Approximating Probabilistic Inference without Losing Guarantees: Combining Hashing and Feasibility

Kuldeep S. Meel
PhD Student
CAVR Group

Joint work with Supratik Chakraborty, Daniel J. Fremont, Sanjit A. Seshia, Moshe Y. Vardi
IoT: Internet of Things

The Internet of Things

Connected Objects
The Era of Data

- How to make sense of data?
- Modeling the events
- Infer likelihood from data
Probabilistic Inference

Given that a Rice CS grad student queried “music” on google, what is the probability they will click on “The best of Justin Beiber”?

Pr [event|evid]
Probabilistic Inference

Given that a Rice CS grad student queried “music” on google, what is the probability they will click on “The best of Justin Beiber”?

Pr [event|evid]
Probabilistic Inference

Given that a Rice CS grad student queried “music” on google, what is the probability they will click on “The best of Justin Beiber”?

\[ \Pr[\text{event}|\text{evid}] \]
Graphical Models

Bayesian Networks
What is $\Pr [ \text{Burglary} | \text{Call} ]$?
Bayes’ Rule to the Rescue

\[ Pr[\text{Burglary}|\text{Call}] = \frac{Pr[\text{Burglary } \cap \text{Call}]}{Pr[\text{Call}]} \]

\[ Pr[\text{Burglary } \cap \text{Call}] = Pr[B, E, A, M, P, C] + Pr[B, \bar{E}, A, M, P, C] + \cdots \]
P(B, \bar{E}, A, M, P, C) = \text{Pr}[B, \bar{E}, A, M, P, C]

\begin{array}{|c|c|c|c|}
\hline
A & M & Pr \\
\hline
T & T & 0.7 \\
T & F & 0.3 \\
F & T & 0.1 \\
F & F & 0.9 \\
\hline
\end{array}

\begin{array}{|c|c|c|c|}
\hline
B & E & A & Pr \\
\hline
T & T & T & 0.88 \\
T & T & F & 0.12 \\
T & F & T & 0.91 \\
T & F & F & 0.09 \\
F & T & T & 0.97 \\
F & T & F & 0.03 \\
F & F & T & 0.02 \\
F & F & F & 0.98 \\
\hline
\end{array}

\begin{array}{|c|c|c|c|}
\hline
M & P & C & Pr \\
\hline
T & T & T & 0.99 \\
T & T & F & 0.01 \\
T & F & T & 0 \\
T & F & F & 1 \\
F & T & T & 0 \\
F & T & F & 1 \\
F & F & T & 0.1 \\
F & F & F & 0.9 \\
\hline
\end{array}
Our Contribution

Scalability

Guarantees

Approximation Guarantees

WeightMC

Exact Methods

BP, MCMC
A wild Idea for a new paradigm?

• Partition the space of paths into “small” “equal weighted” cells

  • “Small”: # of paths in a cell is not large (bounded by a constant)

  • “equal weighted”: All the cells have equal weight
Outline

• Reduction to SAT

• Partition-based techniques via (unweighted) model counting

• Extension to Weighted Model Counting

• Looking forward
Boolean Satisfiability

- **SAT**: Given a Boolean formula $F$ over variables $V$, determine if $F$ is true for some assignment to $V$

- $F = (a \lor b)$

- $R_F = \{(0,1),(1,0),(1,1)\}$

- SAT is NP-Complete (Cook 1971)
  - One of the million dollar problems
Model Counting

Given:
- CNF Formula $F$, Solution Space: $R_F$

Problem (MC):
What is the total number of satisfying assignments (models) i.e. $|R_F|$?

Example
$F = (a \lor b)$; $R_F = \{[0,1], [1,0], [1,1]\}$

$|R_F| = 3$
Weighted Model Counting

Given:
- CNF Formula $F$, Solution Space: $R_F$
- Weight Function $W(.)$ over assignments
  - $W(\sigma)$

Problem (WMC):
What is the sum of weights of satisfying assignments i.e. $W(R_F)$?

