Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Lifted MEU by Weighted Model Counting

Udi Apsel and **Ronen I. Brafman**
Computer Science Department
Ben Gurion University of The Negev
Beer-Sheva, Israel 84105
apsel,brafman@cs.bgu.ac.il

Abstract

Recent work in the field of probabilistic inference demonstrated the efficiency of weighted model counting (WMC) engines for exact inference in propositional and, very recently, first order models. To date, these methods have not been applied to decision making models, propositional or first order, such as influence diagrams, and Markov decision networks (MDN). In this paper we show how this technique can be applied to such models. First, we show how WMC can be used to solve (propositional) MDNs. Then, we show how this can be extended to handle a first-order model – the Markov Logic Decision Network (MLDN). WMC offers two central benefits: it is a very simple and very efficient technique. This is particularly true for the first-order case, where the WMC approach is simpler conceptually, and, in many cases, more effective computationally than the existing methods for solving MLDNs via first-order variable elimination, or via propositionalization. We demonstrate the above empirically.

Introduction

Techniques for aiding in and automating decision making tasks are central to AI. A central goal of work in this area is to develop models that can succinctly capture complex problems, as well as their inherent structure, and utilize this structure, via novel inference techniques, to gain computational efficiency. To this end, recent work has started to explore exact inference and decision making in relational probabilistic and decision models [Nath and Domingos, 2009; den Broeck et al., 2010], using first-order variable elimination (FOVE) [Poole, 2003; Apsel and Brafman, 2011]. Relational models are one of the key ideas introduced to enable scaling up probabilistic models, and they appear to capture structure that is central to many large-scale decision problems. Relational, or lifted inference techniques, such as FOVE, can lead to exponential reduction in computation time, in comparison to standard, propositional variable elimination, allowing to solve models that are much beyond the reach of propositional inference.

But FOVE is a conceptually complex technique, and especially so in the context of decision models. This complexity also affects its efficiency. The main contribution of this paper is to build upon developments in the area of first-order probabilistic inference to provide a much simpler method that is, in many cases, more efficient computationally for solving first-order decision models, based on the idea of Weighted Model Counting (WMC) [Chavira and Darwiche, 2008].

WMC is a very effective technique for exact inference in propositional probabilistic models. It combines tools that efficiently exploit local structure in a given problem, such as CNF encoding methods, knowledge compilation techniques, unit propagation, and caching mechanisms. Moreover, it was recently extended to handle first-order probabilistic models [Gogate and Domingos, 2011; den Broeck et al., 2011]. Its first-order formulation maintains the simplicity of the approach – it is much simpler than FOVE – and very efficient.

Surprisingly, WMC has not been used so far in solving even *propositional* decision models, such as influence diagrams (ID) [Howard and Matheson, 1984] and Markov decision networks (MDN) [Nath and Domingos, 2009]. Thus, this paper starts by showing how WMC can be applied to the main inference task of *propositional* decision models, the computation of maximal expected utility (MEU), and then how these ideas can be extended to first-order decision models. We focus on undirected decision models, the propositional MDN, and its relational variant – the Markov logic decision network [Nath and Domingos, 2009] in which WMC is slightly easier to explain. However, WMC can and has been used to solve both directed and undirected models, and can be applied to IDs, as well. We conclude with several experiments, comparing the results of lifted MEU using WMC with FOVE.

Background

Markov Decision Network

A Markov Decision Network (MDN) [Nath and Domingos, 2009] is an undirected graph, constituting a graphical representation of a probabilistic decision model. Similarly to IDs, MDNs consist of three types of nodes: decision, chance and utility (see Figure 1). Decision nodes represent the actions that are carried by the decision-maker, which influence (via edges) both chance and utility nodes. Under a given set of concrete decisions, the chance nodes of an MDN form a Markov network. Utility nodes, which are connected to

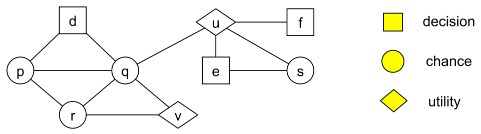


Figure 1: A Markov decision network (MDN)

chance and decision nodes only, are real value functions over the scope of the connected nodes. The task of solving an MDN is synonymous with finding the set of decisions which maximizes the expected utility. In this paper, we define the properties of an MDN using a set of weighted propositional formulas. Hence, the table entries of utility and probability factors are depicted by weights, corresponding to a propositional formula. Formally, an MDN M is a set of triples $M = \bigcup_i (f_i, \phi_i, \mu_i)$, where f_i is a propositional formula over decision variables $dv(M)$ and random variables $rv(M)$, $\phi_i \in \mathbb{R}_0^+$ is a probability weight and $\mu_i \in \mathbb{R}$ is a utility weight. A formula f_i whose respective $\phi_i \neq 1$ is called a probability formula (PF), and a formula f_j whose respective $\mu_j \neq 0$ is called a utility formula (UF). It is possible for a formula to be both PF and UF. As a convention, we use the notation $\varphi|_\tau$ to depict the expression φ under the assignment τ, namely $(x^2 + y)|_{x=2}$ equals $4 + y$. Under a set of assignments d to all decision variables, the expected utility of M is given by

$$eu[M|_d] = \frac{1}{Z|_d} \cdot \sum_{r \in range(rv(M))} \prod_i \phi_i|_{d,r} \sum_i \mu_i|_{d,r} \quad \text{(1a)}$$

