G_E, that is, calculates an equivalent single parfactor g. Then g is subjected to as many partial inversions as possible, possibly reducing the problem to counting or propositional elimination. If E is a single atom with all logical variables in g, applying partial elimination will reduce the problem to a propositional elimination step, essentially reproducing the inversion elimination operation present in FOVE.

FIND-ELIMINABLE-ATOMS returns a set of atoms E that will include all non-ground atoms in G_E, a necessary condition for counting elimination. The only exception to this is when it finds a single atom e which contains all logical variables in $fusion(G_e)$, since this will allow the partial inversion and propositional elimination combination that reproduces inversion elimination.

Because propositional elimination can be construed as a particular case of counting elimination (where propositional assignments are seen as counters indicating the value of single random variables), we can say that FOVE-P consists of partial inversion and counting elimination only. While FOVE-P subsumes FOVE, it is more general because partial inversion applies to parfactors to which FOVE's inversion or counting elimination do not.

It is worthwhile restating what FOVE-P can and cannot do. The algorithm can process parfactors that are subject to one or more partial eliminations (as per condition 3.1) and then to counting (or propositional) elimination. It still cannot deal with cases in which it is not possible to apply partial inversion enough in order to reduce to parfactors without atoms that constrain each other (so that we can apply counting or propositional elimination). A simple example of such case is eliminating $RV(p(X))$ and $RV(q(Y))$ from $\phi(p(X),q(Y)), X \neq Y$, since neither partial inversion or counting elimination apply. We are currently investigating further lifted inference methods to deal with these uncovered cases.

5 Lifted Most Probable Explanation

5.1 Lifted Assignments

In first-order models, assignments might be regular assignments, represented by formulas with equality, or *lifted assignments*. A lifted assignment is in fact a description that fits an entire *set* of assignments, all of them equally maximizing potentials. There are *universally* or *existentially* quantified lifted assignments.

Universally quantified lifted assignments declare that a certain assignment is valid for all values of a set of logical variables. Its general form is $\forall\ C\ \varphi$, where C is a constraint on the quantified logical variables and φ is a formula representing an assignment, of which an example is $\forall\ \ Y \neq b\ \ p(Y) = a$, that is, in a particular lifted assignment, $p(Y)$ is equal to a, for all Y.

An example of a universal lifted assignment is the following:

$$\forall Y\ q(X,Y) = v_1 \wedge r(X,Y) = v_2$$

which means that, for some unbound X, for all values of Y we have $q(X,Y)$ assigned v_1 and $r(X,Y)$ assigned v_2.

We often have *functions* returning lifted assignments. The following is a function on the value of $p(X)$:

$$f(p(X)) \stackrel{\text{def}}{=}$$
$$\forall Y\ q(X,Y) = f'(p(X)) \wedge r(X,Y) = f''(q(X,Y))$$

where f' is a function from the domain of $p(X)$ to the domain of $q(X,Y)$ and f'' is function from the domain of $q(X,Y)$ to the domain of $r(X,Y)$.

PROCEDURE *FOVE-P*(G, Q)
G a set of parfactors, $Q \subseteq RV(G)$, G shattered against Q.
1. If $RV(G) = Q$, return G.
2. $E \leftarrow$ *FIND-ELIMINABLE-ATOMS*(G, Q).
3. $g_E \leftarrow$ *FUSION*(G_E).
4. $g' \leftarrow$ *ELIMINATE*(g_E, E).
5. Return *FOVE*$(\{g'\} \cup G_{\neg E}, Q)$.

PROCEDURE *FIND-ELIMINABLE-ATOMS*(G, Q)
1. Choose e from $A_G \setminus Q$.
2. $g \leftarrow$ *FUSION*(G_e)
3. If $LV(e) = LV(g)$ and $\forall e' \in A_g\ RV(e') \neq RV(e)$ return $\{e\}$ (inversion eliminable).
4. $E \leftarrow \{e\}$.
5. While $E \neq$ non-ground atoms of G_E $E \leftarrow E \cup$ non-ground atoms of G_E.
6. Return E.

