The Rise of Approximate Model Counting: Beyond Classical Theory and Practice of SAT

Kuldeep S. Meel

National University of Singapore

Beyond Satisfiability

The Amazing Collaborators

S. Akshay (IITB, India), Teodora Baluta (NUS, SG), Fabrizio Biondi (Avast, CZ), Supratik Chakraborty (IITB, India), Alexis de Colnet (NUS, SG), Remi Delannoy (NUS, SG), Jeffrey Dudek (Rice,US), Leonardo Duenas-Osorio (Rice,US), Mike Enescu (Inria, France) Daniel Fremont (UCB, US), Dror Fried (Open U., Israel), Stephan Gocht (Lund U., Sweden), Rahul Gupta (IITK, India), Annelie Heuser (Inria, France), Alexander Ivrii (IBM, Israel), Alexey Ignatiev (IST, Portugal), Axel Legay (UCL, Belgium), Sharad Malik (Princeton, US), Joao Marques Silva (IST, Portugal), Rakesh Mistry (IITB, India), Nina Narodytska ((VMWare, US), Roger Paredes (Rice,US), Yash Pote (NUS, SG), Jean Quilbeuf(Inria, France), Subhajit Roy (IITK, India), Mate Soos (NUS, SG), Prateek Saxena (NUS, SG), Sanjit Seshia (UCB, US), Shubham Sharma (IITK, India), Aditya Shrotri(Rice,US), Moshe Vardi (Rice,US)

Special shout out to Mate Soos, **the** maintainer of ApproxMC and UniGen

Counting

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$

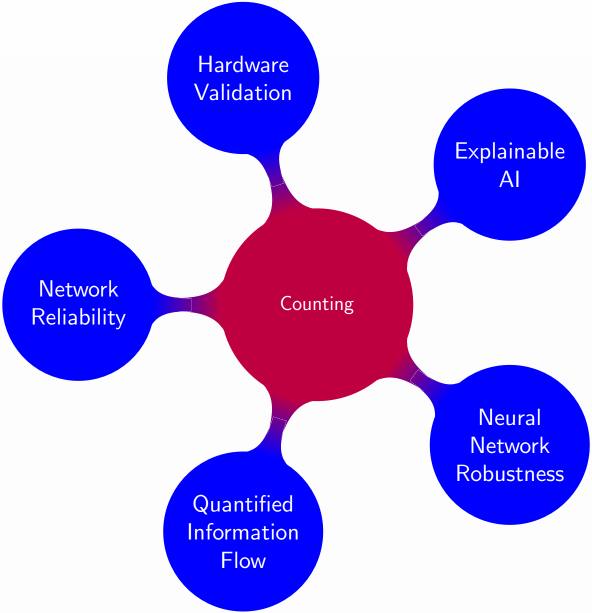
Counting

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- Counting: Determine $|\mathsf{Sol}(F)|$
  - Approximation: $\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq c \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$

Counting

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- Counting: Determine $|\mathsf{Sol}(F)|$
  - Approximation: $\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq c \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$
- Given $F := (X_1 \vee X_2)$
- $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$
- $|\mathsf{Sol}(F)| = 3$

Applications across Computer Science



Through the Lens of SAT Revolution

Obs 1 SAT Oracle $\neq$ NP Oracle

- Returns UNSAT with a proof
- Return a satisfying assignment if satisfiable

Through the Lens of SAT Revolution

Obs 1 SAT Oracle $\neq$ NP Oracle

- Returns UNSAT with a proof
- Return a satisfying assignment if satisfiable

Obs 2 SAT Solver $\neq$ SAT oracle

- The performance of solver depends on the formulas

Through the Lens of SAT Revolution

Obs 1 SAT Oracle $\neq$ NP Oracle

- Returns UNSAT with a proof
- Return a satisfying assignment if satisfiable

Obs 2 SAT Solver $\neq$ SAT oracle

- The performance of solver depends on the formulas

Obs 3 Memoryfulness

- Incremental Solving: Often easier to solve F followed by G if we G can be written as $G = F \wedge H$
- If $F \rightarrow C$ then $(F \wedge H) \implies C$

SAT Oracle vs NP Oracle vs SAT Solver

ThreshSAT(F, thresh): Does F has $\leq$ thresh solutions?
BoundedSAT(F, thresh): $|\mathrm{Sol}(F)|$ If F has $\leq$ thresh solutions, else $\perp$?

- NP Oracle
  - ThreshSAT: #Queries: 1 Size: $|F| \cdot$ thresh
  - BoundedSAT: #Queries: thresh Size: $|F| \cdot$ thresh

SAT Oracle vs NP Oracle vs SAT Solver

ThreshSAT(F, thresh): Does F has $\leq$ thresh solutions?
BoundedSAT(F, thresh): $|\mathsf{Sol}(F)|$ If F has $\leq$ thresh solutions, else $\perp$?

- NP Oracle
  - ThreshSAT: #Queries: 1 Size: $|F| \cdot \mathsf{thresh}$
  - BoundedSAT: #Queries: thresh Size: $|F| \cdot \mathsf{thresh}$
- SAT Oracle
  - ThreshSAT: #Queries: 1 Size: $|F| \cdot \mathsf{thresh}$
  - BoundedSAT: #Queries: thresh Size: $|F| + n \cdot \mathsf{thresh}$

SAT Oracle vs NP Oracle vs SAT Solver

ThreshSAT(F, thresh): Does F has $\leq$ thresh solutions?
BoundedSAT(F, thresh): $|\mathrm{Sol}(F)|$ If F has $\leq$ thresh solutions, else $\perp$?

- NP Oracle
  - ThreshSAT: #Queries: 1 Size: $|F| \cdot \text{thresh}$
  - BoundedSAT: #Queries: thresh Size: $|F| \cdot \text{thresh}$
- SAT Oracle
  - ThreshSAT: #Queries: 1 Size: $|F| \cdot \text{thresh}$
  - BoundedSAT: #Queries: thresh Size: $|F| + n \cdot \text{thresh}$
- SAT Solver
  - ThreshSAT: #Queries: thresh Size: $|F| + n \cdot \text{thresh}$
  - BoundedSAT: #Queries: thresh Size: $|F| + n \cdot \text{thresh}$

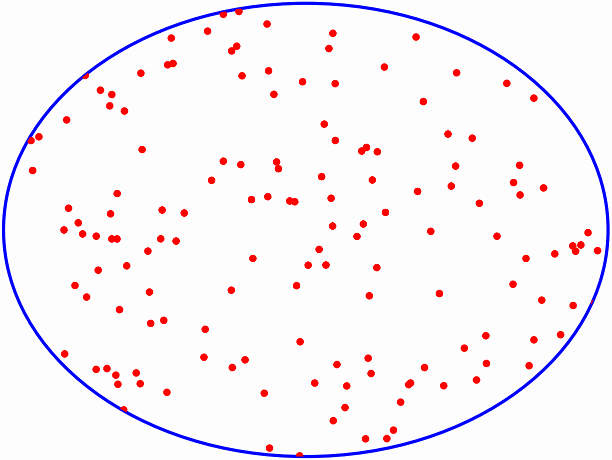
Both ThreshSAT and BoundedSAT have *same complexity*!