Where

$$Z|_d = \sum_{r \in range(rv(M))} \prod_i \phi_i|_{d,r} \quad \text{(1b)}$$

If formula f_i is satisfied by d and r , $\phi_i|_{d,r}$ and $\mu_i|_{d,r}$ equal ϕ_i and μ_i, respectively. Otherwise, their values are 1 and 0, respectively. $Z|_d$ here is the partition function of the underlying Markov network, and its value is dependent on the chosen decisions. The MEU of M is given by

$$meu[M] = (\operatorname*{argmax}_d \; eu[M|_d] \; , \; \max_d \; eu[M|_d]) \quad (2)$$

Lastly, we assume all variables in M to be binary, although this constraint can be lifted. We now move to define the WMC framework, which we adapt later to MDNs.

Weighted Model Counting

Given a CNF formula C defined over the set of variables $\bar{x}$, Model Counting (MC) [Li, 2010] is the #P-complete task of computing the number of distinct assignments (i.e. models) which satisfy C, denoted by $sat(C)$. Weighted Model Counting (WMC) is a generalization of MC, where each variable $x \in \bar{x}$ is assigned a weight, $weight(x)$, for TRUE (w_x) and FALSE ($w_{\neg x}$) assignments. The weight of a model τ is the product of all variable weights. The WMC solution is the sum of weights of all models which satisfy C.

$$\text{WMC(C)} = \sum_{\tau \in sat(C)} \prod_{x \in X} weight(x)|_\tau \quad (3)$$

Algorithm 1: bwmc(C)

```
  input  : CNF $C$ with assigned weights
  output: WMC of $C$
1 if all variables are assigned and C is true then return 1
2 if C contains an empty false clause then return 0
3 $x \leftarrow$ Choose an unassigned variable
4 return $w_x \times bwmc(C|_{x=\mathbb{T}}) + w_{\neg x} \times bwmc(C|_{x=\mathbb{F}})$
```

Algorithm 1 depicts a basic WMC algorithm (bwmc), which is directly derived from the WMC definition. Since probabilistic queries in both Bayesian and Markov networks can be redefined as WMC tasks, WMC algorithms have been extensively studied in recent years [Chavira and Darwiche, 2008]. The general framework requires (a) a conversion of the probabilistic model to a set of weighted propositional formulas, (b) distributing weights to all variables for positive and negative assignments, (c) assembling all formulas to a single CNF formula, and (d) running the WMC engine. State-of-the-art engines incorporate a more sophisticated version of the basic WMC, along with a set of tools for caching [Sang, Beame, and Kautz, 2005], knowledge compilation [Huang and Darwiche, 2005] and simplifying the CNF by unit propagation [Li and Anbulagan, 1997]. Our aim here is to show how to exploit the existing framework and set of tools for the benefit of EU computation. We do so by redefining the EU computation in MDNs as a WMC task.

Expected Utility as WMC

Given MDN $M = \bigcup_i (f_i, \phi_i, \mu_i)$ and a set of assignments d to all decision variables, we seek to compute $eu[M|_d]$. First, we construct a CNF formula $C = CNF\{\bigwedge_i (f_i|_d \Leftrightarrow s_i)\}$. The s_i elements are newly added variables, which indicate whether a formula f_i is satisfied. We use the notation $vars(C)$ to depict the set of all unassigned variables in CNF C. Second, we distribute weights to all unassigned variables in C (namely, non-decision variables). Unlike classical WMC, a weight w is defined as a pair $w = (\phi, \mu)$, where ϕ and μ are probability and utility components, respectively. We provide a weight $w_{\neg x} = (1, 0)$ to each variable x in C, a weight $w_x = (1, 0)$ to all random variables, and a weight $w_{s_i} = (\phi_i, \mu_i)$ to all indicator variables. Third, following variable elimination for joint factors [Koller and Friedman, 2009], we define two new operators on weights $w_x = (\phi_x, \mu_x)$ and $w_y = (\phi_y, \mu_y)$ as follows

- $w_x + w_y = (\phi_x + \phi_y, \frac{\phi_x \cdot \mu_x + \phi_y \cdot \mu_y}{\phi_x + \phi_y})$
- $w_x \times w_y = (\phi_x \cdot \phi_y, \mu_x + \mu_y)$

Where the $\times$ operator takes precedence over $+$, and $(0, -\infty)$ is defined as the identity element w.r.t. $+$, namely: $(0, -\infty) + w_x \equiv w_x$. Note that $(0, -\infty) \times w_x = (0, -\infty)$ and $(1, 0) \times w_x = w_x$ are inherently derived from the definition of the operators. Interestingly, the new operators we introduce are closely related to the concept of a commutative semi-ring presented in [Kimmig, den Broeck, and Raedt, 2011]. Lastly, C is passed as input into Algorithm 2, which computes the EU of M under assignment d.

