Counting, Sampling, and Synthesis: The Quest for Scalability

Kuldeep S. Meel

School of Computing

National University of Singapore
Counting, Sampling, and Synthesis: The Quest for Scalability

Computing: The Story of an Endless Quest for Scalability

Watson, 1940s: “I think there is a world market for about five computers.”

Gates & Allen, 1970s: “A computer on every desk and in every home”

2020: 22 billion IoT connected devices
Automated Reasoning

```c
PC1 (char[] S, char[] UI) {
    for (int i = 0; i < UI.length(); i++) {
        if (S[i] != UI[i])
            return No;
    }
    return Yes;
}
```

\[ \vdash \text{satisfies} \]

Central Question Is it always the case that \( M \| = P \)?

Equivalently, can it be the case that \( M \land \neg P \)?

Boolean Satisfiability (SAT): Given a Boolean formula, is there a solution, i.e., an assignment of 0's and 1's to the variables that makes the formula equal 1?

Example: \((X_1 \lor \neg X_2 \lor \neg X_3) \land (X_2 \lor \neg X_3)\)

\(X_1 = 1, X_2 = 1, X_3 = 1\)

Cook, 1971; Levin, 1973: SAT is NP-complete (= "intractable")

Knuth, 2016: These so-called "SAT solvers" can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

[Circa 2012]: Now that SAT is "easy", it is time to look beyond satisfiability...
Automated Reasoning

```c
int PC1(char[] SP, char[] UI) {
    for (int i = 0; i < UI.length(); i++) {
        if (SP[i] != UI[i])
            return No;
    }
    return Yes;
}
```

satisfies

\[ M(I, O) \models P(I, O) \]

Central Question Is it always the case that \( M \models P \)?
Equivalently, can it be the case that \( M \land \neg P \)?

Boolean Satisfiability (SAT): Given a Boolean formula, is there a solution, i.e., an assignment of 0's and 1's to the variables that makes the formula equal 1?

Example: \((X_1 \lor \neg X_2 \lor \neg X_3) \land (X_2 \lor \neg X_3)\)

\(X_1 = 1, X_2 = 1, X_3 = 1\)

Cook, 1971; Levin, 1973: SAT is NP-complete (= "intractable")

Knuth, 2016: These so-called "SAT solvers" can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

[Circa 2012]: Now that SAT is "easy", it is time to look beyond satisfiability.
Automated Reasoning

```
Automated Reasoning

PC1 (char[] SP, char[] UI) {
    for (int i = 0; i < UI.length(); i++) {
        if (SP[i] != UI[i]) return No;
    }
    return Yes;
}

|= satisfies

\[
M(I, O) \models P(I, O)
\]

Central Question Is it always the case that \( M \models P \)?
Equivalently, can it be the case that \( M \land \neg P \)?

Cook, 1971; Levin, 1973: SAT is NP-complete (= "intractable")
Knuth, 2016: These so-called "SAT solvers" can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

[Circa 2012]: Now that SAT is "easy", it is time to look beyond satisfiability.
Automated Reasoning