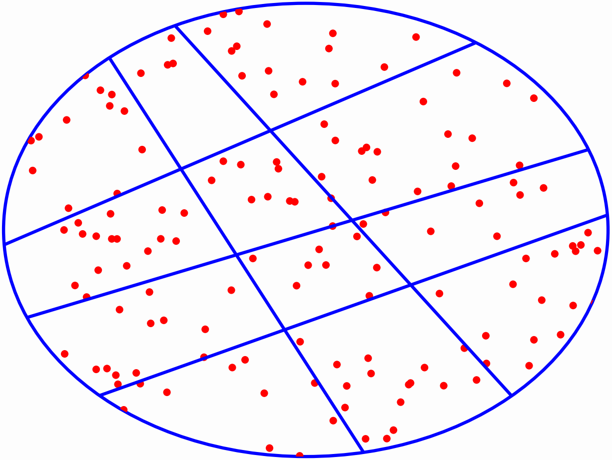
So What Makes Hashing-based Techniques Work?

- Algorithmic
  - From Stockmeyer to ApproxMC
  - The Boon of Dependence
  - Sparse XORs
- System: Efficient CNF+XOR Solving (Soos' possible talk in SAT Seminar?)
- Conceptual
  - Independent Support
  - Projection

The Rise of Hashing-based Approach: Promise of Scalability and Guarantees
(S83,GSS06,GHSS07,CMV13b,EGSS13b,CMV14,CDR15,CMV16,ZCSE16,AD16, KM18,ATD18,SM19,ABM20,SGM20)

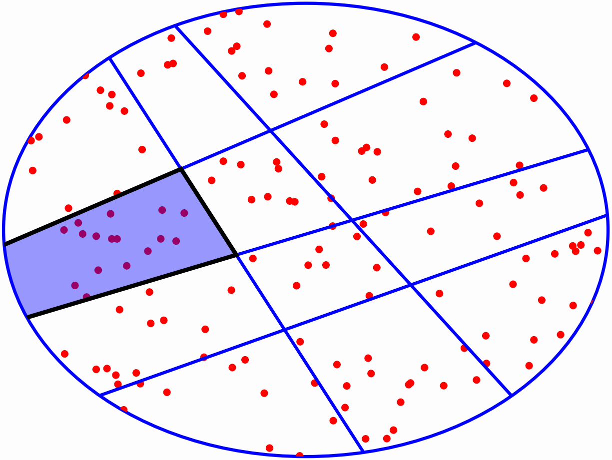
As Simple as Counting Dots





As Simple as Counting Dots

Pick a random cell



Estimate = Number of solutions in a cell $\times$ Number of cells

2-wise independent Hashing

- Let H be family of 2-wise independent hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$

$$\forall y_1, y_2 \in \{0,1\}^n, \alpha_1, \alpha_2 \in \{0,1\}^m, h \xleftarrow{R} H$$
$$\Pr[h(y_1) = \alpha_1] = \Pr[h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)$$

$$\Pr[h(y_1) = \alpha_1 \wedge h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)^2$$

2-wise independent Hashing

- Let H be family of 2-wise independent hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$

$$\forall y_1, y_2 \in \{0,1\}^n, \alpha_1, \alpha_2 \in \{0,1\}^m, h \xleftarrow{R} H$$
$$\Pr[h(y_1) = \alpha_1] = \Pr[h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)$$

$$\Pr[h(y_1) = \alpha_1 \wedge h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)^2$$

- The power of 2-wise independentity
  - Z be the number of solutions in a randomly chosen cell
  - $\mathsf{E}[Z] = \frac{|\mathsf{Sol}(F)|}{2^m}$
  - $\sigma^2[Z] \leq \mathsf{E}[Z]$

2-wise independent Hashing

- Let H be family of 2-wise independent hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$

$$\forall y_1, y_2 \in \{0,1\}^n, \alpha_1, \alpha_2 \in \{0,1\}^m, h \xleftarrow{R} H$$

$$\Pr[h(y_1) = \alpha_1] = \Pr[h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)$$

$$\Pr[h(y_1) = \alpha_1 \wedge h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)^2$$

- The power of 2-wise independentity
  - Z be the number of solutions in a randomly chosen cell
  - $\mathsf{E}[Z] = \frac{|\mathsf{Sol}(F)|}{2^m}$
  - $\sigma^2[Z] \leq \mathsf{E}[Z]$
- $\Pr\left[\frac{\mathsf{E}[Z]}{1+\varepsilon} \leq Z \leq \mathsf{E}[Z](1+\varepsilon)\right] \geq 1 - \frac{1}{(\frac{\varepsilon}{1+\varepsilon})^2(\mathsf{E}[Z])}$

2-wise independent Hashing

- Let H be family of 2-wise independent hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$

$$\forall y_1, y_2 \in \{0,1\}^n, \alpha_1, \alpha_2 \in \{0,1\}^m, h \xleftarrow{R} H$$

$$\Pr[h(y_1) = \alpha_1] = \Pr[h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)$$

$$\Pr[h(y_1) = \alpha_1 \wedge h(y_2) = \alpha_2] = \left(\frac{1}{2^m}\right)^2$$

- The power of 2-wise independentity
  - Z be the number of solutions in a randomly chosen cell
  - $\mathsf{E}[Z] = \frac{|\mathsf{Sol}(F)|}{2^m}$
  - $\sigma^2[Z] \leq \mathsf{E}[Z]$
- $\Pr\left[\frac{\mathsf{E}[Z]}{1+\varepsilon} \leq Z \leq \mathsf{E}[Z](1+\varepsilon)\right] \geq 1 - \frac{1}{(\frac{\varepsilon}{1+\varepsilon})^2(\mathsf{E}[Z])}$
- $\mathsf{E}[Z] = c(\frac{1+\varepsilon}{\varepsilon})^2$ provides $1 - \frac{1}{c}$ lower bound

2-wise independent Hash Functions

- Variables: $X_1, X_2, \cdots X_n$
- To construct $h : \{0, 1\}^n \to \{0, 1\}^m$, choose m random XORs
- Pick every X_i with prob. $\frac{1}{2}$ and XOR them
  - $X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2}$
  - Expected size of each XOR: $\frac{n}{2}$

2-wise independent Hash Functions

- Variables: $X_1, X_2, \cdots X_n$
- To construct $h : \{0,1\}^n \rightarrow \{0,1\}^m$, choose m random XORs
- Pick every X_i with prob. $\frac{1}{2}$ and XOR them
  - $X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2}$
  - Expected size of each XOR: $\frac{n}{2}$
- To choose $\alpha \in \{0,1\}^m$, set every XOR equation to 0 or 1 randomly

$$X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2} = 0 \quad (Q_1)$$
$$X_2 \oplus X_5 \oplus X_6 \cdots \oplus X_{n-1} = 1 \quad (Q_2)$$
$$\cdots \quad (\cdots)$$
$$X_1 \oplus X_2 \oplus X_5 \cdots \oplus X_{n-2} = 1 \quad (Q_m)$$

- Solutions in a cell: $F \wedge Q_1 \cdots \wedge Q_m$

- $(1+\varepsilon, \delta)$-Approximation

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq ApproxCount(F, \varepsilon, \delta) \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$

- $(1+\varepsilon, \delta)$-Approximation

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq \mathit{ApproxCount}(F,\varepsilon,\delta) \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$