Algorithm 2: wmc(C)

```
input  : CNF $C$ with assigned weights
output: Pair $(\phi, \mu)$, where $\mu$ is the EU of MDN $M$
1  if all clauses in $C$ are true then
2      return $\prod_{x \in vars(C)} (w_x + w_{\neg x})$
3  if $C$ contains an empty false clause then
4      return $(0, -\infty)$
   // Decomposition
5  if $C = \bigwedge_i C_i$, s.t. $\forall i \neq j : C_i \perp C_j$ then
6      return $\prod_i wmc(C_i)$
   // Recursive conditioning
7  $x \leftarrow$ Choose an unassigned variable
8  return $w_x \times wmc(C|_{x=\mathbb{T}}) + w_{\neg x} \times wmc(C|_{x=\mathbb{F}})$
```

Algorithm 2, as can be seen, is practically identical to that of partition function computation in probabilistic models [Gogate and Domingos, 2011]. In fact, the probability component of the result *is* the partition function. The difference lies in the weight distribution procedure and in the interpretation of the $+$ and $\times$ operators. Lines 1–4 handle the basic scenarios, where C is a tautology or a false formula. Unlike the basic WMC, which assumes all variables to be assigned, the algorithm takes advantage of unassigned variables in a tautology, thus avoids unnecessary recursions. Lines 5–6 apply the WMC procedure separately on independent components of CNF C, providing a more efficient solution. Two sets of clauses C_i, C_j are considered independent ($C_i \perp C_j$) if they do not share a variable. Lines 7–8 condition on a chosen variable ($x = \mathbb{T}, x = \mathbb{F}$), and combine the two recursive calls into a single return value. The output of the algorithm is a pair (ϕ, μ), where ϕ is the partition function of the underlying Markov Network, and μ is the expected utility of MDN M.

Proposition 1. *Alg. 2 computes the Expected Utility of $M|_d$.*

Proof outline. *We define the weight of variable x as a pair of real numbers, $weight(x) = (prob(x), util(x))$, and apply the new operator interpretations to Equation 3. Since (i) both operators $(+, \times)$ are commutative and associative, and (ii) operator $\times$ distributes over operator $+$, sufficient conditions are met for Algorithm 2 to compute WMC(C). The weighted model count can be formulated as follows*

$$WMC(C) = \Big(\sum_{\tau} prob(\bar{x})|_{\tau}, \frac{\sum_{\tau} prob(\bar{x})|_{\tau} \cdot util(\bar{x})|_{\tau}}{\sum_{\tau} prob(\bar{x})|_{\tau}}\Big) \tag{4}$$

Where each instance of $\sum_{\tau}$ denotes $\sum_{\tau \in sat(C)}$, and

$$prob(\bar{x})|_{\tau} = \prod_{x \in \bar{x}} prob(x)|_{\tau} \quad , \quad util(\bar{x})|_{\tau} = \sum_{x \in \bar{x}} util(x)|_{\tau}$$

It can be shown that each unique assignment $r \in range(rv(M))$ maps one-to-one to a unique model $\tau \in sat(C)$, and vise versa. In each such occurrence, $prob(\bar{x})|_{\tau} = \prod_i \phi_i|_{d,r}$ and $util(\bar{x})|_{\tau} = \sum_i \mu_i|_{d,r}$ (see Equation 1a), since only the weights of the s_i variables of which $f_i|_{d,r} = \mathbb{T}$ affect the result. From there, it follows that the utility component of WMC(C) equals $eu[M|_d]$. □

Algorithm 3: dwmc(C)

```
input  : CNF $C$ with assigned weights
output: Pair $(\phi, \mu)$, where $\mu$ depicts the MEU
1  if all clauses in $C$ are true then
2      return $\prod_{x \in vars(C)} (w_x + w_{\neg x})$
3  if $C$ contains an empty false clause then
4      return $(0, -\infty)$
   // Decomposition
5  if $C = \bigwedge_i C_i$, s.t. $\forall i \neq j : C_i \perp C_j = \emptyset$ then
6      return $\prod_i dwmc(C_i)$
   // Recursive conditioning
7  if $C$ contains unassigned decision variables then
8      $x \leftarrow$ Choose an unassigned decision variable
9      return $\max\{dwmc(C|_{x=\mathbb{T}}) , dwmc(C|_{x=\mathbb{F}})\}$
10 else
11     $x \leftarrow$ Choose an unassigned variable
12     return $w_x \times dwmc(C|_{x=\mathbb{T}}) + w_{\neg x} \times dwmc(C|_{x=\mathbb{F}})$
```