Example
$F = (a \lor b)$; $R_F = \{[0,1], [1,0], [1,1]\}$

$W([0,1]) = W([1,0]) = \frac{1}{3} \quad W([1,1]) = W([0,0]) = \frac{1}{6}$

$W(R_F) = \frac{1}{3} + \frac{1}{3} + \frac{1}{6} = \frac{5}{6}$
Weighted SAT

- Boolean formula $F$
- Weight function over variables (literals)
- Weight of assignment = product of wt of literals

- $F = (a \lor b)$; $W(a=0) = 0.4$; $W(a = 1) = 1-0.4 = 0.6$
  $W(b=0) = 0.3$; $W(b = 1) = 0.7$

- $W[(0,1)] = W(a = 0) \times W(b = 1) = 0.4 \times 0.7 = 0.28$
## Reduction to W-SAT

<table>
<thead>
<tr>
<th>Bayesian Network</th>
<th>SAT Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>Variables</td>
</tr>
<tr>
<td>Rows of CPT</td>
<td>Variables</td>
</tr>
<tr>
<td>Probabilities in CPT</td>
<td>Weights</td>
</tr>
<tr>
<td>Event and Evidence</td>
<td>Constraints</td>
</tr>
</tbody>
</table>
Reduction to W-SAT

- Every satisfying assignment = A valid path in the network
  - Satisfies the constraint (evidence)
  - Probability of path = Weight of satisfying assignment = Product of weight of literals = Product of conditional probabilities
  - Sum of probabilities = Weighted Sum
Why SAT?

• SAT stopped being NP-complete in practice!
• zchaff (Malik, 2001) started the SAT revolution
• SAT solvers follow Moore’s law
Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout
Why SAT?

- SAT stopped being NP-complete in practice!
- zchaff (Malik, 2001) started the SAT revolution
- SAT solvers follow Moore’s law
- “Symbolic Model Checking without BDDs”: most influential paper in the first 20 years of TACAS
- A simple input/output interface
Outline

• Reduction to SAT

• **Partition-based techniques** via (unweighted) model counting

• Extension to Weighted Model Counting

• Looking forward
Counting through Partitioning
Counting through Partitioning
Counting through Partitioning

Pick a random cell

Estimated Total # of solutions = #solutions in the cell * total # of cells
How to Partition?

How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

Universal Hashing
[Carter-Wegman 1979, Sipser 1983]
Universal Hashing

- Hash functions from mapping \(\{0,1\}^n\) to \(\{0,1\}^m\)
  - \((2^n \text{ elements to } 2^m \text{ cells})\)

- Random inputs => All cells are *roughly* equal in expectation

- Universal hash functions:
  - For any distribution) inputs => All cells are *roughly* equal in expectation
Universal Hashing

- Hash functions from mapping \(\{0,1\}^n\) to \(\{0,1\}^m\)
  - \((2^n\) elements to \(2^m\) cells)

- Random inputs \(\Rightarrow\) All cells are \textit{roughly} equal in expectation

- Universal hash functions:
  - For any distribution) inputs \(\Rightarrow\) All cells are \textit{roughly} equal in expectation

- Need stronger bounds on range of the size of cells
Lower Universality $\Rightarrow$ Lower Complexity

- $H(n,m,r)$: Family of $r$-universal hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$ ($2^n$ elements to $2^m$ cells)

- Higher the $r$ $\Rightarrow$ Stronger guarantees on variance of size of cells

- $r$-wise universality $\Rightarrow$ Polynomials of degree $r-1$

- Lower universality $\Rightarrow$ lower complexity
XOR-Based Hashing

- 3-universal hashing
- Partition $2^n$ space into $2^m$ cells
- Variables: $X_1, X_2, X_3, \ldots, X_n$
- Pick every variable with prob. $\frac{1}{2}$, XOR them and equate to 0/1 with prob. $\frac{1}{2}$
- $X_1 + X_3 + X_6 + \ldots + X_{n-1} = 0$
- $m$ XOR equations $\rightarrow 2^m$ cells
Counting through Partitioning
Partitioning

- How large should the cells be?