- Constant Factor Approximation: $(4,\delta)$

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{4} \leq \mathit{ConstantCount}(F,\delta) \leq 4\cdot|\mathsf{Sol}(F)|\right] \geq 1-\delta$$

- $(1+\varepsilon, \delta)$-Approximation

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq ApproxCount(F,\varepsilon,\delta) \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$

- Constant Factor Approximation: $(4,\delta)$

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{4} \leq ConstantCount(F,\delta) \leq 4\cdot|\mathsf{Sol}(F)|\right] \geq 1-\delta$$

- From 4 to 2-factor

  Let $G = F(X_1) \wedge F(X_2)$ (i.e., two identical copies of F)

$$\frac{|\mathsf{Sol}(G)|}{4} \leq C \leq 4\cdot|\mathsf{Sol}(G)| \implies \frac{|\mathsf{Sol}(F)|}{2} \leq \sqrt{C} \leq 2\cdot|\mathsf{Sol}(F)|$$

“Stockmeyer’s Approach”

Constant Factor Suffices

- $(1+\varepsilon, \delta)$-Approximation

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq ApproxCount(F, \varepsilon, \delta) \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$

- Constant Factor Approximation: $(4,\delta)$

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{4} \leq ConstantCount(F, \delta) \leq 4 \cdot |\mathsf{Sol}(F)|\right] \geq 1-\delta$$

- From 4 to 2-factor
  Let $G = F(X_1) \wedge F(X_2)$ (i.e., two identical copies of F)

$$\frac{|\mathsf{Sol}(G)|}{4} \leq C \leq 4 \cdot |\mathsf{Sol}(G)| \implies \frac{|\mathsf{Sol}(F)|}{2} \leq \sqrt{C} \leq 2 \cdot |\mathsf{Sol}(F)|$$

- From 4 to $(1+\varepsilon)$-factor
  Construct $G = F(X_1) \wedge F(X_2) \cdots F(X_{\frac{1}{\varepsilon}})$ And then we can take $\frac{1}{\varepsilon}$-root

- aComp(F, k)
  - If $|\text{Sol}(F)| \geq 2^{k+1}$, then aComp($F, k$) returns YES whp
  - If $|\text{Sol}(F)| < 2^{k}$, then aComp(F, k) returns NO whp

"Stockmeyer's Approach" S83, JVV86, BP95

- aComp(F, k)
  - If $|\text{Sol}(F)| \geq 2^{k+1}$, then aComp($F, k$) returns YES whp
  - If $|\text{Sol}(F)| < 2^{k}$, then aComp(F, k) returns NO whp
- Counter(F)
  - Invoke aComp(F, k) for $k = 0, 1, \ldots n$
  - Use binary search find the first k s.t. aComp(F, k) return NO

"Stockmeyer's Approach"

S83, JVV86, BP95

- aCompp(F, k)
  - If $|\text{Sol}(F)| \geq 2^{k+1}$, then aComp$(F, k)$ returns YES whp
  - If $|\text{Sol}(F)| < 2^{k}$, then aComp(F, k) returns NO whp
- Counter(F)
  - Invoke aComp(F, k) for $k = 0, 1, \ldots n$
  - Use binary search find the first k s.t. aComp(F, k) return NO
- For $(1 + \varepsilon)$-approx, invoke Counter on $F(X_1) \wedge F(X_2) \cdots F(X_{\frac{1}{\varepsilon}})$ and return $\frac{1}{\varepsilon}$-root

"Stockmeyer's Approach"

S83, JVV86, BP95

- aComp(F, k)
  - If $|\text{Sol}(F)| \geq 2^{k+1}$, then aComp($F, k$) returns YES whp
  - If $|\text{Sol}(F)| < 2^{k}$, then aComp(F, k) returns NO whp
- Counter(F)
  - Invoke aComp(F, k) for $k = 0, 1, \ldots n$
  - Use binary search find the first k s.t. aComp(F, k) return NO
- For $(1+\varepsilon)$-approx, invoke Counter on $F(X_1) \wedge F(X_2) \cdots F(X_{\frac{1}{\varepsilon}})$ and return $\frac{1}{\varepsilon}$-root
- aComp(F, k)
  - Call $\mathcal{O}(\log\log n)$ calls to ThreshSAT($F \wedge Q_1 \wedge \ldots \wedge Q_{k-5}, 48$) and return the median.

"Stockmeyer's Approach" S83, JVV86, BP95

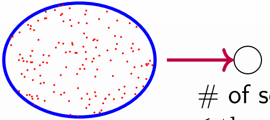
- aComp(F, k)
  - If $|\text{Sol}(F)| \geq 2^{k+1}$, then aComp$(F, k)$ returns YES whp
  - If $|\text{Sol}(F)| < 2^{k}$, then aComp(F, k) returns NO whp
- Counter(F)
  - Invoke aComp(F, k) for $k = 0, 1, \ldots n$
  - Use binary search find the first k s.t. aComp(F, k) return NO
- For $(1+\varepsilon)$-approx, invoke Counter on $F(X_1) \wedge F(X_2) \cdots F(X_{\frac{1}{\varepsilon}})$ and return $\frac{1}{\varepsilon}$-root Too large queries
- aComp(F, k)
  - Call $\mathcal{O}(\log \log n)$ calls to ThreshSAT$(F \wedge Q_1 \wedge \ldots \wedge Q_{k-5}, 48)$ and return the median. Too many calls due to Union Bounds

"Stockmeyer's Approach"

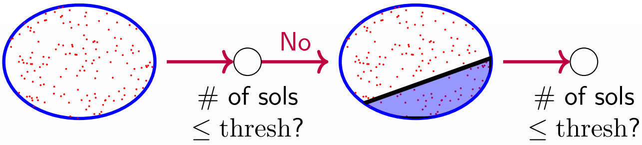
S83, JVV86, BP95

- aComp(F, k)
  - If $|\mathrm{Sol}(F)| \geq 2^{k+1}$, then aComp($F, k$) returns YES whp
  - If $|\mathrm{Sol}(F)| < 2^{k}$, then aComp($F, k$) returns NO whp
- Counter(F)
  - Invoke aComp(F, k) for $k = 0, 1, \ldots n$
  - Use binary search find the first k s.t. aComp(F, k) return NO
- For $(1 + \varepsilon)$-approx, invoke Counter on $F(X_1) \wedge F(X_2) \cdots F(X_{\frac{1}{\varepsilon}})$ and return $\frac{1}{\varepsilon}$-root   Too large queries   thresh $= \frac{9}{\varepsilon^2}$
- aComp(F, k)
  - Call $\mathcal{O}(\log \log n)$ calls to ThreshSAT($F \wedge Q_1 \wedge \ldots \wedge Q_{k-5}, 48$) and return the median.   Too many calls due to Union Bounds
  - Dependence to avoid union bounds