Maximum Expected Utility by Decision-Induced WMC

We now address the task of finding an assignment d to maximize the EU of MDN M. One way of solving this problem is to compute the EU of each valid assignment d. In some cases this is indeed the only possible way to find the MEU, however – even in such tightly-coupled models, the search space can be pruned using branch and bound techniques [Huang, Chavira, and Darwiche, 2006]. We do not elaborate here on such pruning approaches, but rather – point to conditions which allow a decomposition of the problem, and ultimately – a more efficient solution. Fortunately, conditions for decomposition in this case are similar to WMC. Namely, clauses C_i and C_j are independent if they share no variables.

A first step before invoking Algorithm 3, the decision-induced WMC (DWMC), is to construct a CNF C and distribute weights to all variables, as before. In-order to maintain a similar framework of WMC, decision variables are assigned weights $w_x = w_{\neg x} = (0.5, 0)$ s.t. their sum equals $(1, 0)$, and does not modify the weight of the model. We also define an additional operator $\max\{w_x, w_y\}$, where $w_x = (\phi_x, \mu_x)$ and $w_y = (\phi_y, \mu_y)$, which yields w_x if $\mu_x > \mu_y$, and w_y otherwise. We note that any MEU computation should produce both the maximizing assignment and a numerical value, whereas our formulation produces only the numerical result. Adding a "decision path" to the algorithm is an easy task which we chose to omit, since it involves several technical details which may potentially shift the focus from the core principles of WMC.

Proposition 2. *Alg. 3 computes the MEU of MDN M.*

Proof outline. *Let $dv(C)$ denote the set of all decision variables in C. The decision induced WMC is defined as follows*

$$DWMC(C) = \max_{d \in range(dv(C))} WMC(C|_d) \tag{5}$$

Ignoring lines 5–6 for the moment, it can be shown that for each given decision d, Algorithm 3 computes the underly-

ing weighted model count, by recursively conditioning on decision variables first (line 8) and picking the maximum result (line 9). When C is a tautology (lines 1–2), the algorithm ignores the decision variables, since $\forall x \in dv(C)$: $w_x + w_{\neg x} = (1, 0)$. When all decision variables are assigned, the algorithm resorts to a simple WMC(C) computation. Hence, Algorithm 3 computes DWMC(C). Since C and M contain the same set of decision variables, namely $dv(M) = dv(C)$, each $M|_d$ is mapped to a single $C|_d$, and vice versa, where $eu[M|_d] =$ WMC($C|_d$) for any d. From there it immediately follows that $meu[M] =$ DWMC(C). Lastly, the decomposition rule in lines 5–6 is derived from WMC, and maintains correctness under the properties of operators max and $\times$. □

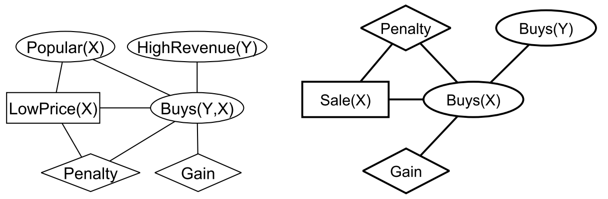
Lifted WMC in First-Order Decision Networks

Here, we build on our decision-induced WMC formulation for a straight-forward adaption of the work of [Gogate and Domingos, 2011], introducing a novel method to compute MEU in first-order models. Unlike the FOVE algorithm, which deviates from the simplicity of propositional variable elimination, lifted WMC closely preserves the framework of the propositional case. We start by a brief introduction to first-order logic concepts, move on to present the relevant decision model, Markov Logic Decision Network, and finally – briefly lay out our adaption of lifted WMC for MEU.

Background – First-Order Models

First-Order Formulas First-order formulas are logical formulas which consist of several types of symbols: *constants*, *logical variables* and *predicates*, along with a set of *quantifiers* ($\exists$, $\forall$) and *logical connectives* ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$) [Nath and Domingos, 2009]. Constant symbols represent objects in some domain of interest (Alice, Bob, etc.). Logical variables (e.g. X, Y) symbolize a range in the objects of some domain, where $dom(X)$ depicts the domain of X. Predicates symbolize relations among objects ($Friends$) or attributes of objects ($Smokes$). *Interpretations* determine which objects and relations are represented by which symbols. Formulas are recursively constructed from first-order formulas, using logical connectives, quantifiers and parenthesis. For example, $\forall X_1 \forall X_2 \;\; Friends(X, Y) \Rightarrow (Smokes(X) \Leftrightarrow Smokes(Y))$. Similarly to other works which focus on lifted inference techniques, the scope of this paper is limited to quantifier-free first-order formulas, such as $Smokes(X) \wedge Friends(X, Y) \Rightarrow Smokes(Y)$.

