A Scalable Approximate Model Counter

Supratik Chakraborty\textsuperscript{1}, Kuldeep S Meel\textsuperscript{2}, Moshe Y Vardi\textsuperscript{2}

\textsuperscript{1}Indian Institute of Technology Bombay, India
\textsuperscript{2}Department of Computer Science, Rice University
What is Model Counting?

- Given a SAT formula $F$
- $R_F$: Set of all solutions of $F$
- Problem (\#SAT): Estimate the number of solutions of $F$ ($\#F$) i.e., what is the cardinality of $R_F$?
- E.g., $F = (a \lor b)$
- $R_F = \{(0,1), (1,0), (1,1)\}$
- The number of solutions ($\#F$) = 3

\#P: The class of counting problems for decision problems in NP!
Practical Applications

Exciting range of applications!

- Probabilistic reasoning/Bayesian inference
- Planning with uncertainty
- Multi-agent/ adversarial reasoning

[Roth 96, Sang 04, Bacchus 04, Domshlak 07]
But it is hard!

- #SAT is #P-complete
  - Even for counting solutions of 2-CNF SAT

- #P is really hard!
  - Believed to be much harder than NP-complete problems
  - $\text{PH} \subseteq \text{P}^\#P$
The Hardness of Model Counting

![Graph showing the hardness of model counting benchmarks with Cachet]
Can we do better?

Approximate counting (with guarantees) suffices for most of the applications
# Prior Work

**Input Formula:** $F$;  **Total Solutions:** $\#F$;  **Return Value:** $C$

<table>
<thead>
<tr>
<th>Counters</th>
<th>Guarantee</th>
<th>Confidence</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact counter</td>
<td>$C = #F$</td>
<td>1</td>
<td>Poor Scalability</td>
</tr>
<tr>
<td>(e.g. sharpSAT, Cachet)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower bound counters</td>
<td>$C \leq #F$</td>
<td>$\delta$</td>
<td>Very weak guarantees</td>
</tr>
<tr>
<td>(e.g. MBound, SampleCount)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper bound counters</td>
<td>$C \geq #F$</td>
<td>$\delta$</td>
<td>Very weak guarantees</td>
</tr>
<tr>
<td>(e.g. MiniCount)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Approximate Model Counting

Design an approximate model counter $G$:

- inputs:
  - CNF formula $F$
  - tolerance $\varepsilon$
  - confidence $\delta$

- the count returned by it is within $\varepsilon$ of the $\#F$ with confidence at least $\delta$
Approximate Model Counting

Design an approximate model counter $G$:

- inputs:
  - CNF formula $F$
  - tolerance $\varepsilon$
  - confidence $\delta$

- the count returned by it is within $\varepsilon$ of the $\#F$ with confidence at least $\delta$ and scales to real world problems

Scalable Approximate Model Counting

Lies in the 2nd level of Polynomial hierarchy: $\Sigma_2^P$
## Our Contribution

**Input Formula:** $F$;  **Total Solutions:** $\#F$

<table>
<thead>
<tr>
<th>Counters</th>
<th>Guarantee</th>
<th>Confidence</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact counter (e.g. sharpSAT, Cachet)</td>
<td>$C = #F$</td>
<td>1</td>
<td>Poor Scalability</td>
</tr>
<tr>
<td>ApproxMC</td>
<td>$\frac{#F}{1+\varepsilon} \leq C \leq (1+\varepsilon) #F$</td>
<td>$\delta$</td>
<td>Scalability + Strong guarantees</td>
</tr>
</tbody>
</table>
Overview

- Our approach
- Theoretical results
- Experimental results
- Where do we go from here?
How do we count?
Explicit Enumeration: Not Scalable

- Enumerate (almost) all solutions
- Exact Counting!
- Cachet, Relsat, sharpSAT

Not Scalable!
Counting through Partitioning
Counting through Partitioning

Pick a random cell

Total # of solutions = # solutions in the cell * total # of cells
Algorithm in Action
Algorithm in Action

Median

Algorithm

690  710  730  730  731  831  ..............  834
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 elements to 2^m cells)

- Random inputs => All cells are *roughly* small

- Universal hash functions:
  - Adversarial (any distribution) inputs => All cells are *roughly* small

- Need stronger bounds on range of the size of cells
Higher Universality  ➡️  Stronger Guarantees

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

- Higher the $r$ ➞ Stronger guarantees on range of size of cells

- $r$-wise universality ➞ Polynomials of degree $r-1$

- Lower universality ➞ lower complexity
Highlights of Our Hashing

- Employs XOR-based hash functions instead of computationally infeasible algebraic hash functions

- Uses off-the-shelf SAT solver CryptoMiniSAT (MiniSAT+XOR support)
Strong Theoretical Results

**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.
Experimental Methodology

- **Benchmarks (over 200)**
  - Grid networks, DQMR networks, Bayesian networks
  - Plan recognition, logistics problems
  - Circuit synthesis

- **Tolerance**: $\varepsilon = 0.75$, **Confidence**: $\delta = 0.9$

- **Objectives**
  - Comparison with exact counters (Cachet) & bounding counters (MiniCount, Hybrid-MBound, SampleCount)
    - Performance
    - Quality of bounds
Results: Performance Comparison

![Graph comparing ApproxMC and Cachet performance](image-url)
Results: Performance Comparison

ApproxMC
Cachet
Can Solve a Large Class of Problems

Large class of problems that lie beyond the exact counters but can be computed by ApproxMC
Mean Error: Only 4% (allowed: 75%)

Mean error: 4% – much smaller than the theoretical guarantee of 75%
Results: Bounding Counters

- Range of count from bounding counters = $C_2 - C_1$
  - $C_1$: From lower bound counters ($MBound$/$SampleSAT$)
  - $C_2$: From upper bound counters ($MiniCount$)

- Range from ApproxMC: $[C/(1+\varepsilon), (1+\varepsilon)C]$
Better Bounds Than Existing Counters

ApproxMC improved the upper bounds significantly while also improving the lower bounds.
Key Takeaways

- Many practical applications can be reduced to (approximate) model counting
- ApproxMC is the first scalable approximate model counter
- Uses easy-to-implement linear hash functions
- Major improvements in performance and quality of bounds compared to existing counters.
Where do we go from here?

- Ongoing work: Probabilistic Inference
- Further scaling: Efficient hash functions
- Extension to CSP and SMT domains
Acknowledgments

- NSF
- ExCAPE
- Intel
- BRNS, India
- Sun Microsystems
- Sigma Solutions, Inc

Thank You for your attention!