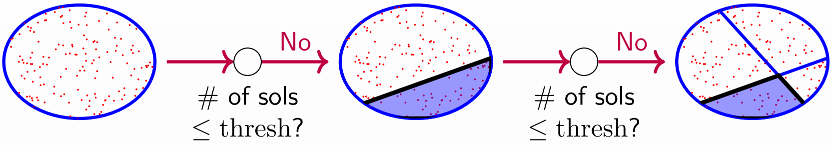
ApproxMC



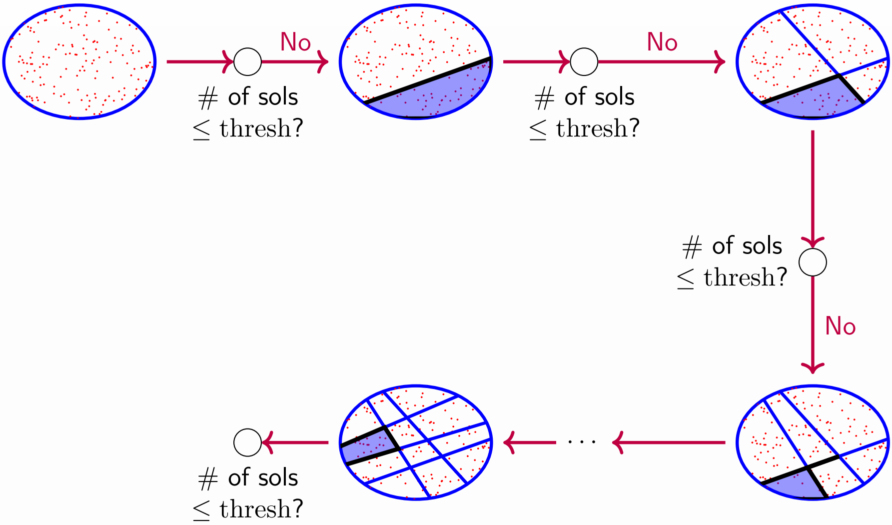
ApproxMC



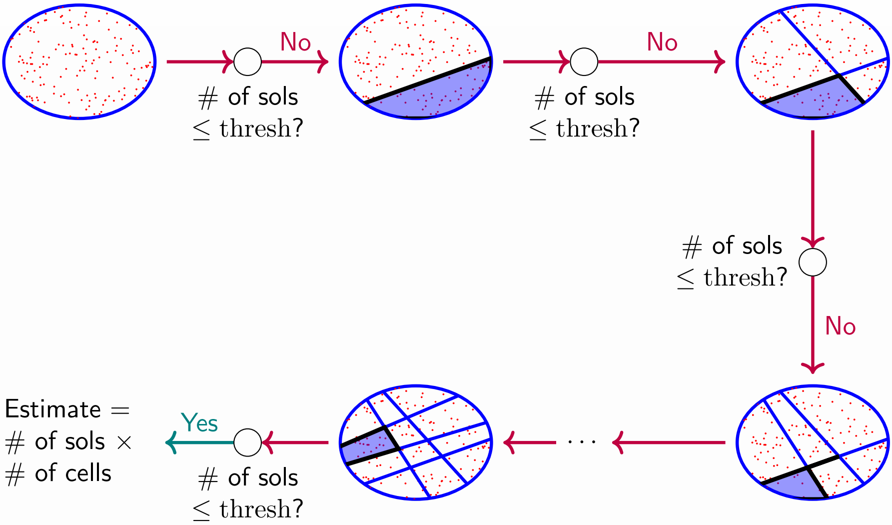
ApproxMC



ApproxMC

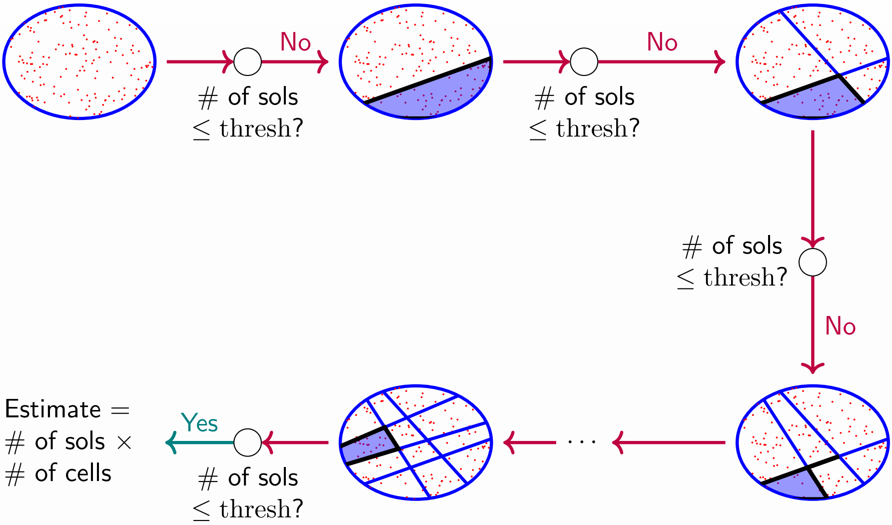

No
# of sols
≤ thresh?
No
# of sols
≤ thresh?
# of sols
≤ thresh?
No
· · ·
# of sols
≤ thresh?

ApproxMC



No
# of sols
≤ thresh?
No
# of sols
≤ thresh?
# of sols
≤ thresh?
No
...
Yes
# of sols
≤ thresh?
Estimate =
# of sols ×
# of cells

ApproxMC



Repeat $\mathcal{O}(\log(1/\delta))$ times and return the median

ApproxMC

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \mathrm{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \mathrm{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \mathrm{thresh}$
- Stop at the first m where Query m returns YES and return estimate as BoundedSAT$(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m, \mathrm{thresh}) \times 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES

ApproxMC

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \mathrm{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \mathrm{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \mathrm{thresh}$
- Stop at the first m where Query m returns YES and return estimate as BoundedSAT$(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m, \text{thresh}) \times 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES
  - Logarithmic search (# of SAT calls: $\mathcal{O}(\log n)$)
  - Incremental Search

ApproxMC

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \mathrm{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \mathrm{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \mathrm{thresh}$
- Stop at the first m where Query m returns YES and return estimate as BoundedSAT($F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m$, thresh) $\times\, 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES
  - Logarithmic search (# of SAT calls: $\mathcal{O}(\log n)$)
  - Incremental Search
- The Boon of Dependence
  - $E_i : \left|\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_i) - \frac{|\mathsf{Sol}(F)|}{2^i}\right| \geq (1+\varepsilon)\frac{|\mathsf{Sol}(F)|}{2^i}$

ApproxMC

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \mathrm{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \mathrm{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \mathrm{thresh}$
- Stop at the first m where Query m returns YES and return estimate as BoundedSAT$(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m, \text{thresh}) \times 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES
  - Logarithmic search (# of SAT calls: $\mathcal{O}(\log n)$)
  - Incremental Search
- The Boon of Dependence
  - $E_i : \left|\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_i) - \frac{|\mathsf{Sol}(F)|}{2^i}\right| \geq (1+\varepsilon)\frac{|\mathsf{Sol}(F)|}{2^i}$
  - (Loosely), $E_i \subseteq E_{i+1}$ for $i < m * -2$
  - (Loosely), $E_j \supseteq E_{j+1}$ for $j > m^* + 1$

