The Secrets of GANAK: Designing Scalable Exact Model Counter

Kuldeep S. Meel\textsuperscript{1}

Joint work with Shubham Sharma\textsuperscript{2}, Mate Soos\textsuperscript{1}, and Subhajit Roy\textsuperscript{2}

\textsuperscript{1}National University of Singapore
\textsuperscript{2}Indian Institute of Technology Kanpur, India
Ganak+ApproxMC won two out of three tracks at Model Counting Competition. Related Paper: IJCAI 2019
• Given:
  – Propositional formula $F$ (CNF) over a set of variables $X$

• Propositional Model Counting ($\#\text{SAT}$):
  – Compute the number of satisfying assignments of $F$

• $\#\text{SAT}$ is $\#P$ complete problem
• Probabilistic Exact Model Counting
  – Given a propositional formula $F$ (CNF) and confidence $\delta \in (0, 1]$, counter returns $\text{count}$ such that:

$$\text{Pr}[|\text{Solutions of } F | = \text{count}] \geq 1 - \delta$$
Propositional Model Counting

• Probabilistic Exact Model Counting
  – Given a propositional formula $F$ (CNF) and confidence $\delta \in (0, 1]$, counter returns $\text{count}$ such that:

  $$\Pr[|\text{Solutions of } F| = \text{count}] \geq 1 - \delta$$

• Probabilistic Exact Model Counting is almost as hard as Exact Model Counting$^1$
Long Line of Work

Knowledge Compilation  c2d [Darwiche, 2004], D4[Lagniez and Marquis, 2017]

Search-based Counters  Cachet[Sang et al, 2004; 2005], sharpSAT [Thurley 2006]

Main Ingredients of Search-Based Counters

- Decision Process:
  - $(F \land I) \lor (F \land \neg I)$ mutually inconsistent
  - $\#(F) = \#(F \land I) + \#(F \land \neg I)$

- Component Decomposition:
  - $F = \Delta_1 \land \Delta_2 \cdots \Delta_n$
  - $\Delta_1 \cdots \Delta_n$ does not share any variables
  - $\#(F) = \#(\Delta_1) \times \#(\Delta_2) \cdots \times \#(\Delta_n)$ mutually disjoint
Main Ingredients of Search-Based Counters

- **Decision Process:**
  - \((F \land I) \lor (F \land \neg I)\) mutually inconsistent
  - \(#(F) = #(F \land I) + #(F \land \neg I)\)

- **Component Decomposition:**
  - \(F = \Delta_1 \land \Delta_2 \cdots \Delta_n\) \(\Delta_1 \cdots \Delta_n\) does not share any variables
  - \(#(F) = #(\Delta_1) \times #(\Delta_2) \cdots \times #(\Delta_n)\) mutually disjoint
Main Ingredients of Search-Based Counters

• Decision Process:
  - \((F \land I) \lor (F \land \neg I)\) mutually inconsistent
  - \(#(F) = #(F \land I) + #(F \land \neg I)\)

• Component Decomposition:
  - \(F = \Delta_1 \land \Delta_2 \cdots \Delta_n\) \(\Delta_1 \cdots \Delta_n\) does not share any variables
  - \(#(F) = #(\Delta_1) \times #(\Delta_2) \cdots \times #(\Delta_n)\) mutually disjoint

• Conflict Driven Clause Learning
Main Ingredients of Search-Based Counters

- **Decision Process:**
  - \((F \land I) \lor (F \land \neg I)\) mutually inconsistent
  - \(#(F) = #(F \land I) + #(F \land \neg I)\)

- **Component Decomposition:**
  - \(F = \Delta_1 \land \Delta_2 \cdots \Delta_n\) \(\Delta_1 \cdots \Delta_n\) does not share any variables
  - \(#(F) = #(\Delta_1) \times #(\Delta_2) \cdots \times #(\Delta_n)\) mutually disjoint

- **Conflict Driven Clause Learning**

- **Component Caching:**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta_1)</td>
<td>#((\Delta_1))</td>
</tr>
<tr>
<td>(\Delta_2)</td>
<td>#((\Delta_2))</td>
</tr>
</tbody>
</table>
Model Counting Algorithm