PROCEDURE *ELIMINATE*(g, E)
1. If $LV(g) = \emptyset$ (propositional case) return parfactor $(\sum_E \phi_g(A_g), A_g \setminus E, \top)$.
2. If $g' \leftarrow$ *PARTIAL-INVERSION*(g, E) suceeds return g'.
3. Return *COUNTING*(g, E).

PROCEDURE *PARTIAL-INVERSION*(g, E)
1. $L \leftarrow$ *INVERTIBLE*(g, E).
2. $U \leftarrow$ USCP(C_g, L) (section 3.2).
3. Return $\prod_{C \in U}$ *ELIMINATE*$(g\theta_C, E\theta_C)^{|D_L|}$, where θ_C is an arbitrary element of $[C]$.

PROCEDURE *INVERTIBLE*(g, E)
1. If there is a largest $L \in LV(g)$ such that, for $A_L = \{a \in A_g : L \subseteq LV(a)\}$, $\forall \theta_1, \theta_2 \in [C_{g|L}]\ RV(A_L\theta_1) \cap RV(A_L\theta_2) = \emptyset$ return L.
2. Fail.

PROCEDURE *COUNTING*(g, E)
1. If counting elimination of E valid for g return $(\sum_{\vec{N}_E} \vec{N}_E! \prod_v \phi_g(v, A_g \setminus E)^{\#(v, \vec{N}_E)}, A_g \setminus E, \top)$.
2. Fail.

PROCEDURE *FUSION*(G)
1. $C_G \leftarrow \bigwedge_{g \in G} C_g$.
2. Return parfactor $(\prod_{g \in G} \phi_g^{|[C_G]|/|[C_g]|}, \bigcup_{g \in G} A_g, C_G)$ (fusion is detailed in (de Salvo Braz, Amir, & Roth 2005). Essentially, it calculates a parfactor equivalent to a *set* of them).

Notation:
- $LV(\alpha)$: logical variables in object α.
- D_L: domain of logical variable set L.
- $|Y|$: size of domain of logical variable Y.
- $g\theta$: parfactor $(\phi_g, A_g\theta, C_g\theta)$.
- $C_{|L}$: constraints projected to a set of logical variables L.
- G_E: subset of parfactors G which depend on $RV(E)$.
- $G_{\neg E}$: subset of parfactors G which do not depend on $RV(E)$.
- $\top$: tautology constraint.

Figure 1: FOVE-P.

Existentially quantified lifted assignments declare that a specific number of substitutions make a subformula true. Its general form is $[\exists_{v \in D}^{n(v)}\ L : C]\ \varphi(v)$, which means that, for every $v \in D$, there are $n(v)$ substitutions for logical variables L satisfying constraint C for which $\varphi(v)$ holds.

An example of an existentially quantified lifted assignment is

$$[\exists_{v \in D}^{n(v)} XY : X \neq Y]\ r(X,Y) = f(v)$$

which indicates that there are, for each $v \in D$, $n(v)$ substitutions of (X,Y) in $[X \neq Y]$ such that $r(X,Y) = f(v)$, for n a function from D to natural numbers and f a function from D to the domain of $r(X,Y)$.

An example of a function on $\vec{N}_{p(X)}$ returning existentially quantified lifted assignments is

$$f'(\vec{N}_{p(X)}) \stackrel{\text{def}}{=} [\exists_{v \in D_{p(X),p(Y)}}^{\#(v, \vec{N}_{p(X)})} XY : X \neq Y]\ r(X,Y) = f(v)$$

where $\#$ is the function used in counting elimination.

5.2 The mpe operator

In the Most Probable Explanation inference problem (MPE) we must identify some assignment to random variables with maximum potential:

$$\text{Maximum Potential of G} = \max_{RV(G)} \Phi(G).$$

The expression above is very similar to $\sum_{RV(G)} \Phi(G)$. In *propositional* probabilistic inference, a common way of solving this problem is to use Variable Elimination, but replacing summations by maximizations. The maximizing assignment can then be trivially obtained as a side effect of maximizations, although it is not explicitly represented in the expression above. In lifted MPE, assignments are manipulated in non-trivial ways and need to be treated more explicitly. This is why we introduce the mpe *operator*, a combination for $\max$ and $\arg\max$ operators, which allows us to explicitly refer to maximizing assignments.