```java
PC1 (char[] SP, char[] UI) {
    for (int i = 0; i < UI.length(); i++) {
        if (SP[i] != UI[i]) return No;
    }
    return Yes;
}
```

satisfies

\[ \mathcal{M}(I, O) \models \mathcal{P}(I, O) \]

Central Question Is it always the case that \( \mathcal{M} \models \mathcal{P} \)?

Equivalently, can it be the case that \( \mathcal{M} \wedge \neg \mathcal{P} \)?

Boolean Satisfiability (SAT): Given a Boolean formula, is there a solution, i.e., an assignment of 0’s and 1’s to the variables that makes the formula equal 1?

Example: \( (X_1 \lor \neg X_2 \lor \neg X_3) \land (X_2 \lor \neg X_3) \)

\[ X_1 = 1, X_2 = 1, X_3 = 1 \]
Automated Reasoning

\[ \text{satisfies} \]

\[ \mathcal{M}(I, O) \] \(\models\) \[ \mathcal{P}(I, O) \]

Central Question Is it always the case that \(\mathcal{M} \models \mathcal{P}\)? Equivalently, can it be the case that \(\mathcal{M} \land \neg \mathcal{P}\)?

Boolean Satisfiability (SAT): Given a Boolean formula, is there a solution, i.e., an assignment of 0’s and 1’s to the variables that makes the formula equal 1?

Example: \((X_1 \lor \neg X_2 \lor \neg X_3) \land (X_2 \lor \neg X_3)\) \(X_1 = 1, X_2 = 1, X_3 = 1\)

Cook, 1971; Levin, 1973: SAT is \textbf{NP-complete} (= “intractable”)

Knuth, 2016: These so-called “SAT solvers” can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

[Circa 2012]: Now that SAT is “easy”, it is time to look beyond satisfiability.
Automated Reasoning

Central Question Is it always the case that $\mathcal{M} \models \mathcal{P}$?
Equivalently, can it be the case that $\mathcal{M} \land \neg \mathcal{P}$?

Boolean Satisfiability (SAT): Given a Boolean formula, is there a solution, i.e., an assignment of 0’s and 1’s to the variables that makes the formula equal 1?

Example: $(X_1 \lor \neg X_2 \lor \neg X_3) \land (X_2 \lor \neg X_3) \quad X_1 = 1, X_2 = 1, X_3 = 1$

Cook, 1971; Levin, 1973: SAT is NP-complete (= “intractable”)

Knuth, 2016: These so-called “SAT solvers” can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.
Automated Reasoning

Central Question Is it always the case that $M \models P$?

Equivalently, can it be the case that $M \land \neg P$?

Boolean Satisfiability (SAT): Given a Boolean formula, is there a solution, i.e., an assignment of 0’s and 1’s to the variables that makes the formula equal 1?

Example: $(X_1 \lor \neg X_2 \lor \neg X_3) \land (X_2 \lor \neg X_3)$  $X_1 = 1, X_2 = 1, X_3 = 1$

Cook, 1971; Levin, 1973: SAT is **NP-complete** (= “intractable”)

Knuth, 2016: These so-called “SAT solvers” can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

[Circa 2012]: Now that SAT is “easy”, it is time to look beyond satisfiability.
Beyond SAT I: Quantification

```c
PC2 (char[] SP, char[] UI) {
    match = true;
    for (int i=0; i<UI.length(); i++) {
        if (SP[i] != UI[i]) match = false;
        else match = match;
    }  
    if (match) return Yes;
    else return No;
}
```

Information Leakage

Fairness

Robustness

Critical Infrastructure
Beyond SAT I: Quantification