Atoms An atomic formula (i.e. *atom*) is a predicate applied to one or more terms, where a *term* is a constant or a logical variable. In case where all terms are constants, the atom is also called a *ground atom*. For example: $Friends(X, Y)$ is an atom, where X, Y are logical variables, and $Friends(Alice, Bob)$ is a ground atom. In our model, each variable (chance, decision or indicator) corresponds to some unique ground atom. The notation $gr(x)$ is used to denote the set of ground atoms depicted by atom x.



(a) The price range problem (b) The seasonal sale problem

Figure 2: Two examples of MLDNs

Substitutions, Propositionalization and Constraints A *substitution* is the assignment of a constant value to one logical variable or more. *Propositionalization* is the operation of issuing all *legal* substitutions on a set of logical variables of some formula φ. The legality of a substitution is determined by a *constraint* on the logical variables of φ, expressing which substitutions are valid. For instance, the propositionalization of $Sm(X) \wedge Fr(X, Y) => Sm(Y)$, where the domain of both X and Y is $\{Alice, Bob\}$ and under the constraint $X \neq Y$, results in two formulas:

- $Sm(Alice) \wedge Fr(Alice, Bob) => Sm(Bob)$
- $Sm(Bob) \wedge Fr(Bob, Alice) => Sm(Alice)$

The influence of a set of constraints R on the set of ground elements is denoted using ": R". For instance, $gr(x) : R$ is the set of all ground atoms of x, and $L : R$ is the set of all legal substitutions on the set of logical variables L.

Shattering *Shattering* [de Salvo Braz, Amir, and Roth, 2005] is the alignment of a set of clauses C s.t. each pair of atoms taken from C, e.g. x and y, conforms to the following: (i) $gr(x) = gr(y)$ or (ii) $gr(x) \cap gr(y) = \emptyset$. For instance, $c_1 = Fr(X, Y) \wedge Sm(Y)$ and $c_2 = Fr(Alice, Y)$ are not shattered, since $gr(Fr(X, Y))$ and $gr(Fr(Alice, Y))$ are neither identical nor disjoint. A *shattered model* of same propositional properties would require a constraint $X \neq Alice$ to be attached to c_1, and a new additional clause to be added, $c_1' = Fr(Alice, Y) \wedge Sm(Y)$.

Markov Logic Decision Network

A Markov Logic Decision Network (MLDN) can be defined, intuitively, as a *first-order* influence diagram with a Markov Logic Network [Richardson and Domingos, 2006] as its probabilistic network. Formally, an MLDN M_L is a set of tuples $M_L = \bigcup_i (f_i, r_i, \phi_i, \mu_i)$, where f_i is a first-order formula over chance atoms $ra(M_L)$ and decision atoms $da(M_L)$, r_i is a constraint over the logical variables of f_i, $\phi_i \in \mathbb{R}_0^+$ is a probability weight and $\mu_i \in \mathbb{R}$ is a utility weight. Once supplied a concrete set of constants which represent domains and domain objects, an MDN can be obtained by applying propositionalization to all elements of M_L. As mentioned earlier, it is preferable, and usually much more efficient, to solve a decision task whilst maintaining the original first-order representation of the problem. This allows bulk operations to be applied simultaneously, while refraining from an explicit extraction of the underlying propositional model.

Example Consider the following problem: given a set of products (X) and a list of potential buyers (Y), a price range (*Low/High*) has to be determined for each of the products, such that the expected profit (utility) is maximized. We define the decision problem using first-order formulas, starting with probability formulas: (i) $Popular(X) \wedge LowPrice(X) \Rightarrow Buys(Y, X)$, (ii) $HighRevenue(Y) \Rightarrow Buys(Y, X)$, continuing with a utility gain formula (iii) $Buys(Y, X)$, and concluding with a utility penalty (a negative utility weight) for selling in a low price: (iv) $Buys(Y, X) \wedge LowPrice(X)$. Naturally, $LowPrice$ is a decision atom, and the rest of the atoms are chance atoms. A graphical representation is given in Figure 2a.

Lifted Decision-Induced WMC

Given MLDN M, we define the lifted decision-induced WMC (LDWMC) framework for lifted MEU computation, which is very similar to the propositional case. First, a (first-order) CNF formula C is constructed out of M, as follows: $C = CNF\{\bigwedge_i (f_i \Leftrightarrow s_i)\}$. s_i here is an indicator atom (rather than an indicator variable), consisting of all logical variables in f_i. For instance, let $f_i = Smokes(X) \wedge Drinks(Y)$. In this case, $f_i \Leftrightarrow s_i$ denotes $(Smokes(X) \wedge Drinks(Y)) \Leftrightarrow S_i(X, Y)$, where $s_i \equiv S_i(X, Y)$. We use $atoms(C)$ to depict the set of all unassigned atoms in C. The second step is a weight distribution procedure, carried identically to the propositional case, with the exception of weights being distributed to atoms instead of variables. Intuitively, this means that in an equivalent propositional model, all ground atoms entailed by atom x are assigned the same weight. To complement the set of operators in the lifted case, we add two additional operators which apply to weight $w_x = (\phi_x, \mu_x)$ and scalar n, as follows: $n \times w_x = (n \cdot \phi_x, \mu_x)$, and $w_x{}^n = (\phi_x{}^n, n \cdot \mu_x)$. Finally, CNF C and a set of constraints R, which consists of all constraints r_i, are passed to the LDWMC algorithm.