ApproxMC

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\text{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \text{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \text{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \text{thresh}$
- Stop at the first m where Query m returns YES and return estimate as BoundedSAT$(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m, \text{thresh}) \times 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES
  - Logarithmic search (# of SAT calls: $\mathcal{O}(\log n)$)
  - Incremental Search
- The Boon of Dependence
  - $E_i : \left|\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_i) - \frac{|\mathsf{Sol}(F)|}{2^i}\right| \geq (1+\varepsilon)\frac{|\mathsf{Sol}(F)|}{2^i}$
  - (Loosely), $E_i \subseteq E_{i+1}$ for $i < m * -2$
  - (Loosely), $E_j \supseteq E_{j+1}$ for $j > m^* + 1$
  - $\Pr[\text{Error}] = \Pr[E_{m^*-2}] + \Pr[E_{m^*-1}] + \Pr[E_{m^*}] + \Pr[E_{m^*+1}]$

ApproxMC

Theorem (Correctness)

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq \mathit{ApproxMC}(F,\varepsilon,\delta) \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$

Theorem (Complexity)

ApproxMC(F,ε,δ) *makes* $\mathcal{O}(\frac{\log n \log(\frac{1}{\delta})}{\varepsilon^2})$ *calls to SAT oracle.*

ApproxMC

Theorem (Correctness)

$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq \mathit{ApproxMC}(F,\varepsilon,\delta) \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$

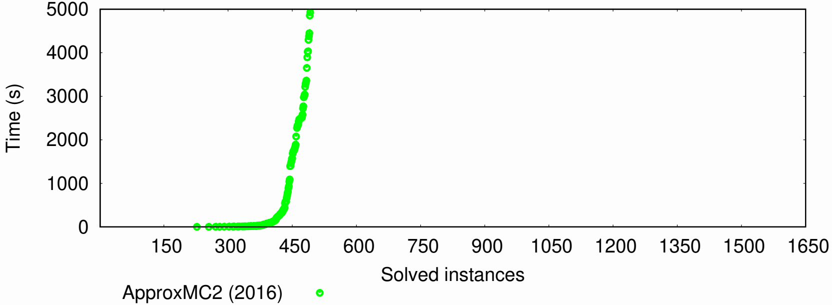
Theorem (Complexity)

ApproxMC(F,ε,δ) *makes* $\mathcal{O}(\frac{\log n \log(\frac{1}{\delta})}{\varepsilon^2})$ *calls to SAT oracle.*

Theorem (FPRAS for DNF; (MSV, FSTTCS 17; CP 18, IJCAI-19))

If F is a DNF formula, then ApproxMC is FPRAS – different from the Monte-Carlo based FPRAS for DNF (Karp, Luby 1983)

A Practical Counter



ApproxMC1 (without dependence) was 10-100$\times$ slower.

The Hope of Short XORs

- If we pick every variable X_i with probability p .
  - Expected Size of each XOR: np
  - $\mathsf{E}[Z_m] = \frac{|\mathsf{Sol}(F)|}{2^m}$
  - $\sigma^2[Z_m] \leq \mathsf{E}[Z_m] + \sum_{\sigma_1 \in \mathsf{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathsf{Sol}(F) \\ w = d(\sigma_1, \sigma_2)}} r(w, m)$
    - where, $r(w, m) = \left(\left(\frac{1}{2} + \frac{(1-2p)^w}{2} \right)^m - \frac{1}{2^m} \right)$
  - For $p = \frac{1}{2}$, we have $\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]} \leq 1$
- Earlier Attempts (GSS07,EGSS14,ZCSE16,AD17,ATD18)
  - $\sum_{\sigma_1 \in \mathsf{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathsf{Sol}(F) \\ w = d(\sigma_1, \sigma_2)}} r(w, m) \leq \sum_{\sigma_1 \in \mathsf{Sol}(F)} \sum_{w=0}^{n} \binom{n}{w} r(w, m)$
  - $\binom{n}{w}$ grows very fast with n, so could not upper bound $\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]}$
  - The weak bounds lead to significant slowdown: typically $100\times$ to $1000\times$ factor of slowdown! (ATD18,ABM20)

- $\sum_{\sigma_1 \in \mathrm{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathrm{Sol}(F) \\ w = d(\sigma_1, \sigma_2)}} r(w, m) = \sum_{w=0}^{n} C_F(w) r(w, m)$
- $C_F(w) = |\{\sigma_1, \sigma_2 \in \mathrm{Sol}(F) \mid d(\sigma_1, \sigma_2) = w\}|$

The Power of Isoperimetric Inequalities

- $\sum_{\sigma_1 \in \mathrm{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathrm{Sol}(F) \\ w = d(\sigma_1, \sigma_2)}} r(w, m) = \sum_{w=0}^{n} C_F(w) r(w, m)$
- $C_F(w) = |\{\sigma_1, \sigma_2 \in \mathrm{Sol}(F) \mid d(\sigma_1, \sigma_2) = w\}|$
- Isoperimetric Inequalities! (Rashtchian and Raynaud 2019)

Lemma

$$\sum_{w=0}^{n} C_F(w) r(w, m) \leq \sum_{w=0}^{n} \binom{8e\sqrt{n \cdot \ell}}{w} r(w, m) \text{ where } \ell = \log |\mathrm{Sol}(F)|$$

- $\frac{\binom{n}{w}}{\binom{8e\sqrt{n \cdot \ell}}{w}} \approx \left(\frac{n}{\ell}\right)^{\frac{w}{2}}$

From Linear to Logarithmic Size XORs

Theorem (Informal)

For all q, k, $|\mathrm{Sol}(F)| \leq k \cdot 2^m$, $p = \mathcal{O}(\frac{\log m}{m})$ *we have*

$$\frac{\sigma^2[Z_m]}{\mathrm{E}[Z_m]} \leq q (\text{a constant})$$

Recall, average size of XORs: $n \cdot p$
Improvement of p *from* $\frac{m/2}{m}$ *to* $\frac{\log m}{m}$

From Linear to Logarithmic Size XORs

Theorem (Informal)

For all q, k, $|\mathrm{Sol}(F)| \leq k \cdot 2^m$, $p = \mathcal{O}(\frac{\log m}{m})$ *we have*

$$\frac{\sigma^2[Z_m]}{\mathrm{E}[Z_m]} \leq q (a\ constant)$$

Recall, average size of XORs: $n \cdot p$
Improvement of p *from* $\frac{m/2}{m}$ *to* $\frac{\log m}{m}$

Challenge: No meaningful bounds on $|\mathrm{Sol}(F)|$

- $\Pr[\mathrm{Error}] = \Pr[E_{m^*-2}] + \Pr[E_{m^*-1}] + \Pr[E_{m^*}] + \Pr[E_{m^*+1}]$
- Key Insight: When adding m-th XOR, theoretical analysis only requires $\frac{\sigma^2[Z_m]}{\mathrm{E}[Z_m]} \leq q$ whenever $|\mathrm{Sol}(F)| \leq \mathrm{thresh} \cdot 2^m$

From Linear to Logarithmic Size XORs

Theorem (Informal)

For all q, k, $|\mathsf{Sol}(F)| \leq k \cdot 2^m$, $p = \mathcal{O}(\frac{\log m}{m})$ we have