For any real function f and variable q, we define

$$\operatorname*{mpe}_q f(q) = \left(\max_q f(q'), q = \arg\max_{q'} f(q')\right).$$

The result of mpe is a pair of the maximum value of $f(q)$ and the maximizing assignment to q, represented by a formula with equality. We call such a pair p a *potential-assignment pair*, or pa-pair, and its components are $p^{\mathbf{P}}$ and $p^{\mathbf{A}}$ respectively. Let $\phi(q)$ be a function returning potential-assignment pairs. Then we define

$$\operatorname*{mpe}_q \phi(q) = \left(\max_q \phi^{\mathbf{P}}(q), q = \arg\max_q \phi^{\mathbf{P}}(q) \wedge \phi^{\mathbf{A}}(q)\right),$$

that is, if given a function that provides pa-pairs, mpe returns a pair maximizing that potential for q, with an assignment represented by the conjunction of maximizing assignment to q and whatever assignment $g^{\mathbf{A}}(q)$ is. It is important to note that, like $\sum$ and $\max$, but unlike $\arg\max$, we can decompose mpe in the following manner: for $\bar{q} = \{q_1, \ldots, q_n\}$,

$$\operatorname*{mpe}_{\bar{q}} \phi(\bar{q}) = \operatorname*{mpe}_{q_1} \ldots \operatorname*{mpe}_{q_n} \phi(\bar{q}).$$

5.3 FOVE-P for MPE

Once we have the mpe operator and parfactors that map to pa-pairs rather than simply potentials, it is straightforward to use the FOVE-P algorithm to solve MPE, by simply replacing $\sum$ by mpe and having an empty query. The use of mpe guarantees that maximization is being performed and that elimination operations produce parfactors returning maximizing pa-pairs with assignments to previously eliminated variables.

However, because the algorithm performs some operations on potentials, it is important to define what these operations mean for pa-pair potentials. These operations are essentially two: (a) the product of potentials and (b) the conversion of potential functions from being defined on assignments to atoms to being defined on counters.

Pa-pair products. The product operation is straightforward. The product of reals is kept for the potential component while conjunction is used for the formula component. For r and s pa-pairs, $r \times s = (r^{\mathbf{P}} s^{\mathbf{P}}, r^{\mathbf{A}} \wedge s^{\mathbf{A}})$.

The pa-pair product contributes in forming universally quantified lifted assignments because universal quantification is a form of conjunction. Assume that g is a parfactor on $p(X), q(X,Y)$ with marginals given by $\phi^{\mathbf{P}}$ and $\phi^{\mathbf{A}}$ defining assignments on a previously removed $r(X,Y)$ by a function f. We can invert X, Y and obtain

$$
\begin{aligned}
&\mathop{\mathrm{mpe}}_{q(X,Y)} \Phi(g) \\
&= \mathop{\mathrm{mpe}}_{q(X,Y)} \prod_{X,Y} \Big(\phi_g^{\mathbf{P}}(p(X), q(X,Y)), \\
&\qquad\qquad r(X,Y) = f(p(X), q(X,Y))\Big) \\
&= \prod_{X,Y} \mathop{\mathrm{mpe}}_{q(X,Y)} \Big(\phi_g^{\mathbf{P}}(p(X), q(X,Y)), \\
&\qquad\qquad r(X,Y) = f(p(X), q(X,Y))\Big) \\
&= \prod_{X,Y} \Big(\max_{q(X,Y)} \phi_g^{\mathbf{P}}(p(X), q(X,Y)), \\
&\qquad q(X,Y) = \arg\max_{q(X,Y)} \phi_g^{\mathbf{P}}(\cdot) \wedge r(\cdot) = f(\cdot)\Big)
\end{aligned}
$$

(by defining appropriate ϕ' and f')

$$
= \prod_{X,Y} \Big(\phi'^{\mathbf{P}}(p(X)), q(X,Y) = f'(p(X)) \wedge r(\cdot) = f(\cdot)\Big)
$$

(because the marginal depends only on $p(X)$ and the pa-pair product results in the conjunction of lifted assignments)

$$
\begin{aligned}
&= \prod_{X} \Big(\phi'^{\mathbf{P}}(p(X))^{|Y|}, \\
&\qquad \forall Y\ q(X,Y) = f'(p(X)) \wedge r(\cdot) = f(\cdot)\Big) \\
&= \Phi(g')
\end{aligned}
$$

for some appropriate parfactor $g' = (\phi_{g'}, \{p(X)\}, C_{g'})$.