```c
PC2 (char[] SP, char[] UI) {
    match = true;
    for (int i=0; i<UI.length(); i++) {
        if (SP[i] != UI[i]) match=false;
        else match = match;
    }
    if (match) return Yes;
    else return No;
}
```

**Information Leakage**

**Fairness**

**Robustness**

**Critical Infrastructure**

**Quantification:** How often does $M$ satisfy $P$?

**Counting**
Beyond SAT II: Sampling

- System is simulated with test vectors
- Constraints represent *relevant* verification scenarios
- **Test vectors**: random solutions of constraints
Beyond SAT III: Automated Synthesis

\[
\begin{align*}
  f(u, v) & \geq u; \\
  f(u, v) & \geq v; \\
  f(u, v) & = u \lor \\
  f(u, v) & = v
\end{align*}
\]

Specification: \( \mathcal{P}(\mathcal{I}, \mathcal{O}) \)

<table>
<thead>
<tr>
<th>age</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>capital-gain</td>
<td>4000</td>
</tr>
<tr>
<td>occupation</td>
<td>coach</td>
</tr>
</tbody>
</table>

Synthesis
\[ f(u, v) \geq u; \]
\[ f(u, v) \geq v; \]
\[ f(u, v) = u \lor f(u, v) = v \]

Specification: \( \mathcal{P}(\mathcal{I}, \mathcal{O}) \)

Inputs \( \mathcal{I} \)

Outputs \( \mathcal{O} \)

Synthesis
Inputs $I$

\[
\begin{align*}
    f(u, v) &\geq u; \\
    f(u, v) &\geq v; \\
    f(u, v) &= u \lor \\
    f(u, v) &= v
\end{align*}
\]

Specification: $P(I, O)$

Outputs $O$

$$
\begin{aligned}
    \text{int } i &= 0 \\
    \text{while} (i < n) \\
    \{ \\
    \quad \text{if } (x_i < x_{i+1}) \\
    \quad \quad y_i = x_i \\
    \quad \quad \text{else} \\
    \quad \quad y_i = x_{i+1} \\
    \quad i = i+1
\}
\end{aligned}
$$
Research Overview

Synthesis

Interpretable Learning
Program Synthesis

Sampling

Hardware Validation

Ride Sharing

Neural Network Verification

Information Leakage

Configuration Checking

Counting

Infrastructure Resilience

Synthesis

Circuit Synthesis

Neural Network Verification

[BEHLMQ18,GJM22]

[BSSMS19,NSIMM19]
[SNIMMY22]

[SGRM18,GSRM19]

[BLM20,GSCM21]

[YLM23]

[MM18,GM19,GMM20]

[GRM20,GSRM21]

[GRM21]

[DMPV17,PDMV19]
Artificial Intelligence  
AAAI:17×, IJCAI:9×, NeurIPS: 6×, SAT:5×, CP:8×, KR:1×

Formal Methods  
CAV:6×, TACAS: 3×, ICCAD: 2×, DATE:2×, DAC: 1×

Logic/Databases  
LICS:2×, LPAR:2×, PODS:3×

Software Engineering  
ICSE:2×, FSE: 2×, CCS:1×

Today’s Talk: Counting
Counting

- **Given**: A Boolean formula $F$ over $X_1, X_2, \cdots X_n$
- $\text{Sol}(F) = \{ \text{solutions of } F \}$
- **SAT**: Determine if $\text{Sol}(F)$ is non-empty
- **Counting**: Determine $|\text{Sol}(F)|$
Counting

- **Given**: A Boolean formula $F$ over $X_1, X_2, \cdots X_n$
- $\text{Sol}(F) = \{ \text{solutions of } F \}$
- **SAT**: Determine if $\text{Sol}(F)$ is non-empty
- **Counting**: Determine $|\text{Sol}(F)|$
- **Example**: $F := (X_1 \lor X_2)$
  - $\text{Sol}(F) = \{(0, 1), (1, 0), (1, 1)\}$
  - $|\text{Sol}(F)| = 3$
Counting

- **Given**: A Boolean formula $F$ over $X_1, X_2, \cdots X_n$
- **Sol($F$)** = \{ solutions of $F$ \}
- **SAT**: Determine if Sol($F$) is non-empty
- **Counting**: Determine $|\text{Sol}(F)|$
- **Example**: $F := (X_1 \lor X_2)$
  - $\text{Sol}(F) = \{(0, 1), (1, 0), (1, 1)\}$
  - $|\text{Sol}(F)| = 3$
- **Generalization to arbitrary weights**
  - **Given** weight function (implicitly represented) $W$: $\{0, 1\}^n \rightarrow [0, 1]$
  - $W(F) = \Sigma_{y \in \text{Sol}(F)} W(y)$
  - **(Weighted) Counting**: Determine $W(F)$

Today's talk: We focus on unweighted case, i.e., $|\text{Sol}(F)|$
Today’s Menu

**Appetizer**  Applications

- Critical Infrastructure Resilience
- Quantitative Analysis of AI Systems

**Main Course**  ApproxMC: A Scalable Counting Framework

**Dessert**  Future Outlook
Can we predict the likelihood of a blackout due to natural disaster?
Can we predict the likelihood of a blackout due to natural disaster?

- $G = (V, E)$; set of source nodes $S$ and terminal node $t$
- failure probability $g : E \rightarrow [0, 1]$
- Compute $\Pr[ t \text{ is disconnected from } S]$?
Can we predict the likelihood of a blackout due to natural disaster?

- $G = (V, E)$; set of source nodes $S$ and terminal node $t$
- failure probability $g : E \rightarrow [0, 1]$