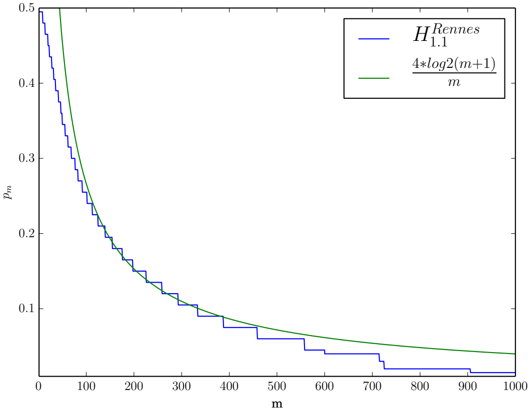
$$\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]} \leq q \text{(a constant)}$$

Recall, average size of XORs: $n \cdot p$
Improvement of p from $\frac{m/2}{m}$ to $\frac{\log m}{m}$

Challenge: No meaningful bounds on $|\mathsf{Sol}(F)|$

- $\Pr[\text{Error}] = \Pr[E_{m^*-2}] + \Pr[E_{m^*-1}] + \Pr[E_{m^*}] + \Pr[E_{m^*+1}]$
- Key Insight: When adding m-th XOR, theoretical analysis only requires $\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]} \leq q$ whenever $|\mathsf{Sol}(F)| \leq \text{thresh} \cdot 2^m$
- Add m-th XOR with $p_m = \mathcal{O}(\frac{\log m}{m})$

Sparse Hash Functions



$H_{1.1}^{Rennes}$: Sparse hash functions that guarantee $q = 1.1$

Sparse XORs

Benchmark	Vars	$\log_2$(Count)	ApproxMC	ApproxMC+Sparse	Speedup
03B-4	27966	28.55	983.72	1548.96	0.64
squaring23	710	23.11	0.66	1.21	0.55
case144	765	82.07	102.65	202.06	0.51
modexp8-4-6	83953	32.13	788.23	920.34	0.86
min-28s	3933	459.23	48.63	35.83	1.36
s9234a_7_4	6313	246.0	4.77	2.45	1.95
min-8	1545	284.78	8.86	4.59	1.93
s13207a_7_4	9386	699.0	34.94	17.05	2.05
min-16	3065	539.88	33.67	16.61	2.03
90-15-4-q	1065	839.25	273.1	135.75	2.01
s35932_15_7	17918	1761.0	–	72.32	–
s38417_3_2	25528	1663.02	–	71.04	–
75-10-8-q	460	360.13	–	4850.28	–
90-15-8-q	1065	840.0	–	3717.05	–

Remember; thresh $= \mathcal{O}(\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]} \cdot \frac{1}{\varepsilon^2})$

$\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]} \leq 1$ for 2-wise independent; $\frac{\sigma^2[Z_m]}{\mathsf{E}[Z_m]} \leq q = 1.1$ for $H_{1.1}^{Rennes}$.

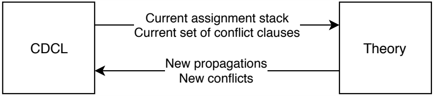
Outline

- Algorithmic
  - From Stockmeyer to ApproxMC
  - The Boon of Dependence
  - Sparse XORs
- System: Efficient CNF+XOR Solving (Soos's possible talk in SAT Seminar?)
- Conceptual
  - Independent Support
  - Projection

CDCL(T)

For theories that are not efficiently simulated by CDCL

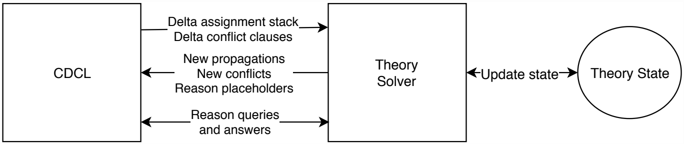
- T is the theory, e.g.:
  - Gauss-Jordan Elimination [SoosNohlCastelluccia'2010]
  - Pseudo-Boolean Reasoning [ChaiKuehlmann'2006]
  - Symmetric Explanation Learning [DevriendtBogaertsBruynooghe'2017]
- Theory is run side-by-side to the CDCL algorithm
- **Propagate** values implied by Theory given current assignment stack of CDCL
- **Conflict** if Theory implies 1=0 given current assignment stack of CDCL
- Theory must give reason for propagations&conflicts



CDCL(T) Cont.

Optimizations:

- Should only send delta of assignment stack + conflict clauses
  - Variables assigned (decisions + propagations)
  - Variables unassigned (backtracking, restarting)
  - New conflict clauses
- Theory only needs to compute delta relative to old state
- Theory can give placeholders for reasons
  - If reason is needed during conflict generation, Theory is queried
  - Called "lazy" (vs "greedy") interpolant generation



CDCL(T) Gauss-Jordan Elimination: Ingredients

What components do we need?

- **Extractor for XOR constraints**: XORs may be encoded as CNF
- **Delta update mechanism** for row-echelon form matrix:
  - how to handle when variable is set
  - how to handle when variable is unset
- **Efficient data structures** to allow for quick updates
- **Reason generation**

CDCL(T) Gauss-Jordan Elimination: Extraction

$$l_1 \oplus l_2 \oplus l_3 = 1 \quad \Leftrightarrow \quad \begin{array}{l} l_1 \vee l_2 \vee l_3 \wedge \\ \bar{l}_1 \vee \bar{l}_2 \vee l_3 \wedge \\ \bar{l}_1 \vee l_2 \vee \bar{l}_3 \wedge \\ l_1 \vee \bar{l}_2 \vee \bar{l}_3 \wedge \end{array}$$

$$l_1 \oplus l_2 \oplus l_3 = 1 \quad \leftarrow \quad \begin{array}{l} l_1 \vee l_2 \vee \wedge \\ \bar{l}_1 \vee \bar{l}_2 \vee l_3 \wedge \\ \bar{l}_1 \vee l_2 \vee \bar{l}_3 \wedge \\ l_1 \vee \bar{l}_2 \vee \bar{l}_3 \wedge \end{array}$$

- Missing literals only mean something stronger than XOR
- XOR is still implied and should be detected

CDCL(T) Gauss-Jordan Elimination: 2-variable watchlist

Let's use a 2-variable watch scheme [HanJiang2012]:

- If 2 or more variables are unset in XOR constraint, it cannot propagate or conflict
- If 1 variable is unset, it must propagate
- If 0 variable is unset, it is either satisfied or is in conflict

Watching for propagation and to perform GJE

- For every row (of XOR), there is a *pivot* variable (among the two variables watching the row)
- A variable is pivot for at most one row.

CDCL(T) GJE: Reason Clauses and Backtracking

What combination of XOR constraints gave us the propagation?

- Each row is a combination of input XOR constraints
- It is guaranteed to propagate/conflict under current variable assignment

During backtracking:

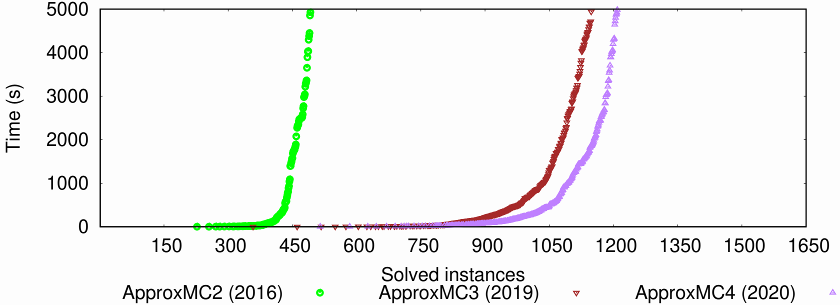
- All previous invariants still hold
- If the column (variable) was pivot for a row, it still is
- Both watches of the row are still good and in the watchlists
- Matrix looks differently than when we last had this assignment... is that a problem?
- No! Observe: new matrix could have been reached from the starting position, pivoting differently(!)