1: \( l \leftarrow \text{DecideLiteral}(F) \)
2: for \( l \leftarrow \{l, \neg l\} \) do
3: \( F_{|l} \leftarrow \text{UnitPropagation}(F, l) \)
4: if \( F_{|l} \) contains an empty clause then
5: \( \text{count}[l] \leftarrow 0 \)
6: else
7: \( \text{count}[l] \leftarrow 1 \)
8: \( \text{Comps} \leftarrow \text{DisjointComponents}(F_{|l}) \quad \triangleright \text{Decomposition} \)
9: for \( C \leftarrow \text{Comps} \) do
10: \( \text{count} \leftarrow \text{GetCache}(C) \)
11: if \( \text{count} = \text{NOT FOUND} \) then
12: \( \text{count} \leftarrow \text{Counter}(C) \)
13: \( \text{count}[l] = \text{count}[l] \times \text{count} \)
14: if \( \text{count} = 0 \) then
15: \( \text{break} \)
16: \( \text{CacheStore}(F, \text{count}[l] + \text{count}[\neg l]) \)
17: return \( \text{count}[l] + \text{count}[\neg l] \)
Example

\[ F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3) \]
Example

\[ F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3) \]

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x_2 \lor x_3))</td>
<td>3</td>
</tr>
<tr>
<td>((x_2 \lor x_3){x_4, x_5})</td>
<td>12</td>
</tr>
</tbody>
</table>
Example

\[ F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3) \]

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x_2 \lor x_3))</td>
<td>3</td>
</tr>
<tr>
<td>((x_2 \lor x_3){x_4, x_5})</td>
<td>12</td>
</tr>
</tbody>
</table>
Example

\[ F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3) \]

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x_2 \lor x_3))</td>
<td>3</td>
</tr>
<tr>
<td>((x_2 \lor x_3){x_4, x_5})</td>
<td>12</td>
</tr>
<tr>
<td>((x_4 \lor x_5))</td>
<td>3</td>
</tr>
</tbody>
</table>
Example

\[ F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3) \]
Example

\[ F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3) \]

---

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x_2 \lor x_3))</td>
<td>3</td>
</tr>
<tr>
<td>((x_2 \lor x_3){x_4, x_5})</td>
<td>12</td>
</tr>
<tr>
<td>((x_4 \lor x_5))</td>
<td>3</td>
</tr>
<tr>
<td>((x_2 \lor x_3) \land (x_4 \lor x_5))</td>
<td>9</td>
</tr>
<tr>
<td>(F = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_2 \lor x_3))</td>
<td>21</td>
</tr>
</tbody>
</table>
Our Contribution

1. Probabilistic Component Caching (PCC)
2. Variable Branching Heuristic (CSVSAADS)
3. Phase Selection Heuristic (PC)
4. Independent Support (IS)
5. Learn and Start Over (LSO)
\[ F = (\neg x_3 \lor \neg x_5 \lor x_6) \land (\neg x_1 \lor x_4 \lor \neg x_6) \land (x_2 \lor x_3 \lor x_6) \]

<table>
<thead>
<tr>
<th>Schema</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>STD^2</td>
<td>-3, -5, 6, 0, -1, 4, -6, 0, 2, 3, 6, 0</td>
<td>#(F)</td>
</tr>
<tr>
<td>HC^3</td>
<td>1, 2, 3, 4, 5, 6, 1, 2, 3</td>
<td>#(F)</td>
</tr>
<tr>
<td>GANAK</td>
<td>m bit hash of HC/STD clhash: universal hash functions</td>
<td>#(F)</td>
</tr>
</tbody>
</table>
Variable Branching Heuristic (CSVSA)

- \( \text{Score}(\text{VSADS})^4 = p \times \text{Score}(\text{VSIDS}) + q \times \text{Score}(\text{DLCS}) \)
  - VSIDS: Prioritize variables present in recently generated conflict clauses
  - Dynamic Largest Com-bined Sum (DLCS): Prioritize the highest occurring variable in the residual formula
Variable Branching Heuristic (CSVADS)

- $\text{Score}(\text{VSADS})^4 = p \times \text{Score}(\text{VSIDS}) + q \times \text{Score}(\text{DLCS})$
  
  - $\text{VSIDS}$: Prioritize variables present in recently generated conflict clauses
  - Dynamic Largest Com-bined Sum (DLCS): Prioritize the highest occurring variable in the residual formula