- Compute $\Pr[\text{t is disconnected from } S]$?

**Key Idea:** Encode disconnectedness using constraints

**Impact:** The first theoretically sound estimates of resilience in power transmission networks of ten medium sized cities in US
Quantitative Analysis of AI Systems
Our Focus: Binarized Neural Networks

Robustness Quantification

\[ \left| \{ x : \mathcal{N}(x + \epsilon) \neq \mathcal{N}(x) \} \right| \]
Robustness Quantification

\[ \{ x : \mathcal{N}(x + \varepsilon) \neq \mathcal{N}(x) \} \]

Encode Symbolically

Constrained Counting
Quantitative Analysis of AI Systems

Our Focus: Binarized Neural Networks

Robustness Quantification

$$\{x : \mathcal{N}(x + \varepsilon) \neq \mathcal{N}(x)\}$$

Encode Symbolically

Fairness Quantification

$$\{x : \mathcal{N}(x \land \text{Black}) \neq \mathcal{N}(x \land \text{White})\}$$

Encode Symbolically

Constrained Counting
Quantitative Analysis of AI Systems

Our Focus: Binarized Neural Networks

Robustness Quantification

\[
\{ x : \mathcal{N}(x + \varepsilon) \neq \mathcal{N}(x) \} \]

Encode Symbolically

Fairness Quantification

\[
\{ x : \mathcal{N}(x \land \text{BLACK}) \neq \mathcal{N}(x \land \text{WHITE}) \} \]

Encode Symbolically

Constrained Counting

Impact: The first scalable technique for rigorous quantification of robustness and fairness of Binarized Neural Networks
Applications across Computer Science

Impact: Counting-based approach is now the state of the art for all these applications
Valiant, 1979: Counting exactly is $\#P$-hard
Valiant, 1979: Counting exactly is \textbf{\#P-hard}

Stockmeyer, 1983: Probably Approximately Correct (PAC) aka $(\epsilon, \delta)$-guarantees

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

Stoc83, JVV86, BP94: Polynomial calls to SAT oracle suffice
Valiant, 1979: Counting exactly is \#P-hard

Stockmeyer, 1983: Probably Approximately Correct (PAC) aka \((\varepsilon, \delta)\)-guarantees

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

Stoc83, JVV86, BP94: Polynomial calls to SAT oracle suffice
- Not practical

SAT Solver \(\neq\) SAT Oracle

Performance of state of the art SAT solvers depends on the formulas
Snapshot from 2012

State of the art tool in 2012 could handle one out of 10^76 robustness instances. Can we bridge the gap between theory and practice?
Snapshot from 2012

State of the art tool in 2012 could handle one out of 10^76 robustness instances.

Can we bridge the gap between theory and practice?
State of the art tool in 2012 could handle one out of 1076 robustness instances

Can we bridge the gap between theory and practice?
State of the art tool in 2012 could handle one out of 1076 robustness instances

Can we bridge the gap between theory and practice?
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \( n = \) 23 bit identifier \( 2^n \approx 6.1M \)
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \((n =)\) 23 bit identifier \((2^n \approx 6.1M)\)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \((n =)\) 23 bit identifier \((2^n \approx 6.1M)\)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \( n \) 23 bit identifier \( 2^n \approx 6.1M \)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

- SAT Query: Find a person who likes coffee

Slide 14/37
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \( n = 23 \) bit identifier \( 2^n \approx 6.1M \)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

- SAT Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person y
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \((n =)\) 23 bit identifier \((2^n \approx 6.1M)\)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

- SAT Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person \(y\)

- Attempt #2: Enumerate every person who likes coffee
Counting in Atlanta

How many people in Atlanta like coffee?

- Population of Atlanta = 6.1M
- Assign every person a unique \( n \) = 23 bit identifier \( 2^n \approx 6.1M \)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

- SAT Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person \( y \)

- Attempt #2: Enumerate every person who likes coffee
  - Potentially \( 2^n \) queries