CDCL(T) Gauss-Jordan Elimination: Recap

Let's recap! What was hard:

- Extracting XOR constraints
- Keeping CDCL and GJ in sync:
  - Fast update for variable setting (propagation)
  - Fast update for backtracking (conflict)
- Reason clause generation

Improvements Over the Years



Outline

- Algorithmic
  - From Stockmeyer to ApproxMC
  - The Boon of Dependence
  - Sparse XORs
- System: Efficient CNF+XOR Solving (Soos's possible talk in SAT Seminar?)
- Conceptual
  - Independent Support
  - Projection

Improved 2-wise Independent Hash Functions

- Not all variables are required to specify solution space of F
  - $F := X_3 \iff (X_1 \vee X_2)$
  - X_1 and X_2 uniquely determines rest of the variables (i.e., X_3)
- Formally: if I is independent support, then $\forall \sigma_1, \sigma_2 \in \mathsf{Sol}(F)$, if σ_1 and σ_2 agree on I then $\sigma_1 = \sigma_2$
  - $\{X_1, X_2\}$ is independent support but $\{X_1, X_3\}$ is not

Improved 2-wise Independent Hash Functions

- Not all variables are required to specify solution space of F
  - $F := X_3 \iff (X_1 \vee X_2)$
  - X_1 and X_2 uniquely determines rest of the variables (i.e., X_3)
- Formally: if I is independent support, then $\forall \sigma_1, \sigma_2 \in \mathsf{Sol}(F)$, if σ_1 and σ_2 agree on I then $\sigma_1 = \sigma_2$
  - $\{X_1, X_2\}$ is independent support but $\{X_1, X_3\}$ is not
- Random XORs need to be constructed only over I (CMV DAC14)

Improved 2-wise Independent Hash Functions

- Not all variables are required to specify solution space of F
  - $F := X_3 \iff (X_1 \vee X_2)$
  - X_1 and X_2 uniquely determines rest of the variables (i.e., X_3)
- Formally: if I is independent support, then $\forall \sigma_1, \sigma_2 \in \mathsf{Sol}(F)$, if σ_1 and σ_2 agree on I then $\sigma_1 = \sigma_2$
  - $\{X_1, X_2\}$ is independent support but $\{X_1, X_3\}$ is not
- Random XORs need to be constructed only over I (CMV DAC14)
- Typically I is 1-2 orders of magnitude smaller than X
- Auxiliary variables introduced during encoding phase are *dependent* (Tseitin 1968)

Algorithmic procedure to determine I?

Determining Independent Support

Independent Support: I Defined Variables: $X \setminus I$

- If I is independent support and x_n is defined in terms of $I \setminus \{x_n\}$, then $I \setminus \{x_n\}$ is independent support.

Determining Independent Support

Independent Support: I Defined Variables: $X \setminus I$

- If I is independent support and x_n is defined in terms of $I \setminus \{x_n\}$, then $I \setminus \{x_n\}$ is independent support.
- Padao's Theorem [1901] x_n is defined in terms of I if and only if

$$F(X) \wedge F(Y) \wedge \bigwedge_{x_i \in I} (x_i = y_i) \implies (x_n = y_n) \text{ is VALID}$$

$$\text{i.e., } F(X) \wedge F(Y) \wedge \bigwedge_{x_i \in I} (x_i = y_i) \wedge x_n \wedge \neg y_n \text{ is UNSAT}$$

- So iterative procedure with initial $I = X$ and remove x_i from I if x_i is defined in terms of $I \setminus \{x_i\}$
- $\mathcal{O}(n)$ SAT calls

Outline

- Algorithmic
  - From Stockmeyer to ApproxMC
  - The Boon of Dependence
  - Sparse XORs
- System: Efficient CNF+XOR Solving (Soos's possible talk in SAT Seminar?)
- Conceptual
  - Independent Support
  - Projection

Projected Counting

- Given F on $X \cup Y$, count number of solutions of $\exists Y F(X, Y)$
- Let $X = x_1$; $Y = y_1$; $F = (x_1 \vee y_1)$.
- So $\mathrm{Sol}(\exists Y F(X, Y)) = \{(x_1 = 0), (x_1 = 1)\}$
- Therefore, $|\mathrm{Sol}(\exists Y F(X, Y))| = 2$
- How do we compute $|\mathrm{Sol}(\exists Y F(X, Y))|$?

Projected Counting

- Given F on $X \cup Y$, count number of solutions of $\exists Y F(X, Y)$
- Let $X = x_1$; $Y = y_1$; $F = (x_1 \vee y_1)$.
- So $\mathsf{Sol}(\exists Y F(X, Y)) = \{(x_1 = 0), (x_1 = 1)\}$
- Therefore, $|\mathsf{Sol}(\exists Y F(X, Y))| = 2$
- How do we compute $|\mathsf{Sol}(\exists Y F(X, Y))|$?
- Approach 1: Perform quantifier elimination

Projected Counting

- Given F on $X \cup Y$, count number of solutions of $\exists Y F(X, Y)$
- Let $X = x_1$; $Y = y_1$; $F = (x_1 \vee y_1)$.
- So $\mathrm{Sol}(\exists Y F(X, Y)) = \{(x_1 = 0), (x_1 = 1)\}$
- Therefore, $|\mathrm{Sol}(\exists Y F(X, Y))| = 2$
- How do we compute $|\mathrm{Sol}(\exists Y F(X, Y))|$?
- Approach 1: Perform quantifier elimination
- Approach 2: ApproxMC with minor changes
  - XORs over X and also enumerate solutions over X.
  - ProjThresh(F, X, thresh):
    #Queries: thresh Size: $|F| + \text{thresh} * |X|$ for SAT Oracle
- Usage of $\exists$ can lead to exponentially succinct formulas

Reliability of Critical Infrastructure Networks

- $G = (N, E)$; source node: s and terminal node t
- (wlog) every edge fails with prob $\frac{1}{2}$
- Compute Pr[s and t are disconnected]?

