Probabilistic Inference via Weighted Model Counting

Algorithms, Encodings, and Random Instances

Paulius Dilkas

15th June 2021

The Problem of Computing Probability

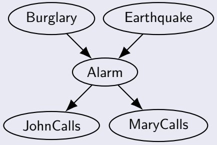
ProbLog

```
0.001  ::  burglary.
0.002  ::  earthquake.
0.95   ::  alarm     :- burglary, earthquake.
0.94   ::  alarm     :- burglary, \+ earthquake.
0.29   ::  alarm     :- \+ burglary, earthquake.
0.001  ::  alarm     :- \+ burglary, \+ earthquake.
0.9    ::  johnCalls :- alarm.
0.05   ::  johnCalls :- \+ alarm.
0.7    ::  maryCalls :- alarm.
0.01   ::  maryCalls :- \+ alarm.
```

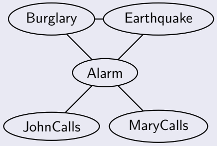
BLOG

```
random Boolean Burglary ~ BooleanDistrib(0.001);
random Boolean Earthquake ~ BooleanDistrib(0.002);
random Boolean Alarm ~
  if Burglary then
    if Earthquake then BooleanDistrib(0.95)
    else BooleanDistrib(0.94)
  else
    if Earthquake then BooleanDistrib(0.29)
    else BooleanDistrib(0.001);
random Boolean JohnCalls ~
  if Alarm then BooleanDistrib(0.9)
  else BooleanDistrib(0.05);
random Boolean MaryCalls ~
  if Alarm then BooleanDistrib(0.7)
  else BooleanDistrib(0.01);
```

Bayesian Network



Markov Random Field



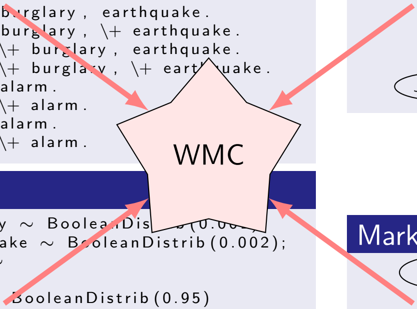
The Problem of Computing Probability

ProbLog

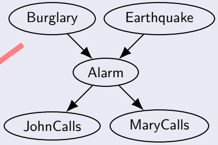
```
0.001  ::  burglary.
0.002  ::  earthquake.
0.95   ::  alarm      :- burglary, earthquake.
0.94   ::  alarm      :- burglary, \+ earthquake.
0.29   ::  alarm      :- \+ burglary, earthquake.
0.001  ::  alarm      :- \+ burglary, \+ earthquake.
0.9    ::  johnCalls  :- alarm.
0.05   ::  johnCalls  :- \+ alarm.
0.7    ::  maryCalls  :- alarm.
0.01   ::  maryCalls  :- \+ alarm.
```

BLOG

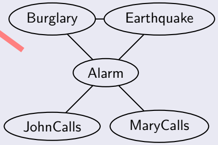
```
random Boolean Burglary ~ BooleanDistrib(0.001);
random Boolean Earthquake ~ BooleanDistrib(0.002);
random Boolean Alarm ~
  if Burglary then
    if Earthquake then BooleanDistrib(0.95)
    else BooleanDistrib(0.94)
  else
    if Earthquake then BooleanDistrib(0.29)
    else BooleanDistrib(0.001);
random Boolean JohnCalls ~
  if Alarm then BooleanDistrib(0.9)
  else BooleanDistrib(0.05);
random Boolean MaryCalls ~
  if Alarm then BooleanDistrib(0.7)
  else BooleanDistrib(0.01);
```



Bayesian Network



Markov Random Field



Weighted Model Counting (WMC)

- Generalises propositional model counting (#SAT)
- Applications:
  - graphical models
  - probabilistic programming
  - neural-symbolic artificial intelligence
- Main types of algorithms:
  - using knowledge compilation
  - using a SAT solver
  - manipulating pseudo-Boolean functions

Example

$w(x) = 0.3,\ w(\neg x) = 0.7,$
$w(y) = 0.2,\ w(\neg y) = 0.8$

$\mathrm{WMC}(x \lor y) = w(x)w(y) +$
$w(x)w(\neg y) + w(\neg x)w(y) = 0.44$

Outline

Outline

An Alternative Way to Think About WMC

- Let V be the set of variables.
- Then 2^{2^V} is the Boolean algebra of propositional formulas.