Can we do with lesser # of SAT queries – \( O(n) \) or \( O(\log n) \)?
As Simple as Counting Dots

Pick a random cell

Estimate = \text{Number of solutions in a cell} \times \text{Number of cells}
As Simple as Counting Dots

Pick a random cell

Estimate = Number of solutions in a cell × Number of cells
As Simple as Counting Dots

Pick a random cell

Estimate = Number of solutions in a cell \times Number of cells
**Challenges**

**Challenge 1** How to partition into *roughly equal small* cells of solutions without knowing the distribution of solutions?

**Challenge 2** How many cells?
Challenge 1  How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

- Designing function $h : \text{assignments} \rightarrow \text{cells}$ (hashing)
Challenge 1  How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

- Designing function $h : \text{assignments} \rightarrow \text{cells}$ (hashing)
- Deterministic $h$ unlikely to work
Challenges

**Challenge 1** How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

- Designing function $h: \text{assignments} \rightarrow \text{cells}$ (hashing)
- Deterministic $h$ unlikely to work
- Choose $h$ randomly from a large family $H$ of hash functions

2-wise Independent Hashing [CW77]
2-wise Independent Hash Functions

- 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.
  - $X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2}$
2-wise Independent Hash Functions

- 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
  - \( X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2} \)
- 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)
  
  \vdots
  
  X_1 \oplus X_2 \oplus X_5 \cdots \oplus X_{n-2} = 1 \quad (Q_m)
  \]
- Solutions in a cell: \( F \land Q_1 \cdots \land Q_m \)
Challenges

**Challenge 1** How to partition into *roughly equal small* cells of solutions without knowing the distribution of solutions?

Random XOR-based Hash Functions

**Challenge 2** How many cells?
Challenge 2: How many cells?

- A cell is small if it has $\approx \text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells

[CMV13, CMV16]
Challenge 2: How many cells?

- A cell is small if it has $\approx \text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells

Theorem: $\Pr_{\|\text{Sol}(F)\| > 1 + \varepsilon} \leq \text{ApproxMC}(F, \varepsilon, \delta) \leq \|\text{Sol}(F)\|(1 + \varepsilon) \geq 1 - \delta$
Challenge 2: How many cells?

- A cell is small if it has $\approx \text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells

No

$\# \text{ of sols} \leq \text{thresh}?$

$F \land Q_1$

$\# \text{ of sols} \leq \text{thresh}?$
Challenge 2: How many cells?

- A cell is small if it has $\approx$ $\text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells

$$\# \text{ of sols} \leq \text{thresh} ?$$

$$\text{No}$$

$$F \land Q_1$$

$$\# \text{ of sols} \leq \text{thresh} ?$$

$$\text{No}$$

$$F \land Q_1 \land Q_2$$

[CMV13,CMV16]
Challenge 2: How many cells?

- A cell is small if it has $\approx \text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells

[CMV13, CMV16]
Challenge 2: How many cells?

- A cell is small if it has \( \approx \text{thresh} = 5(1 + \frac{1}{\varepsilon})^2 \) solutions
- Many solutions \( \implies \) Many cells & Fewer solutions \( \implies \) Fewer cells
Challenge 2: How many cells?

- A cell is small if it has ≈ \( \text{thresh} = 5(1 + \frac{1}{\epsilon})^2 \) solutions
- Many solutions \implies\ Many cells & Fewer solutions \implies\ Fewer cells

\[
\text{Estimate} = \text{# of sols} \times \text{# of cells}
\]

**Theorem:** \( \Pr \left[ \frac{|\text{Sol}(F)|}{1+\epsilon} \leq \text{ApproxMC}(F, \epsilon, \delta) \leq |\text{Sol}(F)|(1 + \epsilon) \right] \geq 1 - \delta \)
ApproxMC: Early Years (2013-16)

Handle **reasonable** formulas: **reasonable** grids, **reasonable** programs

**2019:** CP-13 paper selected as one of the 25 papers across 25 years of CP conference
Handle *reasonable* formulas: *reasonable* grids, *reasonable* programs

2019: CP-13 paper selected as one of the 25 papers across 25 years of CP conference