Reliability of Critical Infrastructure Networks

- $G = (N, E)$; source node: s and terminal node t
- (wlog) every edge fails with prob $\frac{1}{2}$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a $0/1$ vector of size $|E|$

Reliability of Critical Infrastructure Networks

- $G = (N, E)$; source node: s and terminal node t
- (wlog) every edge fails with prob $\frac{1}{2}$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a $0/1$ vector of size $|E|$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables

Reliability of Critical Infrastructure Networks

- $G = (N, E)$; source node: s and terminal node t
- (wlog) every edge fails with prob $\frac{1}{2}$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a $0/1$ vector of size $|E|$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables
- Pr[s and t are disconnected] $= \sum_{\pi_{s,t}} 2^{-E}$
- Variables for Nodes: $P_N = \{p_u\}_{u \in N}$ and Edges: $\{p_e\}_{e \in E}$
- Consider $e = (u, v)$: $p_u \wedge e_{u,v} \rightarrow p_v$
- $\varphi = p_s \wedge \neg p_t \wedge \bigwedge_{(u,v) \in E}(p_u \wedge e_{u,v} \rightarrow p_v))$

Reliability of Critical Infrastructure Networks

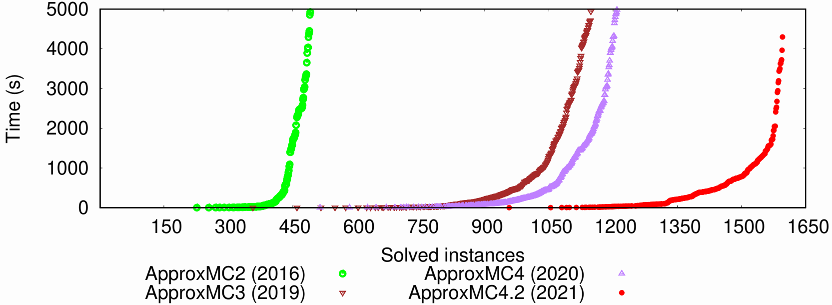
- $G = (N, E)$; source node: s and terminal node t
- (wlog) every edge fails with prob $\frac{1}{2}$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a $0/1$ vector of size $|E|$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables
- Pr[s and t are disconnected] $= \sum_{\pi_{s,t}} 2^{-E}$
- Variables for Nodes: $P_N = \{p_u\}_{u \in N}$ and Edges: $\{p_e\}_{e \in E}$
- Consider $e = (u, v)$: $p_u \wedge e_{u,v} \rightarrow p_v$
- $\varphi = p_s \wedge \neg p_t \wedge \bigwedge_{(u,v) \in E} (p_u \wedge e_{u,v} \rightarrow p_v))$
- Count $\exists P_N(\varphi)$: Projected Counting

So What Makes Hashing-based Techniques Work?

- Algorithmic
  - From Stockmeyer to ApproxMC
  - The Boon of Dependence
  - Sparse XORs
- System: Efficient CNF+XOR Solving (Soos' possible talk in SAT Seminar?)
- Conceptual
  - Independent Support
  - Projection

The Rise of Hashing-based Approach: Promise of Scalability and Guarantees
(S83,GSS06,GHSS07,CMV13b,EGSS13b,CMV14,CDR15,CMV16,ZCSE16,AD16, KM18,ATD18,SM19,ABM20,SGM20)

Improvements Over the Years



Reliability of Critical Infrastructure Networks

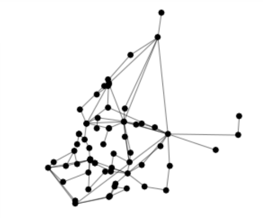
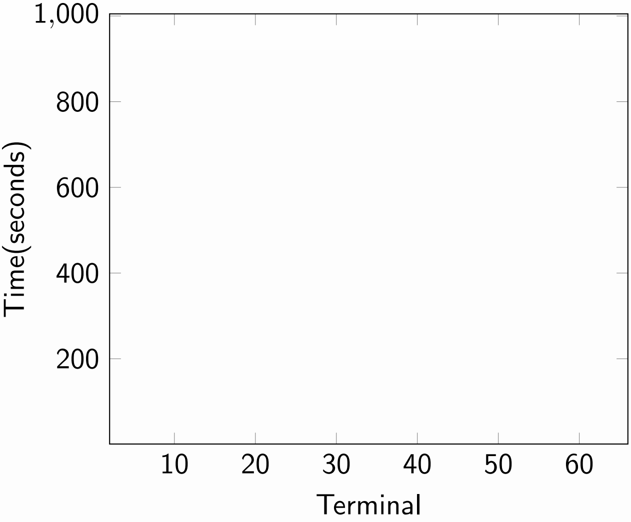


Figure: Plantersville, SC

- $G = (V, E)$; source node: s
- Compute Pr[t is disconnected]?

Timeout = 1000 seconds



(DMPV, AAAI17)

Reliability of Critical Infrastructure Networks

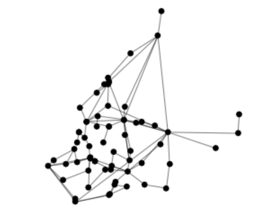
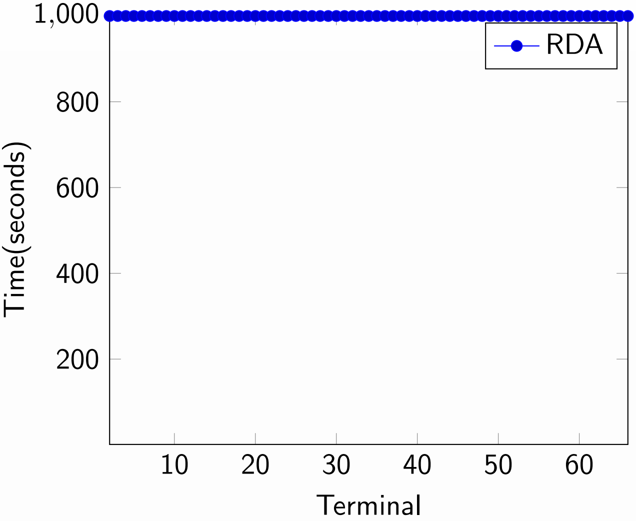


Figure: Plantersville, SC

- $G = (V, E)$; source node: s
- Compute Pr[t is disconnected]?

Timeout = 1000 seconds



(DMPV, AAAI17)

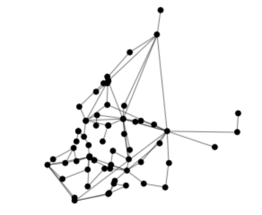
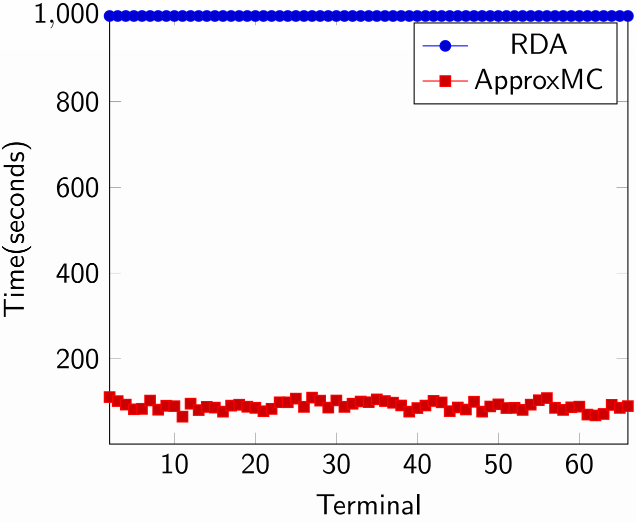


Figure: Plantersville, SC

- $G = (V, E)$; source node: s
- Compute Pr[t is disconnected]?

Timeout = 1000 seconds



(DMPV, AAAI17)

Where do we go from here?

Algorithmic	• Design of instance-dependent sparse XORs • Can we prove accuracy observed in practice?
System	• Better system for Sparse XORs • Hybrid Counter to exploit complimentary exact and approximate counting
Coneptual	• Independent support is model counting preserving but approximation would suffice • Proof of correctness