The Algorithm

Algorithm 4 introduces a lifted method for decision-induced WMC. Its input is the CNF C and the set R of all constraints from the MLDN. Unlike [Gogate and Domingos, 2011], which shatter C only when needed, we simplify the formulation (while compromising efficiency) by shattering before each step of the algorithm. The lifted inference/decision making is achieved by exploiting the symmetry inherent to the first-order representation, in each of the cases defined in the propositional algorithm. For instance, lines 1–2 handle the tautology case, where each atom x represents a set of ground atoms of size $|gr(x) : R|$. The power operator simply simulates the repeated multiplications required in an equivalent propositional representation.

CNF Decomposition is lifted by two separate procedures. Lines 5–6 repeat the procedure of the propositional case, by decomposing C into separate sets of clauses, sharing no atoms. Lines 7–9 exploit a unique feature of the first-order representation, by finding a set of logical variables $K = \bigcup_i \{X_i\}$ called a *lifted decomposer*, which, when propositionalized, produces $|X_i : R|$ symmetrical and independent components. The decomposer K consists of one X_i from each clause, forming a set of unifiable logical variables. The idea is to compute the LWDMC on one component only, by substituting each X_i with an arbitrary constant κ_0 (hence, $C|_{K/\kappa_0}$ and $R|_{K/\kappa_0}$), and raising the result to the power of the number of symmetrical components. The rules for finding such a decomposer are found in [Jha et al., 2010].

Algorithm 4: ldwmc(C,R)

```
   input : Shattered first-order CNF $C$ with assigned
           weights, set of constraints $R$
   output: Pair $(\phi, \mu)$, where $\mu$ depicts the MEU
 1 if all clauses in $C$ are true then
 2     return $\prod_{x \in atoms(C)} (w_x + w_{\neg x})^{|gr(x):R|}$
 3 if $C$ contains an empty false clause then
 4     return $(0, -\infty)$
   // Decomposition
 5 if $C = \bigwedge_i C_i$, s.t. $\forall i \neq j : C_i \perp C_j$ then
 6     return $\prod_i ldwmc(C_i, R)$
   // Lifted Decomposition
 7 if $K = \{X_1, \ldots, X_m\}$ is a lifted decomposer of $C$ then
 8     return $ldwmc(C|_{K/\kappa_0}, R|_{K/\kappa_0})^{|X_i:R|}$
   // Conditioning by Atom Count
 9 if $C$ contains unassigned decision atoms then
10     $x \leftarrow$ Choose an unassigned decision atom
11     return $\max_{k \in \{0 \ldots n\}} ldwmc(C|_{\chi_k^n}, R|_{\chi_k^n})$
12 else
13     $x \leftarrow$ Choose an unassigned atom
14     return
       $\sum_{k=0}^{n} \binom{n}{k} \times w_x{}^k \times w_{\neg x}{}^{n-k} \times ldwmc(C|_{\chi_k^n}, R|_{\chi_k^n})$
```

Lines 10–17 describe conditioning by *atom count* [den Broeck et al., 2011], which we only briefly cover. Such operation requires the weighted model count to be affected only by the number of true assignments to the ground atoms depicted by x. Generally, such conditioning is allowed on any atoms containing a single logical variable, but can be carried in some other cases as well. Assuming the condition is met, χ_k^n depicts the operation of applying k true assignments and $n - k$ false assignments to x, where $n \equiv |gr(x) : R|$ and $(C|_{\chi_k^n}, R|_{\chi_k^n})$ are the affected CNF and set of constraints. The coefficient $\binom{n}{k}$ in line 14 simulates repeated count permutations under the same count of true assignments, and the power operator reflects the effect of simultaneously conditioning on k true and $n - k$ false assignments.

We note that the algorithm may always resort to propositionalization if none of the lifting operators can be applied.