Definition

A measure is a function $\mu\colon 2^{2^V} \to \mathbb{R}_{\geq 0}$ such that:

- $\mu(\bot) = 0$;
- $\mu(x \vee y) = \mu(x) + \mu(y)$ whenever $x \wedge y = \bot$.

Observation

WMC corresponds to the process of calculating the value of $\mu(x)$ for some $x \in 2^{2^V}$.

The Limitations and Capabilities of WMC

Observation

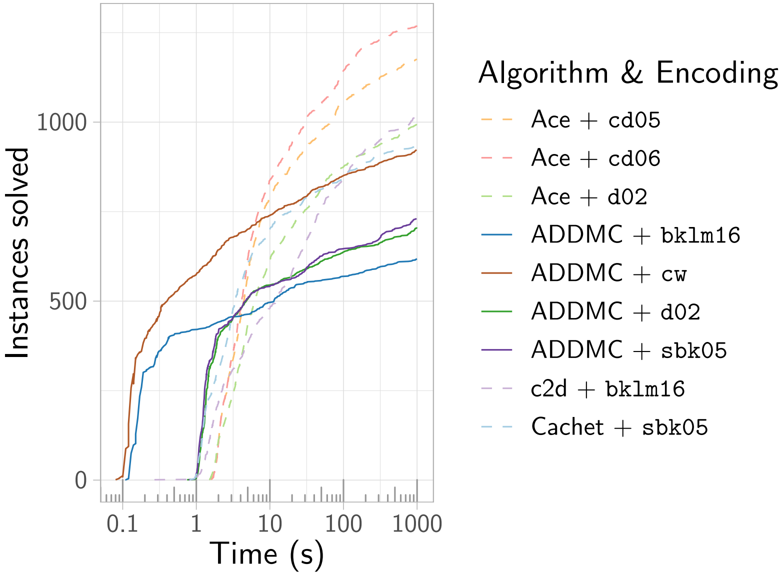
Classical WMC is only able to evaluate factorable measures (c.f., a collection of mutually independent random variables).

Theorem (Informal Version)

It is always possible to add more variables to turn a non-factorable measure into a factorable measure.

However, that is not necessarily a good idea!

Experimental Results



Outline

Formalising the Intuition from Before

For any propositional formula ϕ over a set of variables X and $p, q \in \mathbb{R}$, let $[\phi]_q^p \colon 2^X \to \mathbb{R}$ be the pseudo-Boolean function defined as

$$[\phi]_q^p(Y) \coloneqq \begin{cases} p & \text{if } Y \models \phi \\ q & \text{otherwise} \end{cases}$$

for any $Y \subseteq X$.

Definition (Pseudo-Boolean Projection (PBP))

A PBP instance is a tuple (F, X, ω), where X is the set of variables, F is a set of two-valued pseudo-Boolean functions $2^X \to \mathbb{R}$, and $\omega \in \mathbb{R}$ is the scaling factor.

From WMC to PBP

The WMC instance has x as the only indicator variable and p, q as parameter variables with weights $w(p) = 0.2$, $w(q) = 0.8$, and $w(\neg p) = w(\neg q) = 1$.

WMC Clause
$\neg x \Rightarrow p$
$p \Rightarrow \neg x$
$x \Rightarrow q$
$q \Rightarrow x$
$\neg x$

From WMC to PBP

The WMC instance has x as the only indicator variable and p, q as parameter variables with weights $w(p) = 0.2$, $w(q) = 0.8$, and $w(\neg p) = w(\neg q) = 1$.

WMC Clause	In CNF
$\neg x \Rightarrow p$	$x \vee p$
$p \Rightarrow \neg x$	$\neg x \vee \neg p$
$x \Rightarrow q$	$\neg x \vee q$
$q \Rightarrow x$	$x \vee \neg q$
$\neg x$	$\neg x$

From WMC to PBP

The WMC instance has x as the only indicator variable and p, q as parameter variables with weights $w(p) = 0.2$, $w(q) = 0.8$, and $w(\neg p) = w(\neg q) = 1$.