- How many cells?
Size of cell

- Too large => Hard to enumerate
- Too small => Variance can be very high
- More tight bounds => larger cell

\[
pivot = 5\left(1 + \frac{1}{\varepsilon}\right)^2
\]
ApproxMC
ApproxMC

\[ R_F \]

\[ \text{#of sols} < \text{pivot} \]

NO
ApproxMC

\[ F \land \text{XOR}_1 \]
ApproxMC

\[ \text{No of sols < pivot} \]
ApproxMC

Estimated Count =
# of solution in the cell X
$2^m$

F $\wedge$ XOR$_1$ $\wedge$ XOR$_2$ $\ldots\ldots$ $\wedge$ XOR$_m$
ApproxMC

Partitioning

690 710 730 730 731 831 .............. 834
ApproxMC in Action

Median

Algorithm

690  710  730  730  731  831  ...........  834

t
ApproxMC (CNF: F, tolerance: $\varepsilon$, confidence: $\delta$)

Suppose ApproxMC($F, \varepsilon, \delta$) returns $C$. Then,

$$\Pr \left[ \frac{\#F}{1+\varepsilon} \leq C \leq (1+\varepsilon) \#F \right] \geq \delta$$

ApproxMC runs in time polynomial in $\log (1-\delta)^{-1}$, $|F|$, $\varepsilon^{-1}$ relative to SAT oracle.
Key Idea behind the Proof

Let $I_1, I_2, I_3, \cdots I_n$ be 3-wise independent variables in $[0, 1]$, then for $I = \sum I_k$, $\mu = E[I]$

\[ Pr[|I - \mu| < \beta \mu] \geq 0.7 \]

$I_k = 1$ if $y_k$ is in the cell

\[ Pr[I_k = 1] = 1/2^m \quad \mu = \frac{R_F}{2m} \]

\[ Pr[\#F/(1+\varepsilon) \leq C \leq (1+\varepsilon) \#F] \geq 0.7 \]
Can Solve a Large Class of Problems

Large class of problems that lie beyond the exact counters but can be computed by ApproxMC.
Outline

• Reduction to SAT

• Partition-based techniques via (unweighted) model counting

• Extension to Weighted Model Counting

• Looking forward
Partitioning into equal (weighted) “small” cells
Partitioning into equal (weighted) “small” cells

Pick a random cell

Estimated Weighted Count = Weighted Count of cell $X \times$ # of cells
Key Modifications

Let $I_1, I_2, I_3, \ldots, I_n$ be 3 - wise independent variables in $[0, 1]$, then for $I = \sum I_k$, $\mu = E[I]$

$$Pr[|I - \mu| < \beta \mu] \geq 0.7$$

$$I_k = \frac{w(y_k)}{w_{max}} \text{ if } y_k \text{ is in the cell}$$

$$\mu = \frac{W(R_F)}{2^m} \text{ # of solutions in a cell} < \frac{w_{max}}{w_{min}} \text{ pivot}$$

$$\rho = \frac{w_{max}}{w_{min}}$$
Strong Theoretical Guarantees

- **Approximation:** \( \text{WeightMC}(B, \epsilon, \delta) \), returns \( C \) s.t.
  \[
  Pr\left[ \frac{f}{1 + \epsilon} \leq C \leq f(1 + \epsilon) \right] \geq 1 - \delta
  \]

- **Complexity:** # of calls to SAT solver is linear in
  \( \log \delta^{-1}, |F|, 1/\epsilon \)
  and polynomial in
Handling Large Tilt

Tilt: 992
Handling Large Tilt

Requires Pseudo-Boolean solver: Still a SAT problem not Optimization

Tilt: 992
Tilt for each region: 2
Main Contributions

- Novel parameter, tilt ($\rho$), to characterize complexity
  - $\rho = \frac{W_{\text{max}}}{W_{\text{min}}}$ over satisfying assignments
- Small Tilt ($\rho$)
  - Efficient hashing-based technique requires only SAT solver
- Large Tilt ($\rho$)
  - Divide-and-conquer using Pseudo-Boolean solver
Strong Theoretical Guarantees