- $\text{Score}(\text{CSVADS}) = \alpha \times \text{CacheScore} + \beta \times \text{Score}(\text{VSADS})$
Phase Selection Heuristic (PC)

\[ DLIS = \begin{cases} 
  l & |l| \geq |-l| \\
  \neg l & otherwise
\end{cases} \]
Phase Selection Heuristic (PC)

- \[ DLIS = \begin{cases} 
  l & |l| \geq |\neg l| \\
  \neg l & \text{otherwise} 
\end{cases} \]

- We reduce our trust on DLIS by adding randomness in DLIS if the difference in \(|l|\) and \(|\neg l|\) is not overwhelmingly high.
An independent support, $\mathcal{I} \subseteq \text{Vars}(F)$, is a subset of the support such that if two satisfying assignments $\sigma_1$ and $\sigma_2$ agree on $\mathcal{I}$, then $\sigma_1 = \sigma_2$.
An independent support, $\mathcal{I} \subseteq \text{Vars}(F)$, is a subset of the support such that if two satisfying assignments $\sigma_1$ and $\sigma_2$ agree on $\mathcal{I}$, then $\sigma_1 = \sigma_2$.

Example: $(x \lor \neg y) \land (\neg x \lor y)$, $\mathcal{I} = \{x\}$
Independent Support (IS)

- An independent support, $\mathcal{I} \subseteq \text{Vars}(F)$, is a subset of the support such that if two satisfying assignments $\sigma_1$ and $\sigma_2$ agree on $\mathcal{I}$, then $\sigma_1 = \sigma_2$.
- Example: $(x \lor \neg y) \land (\neg x \lor y) \quad \mathcal{I} = \{x\}$
- We use the MIS$^5$ algorithm for computing the minimal $\mathcal{I}$ for hard instances.
Independent Support (IS)

- An independent support, $\mathcal{I} \subseteq \text{Vars}(F)$, is a subset of the support such that if two satisfying assignments $\sigma_1$ and $\sigma_2$ agree on $\mathcal{I}$, then $\sigma_1 = \sigma_2$.
- Example: $(x \lor \neg y) \land (\neg x \lor y)$ \quad $\mathcal{I} = \{x\}$
- We use the MIS$^5$ algorithm for computing the minimal $\mathcal{I}$ for hard instances
- Perform decision process on variables from $\mathcal{I}$
  1. If residual formula is SAT – model count equal to 1
  2. If residual formula is UNSAT – model count equal to 0
Learn and Start Over (LSO)

- Modern SAT solvers use random restarts aggressively in search of a good variable ordering that can quickly lead to a satisfiable assignment\(^6\)
Learn and Start Over (LSO)

- Modern SAT solvers use random restarts aggressively in search of a good variable ordering that can quickly lead to a satisfiable assignment\(^6\)
- Restart solver after the first 5000 decisions
- Learn from the previous invocation by maintaining all the scores obtained in the previous run to explore different and better ordering of decision variables
• GANAK\textsuperscript{7}: First Scalable Probabilistic Exact Model Counter

• Given a propositional formula $F$ (CNF) and confidence $\delta \in (0, 1]$,
$\text{GANAK}(F, \delta)$ returns $\text{count}$ such that

$$\Pr[|\text{Sol}(F)| = \text{count}] \geq 1 - \delta$$

• Tool is available at: https://github.com/meelgroup/ganak
Experimental Evaluation

- Benchmarks arising from probabilistic reasoning, plan recognition, DQMR networks, ISCAS89 combinatorial circuits, quantified information flow, etc.
Experimental Evaluation

• Benchmarks arising from probabilistic reasoning, plan recognition, DQMR networks, ISCAS89 combinatorial circuits, quantified information flow, etc

• Objectives:
  1. Study the impact of different configurations of heuristics Shubham’s Talk
  2. Study the performance of GANAK with respect to the state-of-the-art model counters
In our experiments, the model count returned by GANAK was equal to the exact model count for all benchmarks.
• GANAK performed best when all the heuristics are turned on
Conclusion

- GANAK demonstrates that #SAT solvers can significantly benefit from probabilistic component caches, especially when ably supported by heuristics like IS, CSVSADS, PC and LSO.
- We believe that the heuristics proposed in this work will also significantly benefit exhaustive DPLL-based knowledge compilation frameworks and related tools (like c2d [Darwiche, 2004], D4 [Lagniez and Marquis, 2017], DSHARP [Muise et al., 2012]).
- Tool is available at: https://github.com/meelgroup/ganak