Related Work

At present,[Apsel and Brafman, 2011] is the only method for solving relational decision models. It is based on C-FOVE [Milch et al., 2008], and, as a result, suffers from several inherent disadvantages. First and foremost is the reliance on fusion [de Salvo Braz, Amir, and Roth, 2005] to eliminate atoms from the model, thereby consuming much memory

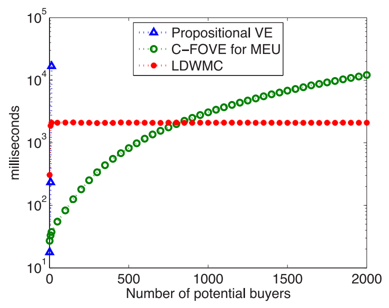


Figure 3: The price range problem

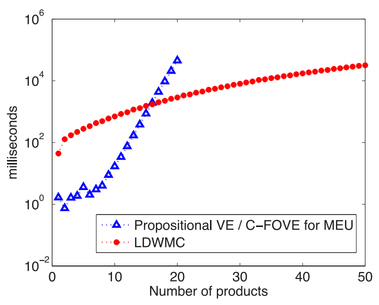


Figure 4: The seasonal sale problem

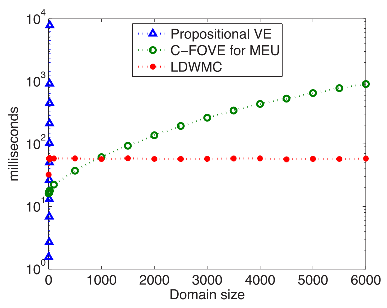


Figure 5: PF $d(X) \Rightarrow p(X)$ and UF $p(X)$

and time, where in many cases lifted WMC would solve the problem quickly and efficiently by exploiting conditional independence. Second, is the restrictive conditions under which C-FOVE allows an atom to be counted. WMC, in contrast, can count any atom consisting of a single logical variable. Third, is the indifference to table content, which prevents FOVE from exploiting sparse factors. Fourth, which is specific to MEU, is the unique property of the FOVE inversion operator, which imports decision atoms from probability factors into utility factors. This property has a potentially devastating computational effect, since atoms contained in both types of factors cannot be eliminated by inversion. We demonstrate a few of these points in the experiments section.

In contrast to the relational case, there are many algorithms for solving propositional models, such as variable elimination [Dechter, 2000], recursive conditioning [Ahlmann-Ohlsen et al., 2009] and Branch and Bound [Yuan, Wu, and Hansen, 2010]. None of which, however, explicitly adopt WMC. The most relevant work is of [Bhattacharjya and Shachter, 2007; Shachter and Bhattacharjya, 2010], where the concept of a decision circuit is introduced, followed by a method for solving IDs using dynamic programming. As with arithmetic circuits [Darwiche, 2003] and knowledge compilation for WMC [Chavira and Darwiche, 2008], the two approaches (decision circuits and WMC) are closely related. The main difference, offered by WMC, lies in the distinction between independent phases in the inference procedure. Namely, WMC does not require knowledge compilation, although such a preprocess phase would assist with finding a good variable ordering, whereas circuit evaluation requires a prior circuit construction. This distinction allows the transition to first-order MEU which we introduce in this paper. Indeed, incorporating existing first-order knowledge compilation methods [den Broeck et al., 2011] for decision making, is an interesting subject for further study.

Empirical Evaluation

We present three sets of experiments, in which time results are presented in a logarithmic scale. All three sets compare the performance of our own lifted decision-induced WMC implementation (LDWMC), which was written in Scala, against propositional VE (Java) and C-FOVE for MEU (Java) [Apsel and Brafman, 2011]. Figure 3 depicts the results of the price range problem, where a price range for 10 products is chosen, for a varying number of potential buyers. We see that propositional VE quickly runs out of memory, and that LWDMC outperforms C-FOVE from a domain size of 850 and on. The first reason for this result is that $LowPrice(X)$ (decision atom) and $Popular(X)$ (chance atom) share a factor, a structure which prevents C-FOVE from applying counting conversion on $LowPrice(X)$. The second reason is that the two atoms differ in their type, which means they cannot be joined into a joint formula, hence – cannot be counted together. Once $Buys(Y, X)$ is eliminated by inversion, $HighRevenue(Y)$ is applied with counting conversion and $Popular(X)$ is eliminated by inversion, only then $LowPrice(X)$ is being counted. LWDMC, in comparison, is domain size independent since it applies atom-count conditioning on $LowPrice(X)$, followed by a lifted decomposition on all the Y logical variables. Namely, LWDMC is dependent only on the domain size of logical variable X, which is a constant.

Figure 4 depicts the results of the seasonal sale problem (see Figure 2b), where a set of products has to be chosen, in-order to maximize the revenue per client in a seasonal sale. The problem is presented by two probability formulas, (i) $Sale(X) \Rightarrow Buys(X)$ and (ii) $Buys(X) \Rightarrow Buys(Y)$ (a buyer is likely to buy again), one utility gain formula (iii) $Buys(X)$, and one utility penalty formula (iv) $Buys(X) \wedge Sale(X)$. As before, C-FOVE is unable to count the decision atom $Sale(X)$, which shares a factor with the chance atom $Buys(X)$. Unlike previously, C-FOVE is also unable to eliminate $Buys(X)$ by inversion. The result is C-FOVE resorting to propositional inference, which is represented by the propositional VE data set. LWDMC, again, has no problem with conditioning on the $Sale(X)$ atom, and quickly outperforms the alternative.