The interpretation of this lifted assignment is that, for the X in question, it is true that, for every Y, $q(X,Y)$ is assigned $f'(v)$ and $r(X,Y)$ is assigned $f(v, q(X,Y)) = f(v, f'(v))$.

In general, universally quantified lifted assignments are formed at the point of the algorithm where a parfactor g has an irrelevant subset L of logical variables so that we have

$$
\begin{aligned}
&\prod_{\theta_L \in [C_{g|L}]} \prod_{\theta \in [C_{g|LV(g)\setminus L}]} \Big(\phi^{\mathbf{P}}(A_g\theta), \phi^{\mathbf{A}}(A_g\theta)\Big) \\
&= \prod_{\theta \in [C_{|LV(g)\setminus L}]} \Big((\phi^{\mathbf{P}})^{|[C_{g|L}]|}(A_g\theta), [\forall L : C_{g|L}]\phi^{\mathbf{A}}(A_g\theta)\Big)
\end{aligned}
$$

Conversion to potential functions on counters. We now show the conversion of potential functions to functions on counters by example and in general. In the following, mpe is over $RV(p(X))$, which is the same as $RV(p(Y))$:

$$
\mathop{\mathrm{mpe}}_{p(X)} \prod_{X \neq Y} \Big(\phi^{\mathbf{P}}(p(X), p(Y)), r(X,Y) = f(p(X), p(Y))\Big)
$$

(by expressing the product as the number of times ϕ is applied to each possible argument v)

$$
\begin{aligned}
&= \mathop{\mathrm{mpe}}_{p(X)} \Big(\prod_{v \in D_{p(X),p(Y)}} \phi^{\mathbf{P}}(v)^{\#(v, \vec{N}_{p(X)})}, \\
&\qquad [\exists^{\#(v,\vec{N}_{p(X)})}_{v \in D_{p(X),p(Y)}} XY : X \neq Y]\ r(X,Y) = f(v)\Big)
\end{aligned}
$$

(by grouping assignments on $RV(p(X))$ by $\vec{N}_{p(X)}$)

$$
\begin{aligned}
&= \mathop{\mathrm{mpe}}_{\vec{N}_{p(X)}} \Big(\prod_{v \in D_{p(X),p(Y)}} \phi^{\mathbf{P}}(v)^{\#(v, \vec{N}_{p(X)})}, \\
&\qquad [\exists^{\#(v,\vec{N}_{p(X)})}_{v \in D_{p(X),p(Y)}} XY : X \neq Y]\ r(X,Y) = f(v)\Big)
\end{aligned}
$$

(by defining appropriate f' and f'')

$$
= \mathop{\mathrm{mpe}}_{\vec{N}_{p(X)}} \Big(f'(\vec{N}_{p(X)}), f''(\vec{N}_{p(X)})\Big)
$$

(by applying the definition of mpe)

$$
\begin{aligned}
&= \Big(\max_{\vec{N}_{p(X)}} f'(\vec{N}_{p(X)}), \\
&\qquad \vec{N}_{p(X)} = \arg\max_{\vec{N'}_{p(X)}} f'(\vec{N'}_{p(X)}) \wedge f''(\vec{N}_{p(X)})\Big)
\end{aligned}
$$

(by defining an appropriate parfactor $g' = (\phi', \emptyset, \top)$)

$$
= \Big(\phi'^{\mathbf{P}}(), \phi'^{\mathbf{A}}()\Big) = \Phi(g').
$$

g' is a constant parfactor returning the maximum potential and the lifted assignment formed by the distribution of values on $p(X)$ (represented by $\vec{N}_{p(X)}$) maximizing f', and, for each $v \in D_{p(X),p(Y)}$, there are $\#(v, \vec{N}_{p(X)})$ pairs (X,Y) satisfying $X \neq Y$ such that $r(X,Y)$ is assigned $f''(v)$.