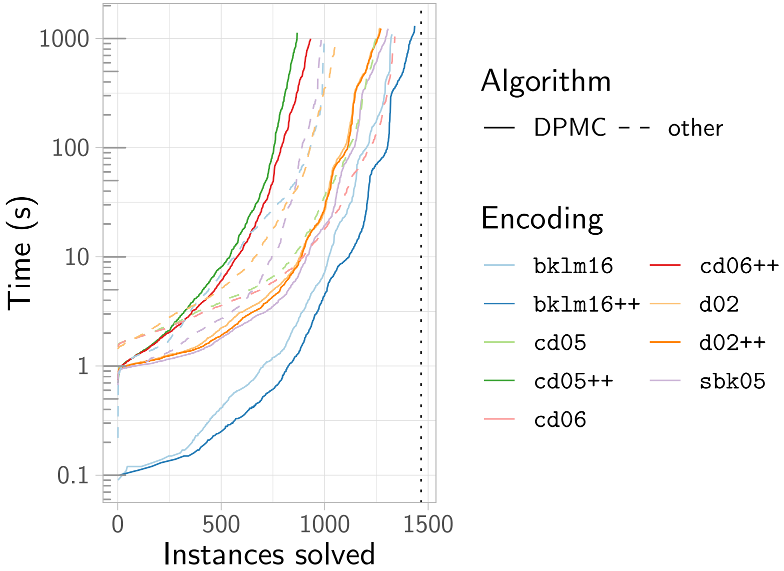
WMC Clause	In CNF	Pseudo-Boolean Function
$\neg x \Rightarrow p$	$x \vee p$	$[\neg x]_1^{0.2}$
$p \Rightarrow \neg x$	$\neg x \vee \neg p$	
$x \Rightarrow q$	$\neg x \vee q$	$[x]_1^{0.8}$
$q \Rightarrow x$	$x \vee \neg q$	
$\neg x$	$\neg x$	$[\neg x]_0^1$

From WMC to PBP

The WMC instance has x as the only indicator variable and p, q as parameter variables with weights $w(p) = 0.2$, $w(q) = 0.8$, and $w(\neg p) = w(\neg q) = 1$.

WMC Clause	In CNF	Pseudo-Boolean Function	
$\neg x \Rightarrow p$	$x \vee p$	$[\neg x]_1^{0.2}$	
$p \Rightarrow \neg x$	$\neg x \vee \neg p$		$[x]_{0.2}^{0.8}$
$x \Rightarrow q$	$\neg x \vee q$	$[x]_1^{0.8}$	
$q \Rightarrow x$	$x \vee \neg q$		
$\neg x$	$\neg x$	$[\neg x]_0^1$	$[\neg x]_0^1$

Experimental Results



Outline

Plan for the Next Paper

Parameterized Complexity

Cachet:	$2^{\mathcal{O}(w_b)} n^{\mathcal{O}(1)}$ (n is the number of variables, and w_b is the branch width).
c2d:	$\mathcal{O}(mw2^w)$ (m is the number of clauses, and w is the primal treewidth).
DPMC:	$\mathcal{O}(4^w nm)$ (my result, and the 4^w part is tight).

Empirical Study on Random Instances

- New random model for 3-CNF formulas (done)
- Running experiments (almost done)
- Plots and writing (some disorganised notes)

Outline

How Everything Fits Together

- My thesis is centered around two ideas:
  - Manipulating more expressive representations can lead to more efficient algorithms (c.f., cutting planes vs. resolution in SAT).
  - Random problem instances can help reveal fundamental differences in how algorithms behave in practice.
- UAI'21 and SAT'21 papers address the first idea.
  - There is no reason for weights to only be defined on literals (and there is no reason for a clause to be a just clause).
- CP'20 and the next paper undertake the second idea.
  - Random probabilistic logic programs and random formulas in CNF.
- Future work: perhaps tackle the first-order setting.