Figure 5 depicts the results of a simple model, consisting of one probability formula, $d(X) \Rightarrow p(X)$, and one utility formula, $p(X)$, where $d(X)$ is a decision atom. Here, C-FOVE eliminates $p(X)$ by inversion, and imports $d(X)$ to a utility factor. As a result, C-FOVE needs to count the $d(X)$ atom. LDWMC, in comparison, simply applies lifted decomposition on logical variable X. Hence, the run-time of LDWMC in this case is domain-size independent.

Conclusion

We introduced a technique for solving probabilistic decision models, and specifically MDN, by formulating exact EU computation as a WMC task, and exact MEU computation as a decision-induced WMC. This formulation allowed us to build on existing work, and introduce a novel WMC-based method for solving first-order decision models, such as MLDN. Finally, we demonstrated the discussed advantages of the new method by comparing it with C-FOVE for MEU and propositional VE, in a number of models.

Acknowledgments

The authors are partially supported by ISF grant 8254320, the Paul Ivanier Center for Robotics Research and Production Management, and the Lynn and William Frankel Center for Computer Science.

References

Ahlmann-Ohlsen, K. S.; Jensen, F. V.; Nielsen, T. D.; Pedersen, O.; and Vomlelová, M. 2009. A comparison of two approaches for solving unconstrained influence diagrams. *Int. J. Approx. Reasoning* 50:153–173.

Apsel, U., and Brafman, R. 2011. Extended lifted inference with joint formulas. In *Proceedings of the Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 11–18. Corvallis, Oregon: AUAI Press.

Bhattacharjya, D., and Shachter, R. D. 2007. Evaluating influence diagrams with decision circuits. In *UAI*, 9–16.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artif. Intell.* 172:772–799.

Darwiche, A. 2003. A differential approach to inference in bayesian networks. *J. ACM* 50(3):280–305.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *IJCAI*, 1319–1325.

Dechter, R. 2000. A new perspective on algorithms for optimizing policies under uncertainty. In *AIPS*, 72–81.

den Broeck, G. V.; Thon, I.; van Otterlo, M.; and Raedt, L. D. 2010. Dtproblog: A decision-theoretic probabilistic prolog. In *AAAI*.

den Broeck, G. V.; Taghipour, N.; Meert, W.; Davis, J.; and Raedt, L. D. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, 2178–2185.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of the Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 256–265. Corvallis, Oregon: AUAI Press.

Howard, R. A., and Matheson, J. E. 1984. Influence diagrams. In Howard, R. A., and Matheson, J. E., eds., *Readings on the Principles and Applications of Decision Analysis*. Menlo Park, CA: Strategic Decision Group.

Huang, J., and Darwiche, A. 2005. Dpll with a trace: From sat to knowledge compilation. In *IJCAI*, 156–162.

Huang, J.; Chavira, M.; and Darwiche, A. 2006. Solving map exactly by searching on compiled arithmetic circuits. In *AAAI*.

Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted inference seen from the other side : The tractable features. In Lafferty, J.; Williams, C. K. I.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 23*. 973–981.

Kimmig, A.; den Broeck, G. V.; and Raedt, L. D. 2011. An algebraic prolog for reasoning about possible worlds. In *AAAI*.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*.

Li, C. M., and Anbulagan. 1997. Heuristics based on unit propagation for satisfiability problems. In *IJCAI (1)*, 366–371.

Li, W. 2010. *Exploiting Structure in Backtracking Algorithms for Propositional and Probabilistic Reasoning*. Ph.D. Dissertation, University of Waterloo.

Milch, B.; Zettlemoyer, L. S.; Kersting, K.; Haimes, M.; and Kaelbling, L. P. 2008. Lifted probabilistic inference with counting formulas. In *AAAI*, 1062–1068.

Nath, A., and Domingos, P. 2009. A language for relational decision theory, in proceedings of the international workshop on statistical relational learning, leuven, belgium.

Poole, D. 2003. First-order probabilistic inference. In Gottlob, G., and Walsh, T., eds., *IJCAI*, 985–991. Morgan Kaufmann.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Sang, T.; Beame, P.; and Kautz, H. A. 2005. Performing bayesian inference by weighted model counting. In *AAAI*, 475–482.

Shachter, R. D., and Bhattacharjya, D. 2010. Dynamic programming in in uence diagrams with decision circuits. In *UAI*, 509–516.

Yuan, C.; Wu, X.; and Hansen, E. A. 2010. Solving multi-stage influence diagrams using branch-and-bound search. In *UAI*, 691–700.