- **Approximation:** \text{WeightMC}(B, \epsilon, \delta), returns C s.t.
  \[ Pr\left[ \frac{f}{1 + \epsilon} \leq C \leq f(1 + \epsilon) \right] \geq 1 - \delta \]

- **Complexity:** # of calls to SAT solver is linear in
  \[ \log \rho, \log \delta^{-1}, |F|, 1/\epsilon \]
  and polynomial in
Significantly Faster than SDD

Run Time (seconds)

Benchmarks

WeightMC
SDD

# of variables ——>
Mean Error: 4% (Allowed: 80%)
Outline

- Reduction to SAT
- Partition-based techniques via (unweighted) model counting
- Extension to Weighted Model Counting
- Looking forward
Distribution-Aware Sampling

Given:
- CNF Formula F, Solution Space: $R_F$
- Weight Function $W(.)$ over assignments
  - $W(\sigma)$

Problem (Sampling):
Pr (Solution $y$ is generated) = $W(y)/W(R_F)$

Example:
$F = (a \lor b)$; $R_F = \{[0,1], [1,0], [1,1]\}$

$W([0,1]) = W([1,0]) = 1/3$ \hspace{1cm} $W([1,1]) = W([0,0]) = 1/6$

Pr ([0,1] is generated] = (1/3) / (5/6) = 2/5
Partitioning into equal (weighted) “small” cells
Partitioning into equal (weighted) “small” cells

Pick a random cell

Pick a solution according to its weight
• Benchmark: case110.cnf;  \#var: 287; \#clauses: 1263
• Total Runs: $4 \times 10^6$; Total Solutions : 16384
Tackling Tilt

• What kind of problems have low tilt?
• How to handle CNF+PBO+XOR
  • Current PBO solvers can’t handle XOR
  • CMS can’t handle PBO queries
Extension to More Expressive Domains (SMT, CSP)

- Efficient 3-independent hashing schemes
  - Extending bit-wise XOR to SMT provides guarantees but no advantage of SMT progress

- Solvers to handle F + Hash efficiently
  - CryptoMiniSAT has fueled progress for SAT domain
  - Similar solvers for other domains?
Collaborators
EXTRA SLIDES
Complexity

- Tilt captures the ability of hiding a large weight solution.

- Is it possible to remove tilt from complexity?
Exploring CNF+XOR

- Very little understanding as of now
- Can we observe phase transition?
- Eager/Lazy approach for XORs?
- How to reduce size of XORs further?
Outline

• Reduction to SAT

• Partition-based techniques via (unweighted) model counting

• Extension to Weighted Model Counting

• Discussion on hashing

• Looking forward
XOR-Based Hashing

- 3-universal hashing
- Partition $2^n$ space into $2^m$ cells
- Variables: $X_1, X_2, X_3, \ldots, X_n$
- Pick every variable with prob. $\frac{1}{2}$, XOR them and equate to 0/1 with prob. $\frac{1}{2}$
- $X_1 + X_3 + X_6 + \ldots + X_{n-1} = 0$ (Cell ID: 0/1)
- $m$ XOR equations -> $2^m$ cells
- The cell: $F$ && XOR (CNF+XOR)
XOR-Based Hashing

- CryptoMiniSAT: Efficient for CNF+XOR
- Avg Length : n/2
- Smaller the XORs, better the performance

How to shorten XOR clauses?
Independent Variables

- Set of variables such that assignments to these uniquely determine assignments to rest of variables for formula to be true

- \((a \lor b = c) \Rightarrow \) Independent Support: \(\{a, b\}\)

- # of auxiliary variables introduced: 2-3 orders of magnitude

- Hash only on the independent variables (huge speedup)