In general, existentially quantified lifted assignments are formed at the point of the algorithm where a parfactor g is rewritten as a function of counters, so that we have

$$\prod_{\theta \in [C_g]} \left(\phi_g^{\mathbf{P}}(A_g\theta), \phi_g^{\mathbf{A}}(A_g\theta) \right)$$
$$= \Big(\prod_{v \in D_{A_g}} \phi_g^{\mathbf{P}}(v)^{\#(v,\vec{N}_{A_g})}, \exists_{v \in D_{A_g}}^{\#(v,\vec{N}_{A_g})} \phi_g^{\mathbf{A}}(v) \Big)$$

With these operations defined, FOVE-P can be used to solve the MPE problem.

6 Example

We now illustrate MPE inference with partial inversion in a simple business domain. We want to express the fact that some companies establish partnerships, and it will often be the case that, for each product in question, one of the partners is a retail company while the other is not. We go back to the model presented in the introduction

$\phi_1(partners(P, C_1, C_2)), C_1 \neq C_2.$
$\phi_2(partners(P, C_1, C_2), retail(C_1), retail(C_2)), C_1 \neq C_2.$

where ϕ_1 and ϕ_2 are parfactors with appropriate potential functions for the desired dependencies. ϕ_1 performs the role of a "prior" (although such a notion is not precise in an undirected model as this). If there are 15 companies in the domain, our implementation (available at `http://l2r.cs.uiuc.edu/~cogcomp/`) produces a lifted assignment that can be read as
"for all products, there are 8 retail companies and 7 non-retail companies. 56 pairs of companies are partners and the rest is not".

It is interesting to notice that the numbers above do not depend on the particular potentials, but only on the assignments made more likely by each parfactor. This suggests that FOVE-P can be further developed to take advantage of MPE being an easier problem than BA, avoiding the iteration over counters and instead deducing the maximum one from the parfactor structure alone. This would make MPE's time complexity completely independent from the domain size.

7 Conclusion

This paper makes two contributions towards a greater theoretical understanding of lifted probabilistic inference. The first one is a generalization of FOVE to FOVE-P, which is able to deal with important cases in first-order probabilistic inference (FOPI) where atoms share logical variables. The second one is a representation and manipulation of lifted assignments so that FOVE-P can be used to solve the MPE problem as well. Lifted assignments offer an automatic solution to determine the number of objects in a population to have certain features, given a model of the domain, with maximum probability. Both are steps towards advanced probabilistic inference algorithms that can take advantage of compact and expressive representations.

Many directions remain to be taken: approximate inference, queries with non-ground atoms and introduction of function symbols are among the most important ones. Moreover, FOVE-P does not cover all possible constraint configurations in our language, and it is desirable to either generalize it further or to show this is not possible.

8 Acknowledgements

This work was partly supported by Cycorp in relation to the Cyc technology, the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) program, NSF grant ITR-IIS- 0085980, and a Defense Advanced Research Projects Agency (DARPA) grant HR0011-05-1-0040. We thank the anonymous reviewers for helpful comments, as well as Dafna Shahaf for her valuable help.

References

Abramowitz, M., and Stegun, I. A. 1972. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York, NY: Dover.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *Proceedings of IJCAI-05, 19th International Joint Conference on Artificial Intelligence*.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113(1-2):41–85.

Jaeger, M. 1997. Relational Bayesian networks. In Kaufmann, M., ed., *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 266–273.

Kersting, K., and De Raedt, L. 2000. Bayesian logic programs. In Cussens, J., and Frisch, A., eds., *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 138–155.

Koller, D., and Pfeffer, A. 1998. Probabilistic frame-based systems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, 580–587.

Ng, R. T., and Subrahmanian, V. S. 1992. Probabilistic logic programming. *Information and Computation* 101(2):150–201.

Ngo, L., and Haddawy, P. 1995. Probabilistic logic programming and Bayesian networks. In *Asian Computing Science Conference*, 286–300.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Mateo (Calif.).

Poole, D. 1993. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence* 64(1):81–129.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 985–991.