**B. Cook:** Virtuous cycle: application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. Around and around.

The definition of “*reasonable*” changes after every iteration of the cycle
Mission 2025: Constrained Counting and Sampling Revolution

Requires combinations of ideas from theory, statistics and systems

2025 Target: $100 \times$ speedup over 2016
ApproxMC: In Pursuit of Scalability

1896 instances from diverse applications
All experiments on 2022 hardware
ApproxMC: In Pursuit of Scalability

1896 instances from diverse applications
All experiments on 2022 hardware
ApproxMC: In Pursuit of Scalability

1896 instances from diverse applications
All experiments on 2022 hardware

2016 630 instances, each in $\leq 5000$ seconds
2022 950 instances, each in $\leq 1$ second
ApproxMC: In Pursuit of Scalability

1896 instances from diverse applications
All experiments on 2022 hardware

\[ \begin{array}{c|c}
\text{Year} & \text{Instances} \\
\hline
2016 & 630 \text{ instances, each in } \leq 5000 \text{ seconds} \\
2022 & 950 \text{ instances, each in } \leq 1 \text{ second} \\
\end{array} \]

Time taken (seconds) for an instance
\[ \begin{array}{c|c|c|c|c}
\text{Year} & 2016 & 2019 & 2020 & 2022 \\
\hline
\text{Time taken} & 3552.16 & 32.83 & 19.59 & 0.15 \\
\end{array} \]

A speedup of \(20,000\times\) over 2016

Still provides \((\varepsilon, \delta)\)-guarantees
In Pursuit of Scalability

<table>
<thead>
<tr>
<th>Theoretical Advances</th>
<th>Algorithmic Engineering</th>
<th>Software Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse hashing</td>
<td>Duality</td>
<td>DNF</td>
</tr>
<tr>
<td>SAT-20, LICS-20</td>
<td>CP-19</td>
<td>CP-18, IJCAI-19, PODS-21,22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Phase Transition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IJCAI-16,17,19, CP-20</td>
</tr>
<tr>
<td>Indep Supp</td>
<td>Chain Formulas</td>
<td>Symmetry</td>
</tr>
<tr>
<td>Constraints-16, ICCAD-22</td>
<td>IJCAI-15, NeurIPS-20</td>
<td>TACAS-20, AAAI-21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Prob. Caching</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IJCAI-19, AAAI-23</td>
</tr>
<tr>
<td>CNF-XOR</td>
<td>Pseudo-Boolean</td>
<td>MaxSAT-XOR</td>
</tr>
<tr>
<td>AAAI-19, CAV-20</td>
<td>CP-21</td>
<td>KR-21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hardware Accelerator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAT-21</td>
</tr>
</tbody>
</table>
In Pursuit of Scalability

<table>
<thead>
<tr>
<th>Theoretical Advances</th>
<th>Algorithmic Engineering</th>
<th>Software Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse Hashing</td>
<td>Duality</td>
<td>Phase Transition</td>
</tr>
<tr>
<td>SAT-20</td>
<td>CP-19</td>
<td>IJCAI-16,17,19</td>
</tr>
<tr>
<td>LICS-20</td>
<td>CP-19</td>
<td>CP-20</td>
</tr>
<tr>
<td>Indep Supp</td>
<td>Chain Formulas</td>
<td>Symmetry</td>
</tr>
<tr>
<td>Constraints-16</td>
<td>IJCAI-15</td>
<td>TACAS-20, AAAI-21</td>
</tr>
<tr>
<td>ICCAD-22</td>
<td>NeurIPS-20</td>
<td></td>
</tr>
<tr>
<td>CNF-XOR</td>
<td>Pseudo-Boolean</td>
<td>MaxSAT-XOR</td>
</tr>
<tr>
<td>AAAI-19</td>
<td>CP-21</td>
<td>KR-21</td>
</tr>
<tr>
<td>CAV-20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hardware Accelerator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAT-21</td>
</tr>
</tbody>
</table>

Slide 23/37
How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

2-wise Independent Hash Functions

- Choose m random XORs: $Q_1, Q_2, \ldots, Q_m$
- Solutions in a cell: $F \land Q_1 \land \ldots \land Q_m$
How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

2-wise Independent Hash Functions

- Choose m random XORs: $Q_1, Q_2, \ldots, Q_m$
- Solutions in a cell: $F \land Q_1 \land \cdots \land Q_m$

Need to handle $F \land \bigwedge_{	ext{CNF}} Q_1 \cdots \land \bigwedge_{	ext{XOR}} Q_m$
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose m random XORs: \( Q_1, Q_2, \ldots, Q_m \)
- Solutions in a cell: \( F \land Q_1 \land \ldots \land Q_m \)

Need to handle \( F \land (Q_1 \lor \ldots \lor Q_m) \)

Performance of state of the art SAT solvers depends on the formulas

SAT Solvers \( \neq \) SAT oracles
New Architecture for CNF-XOR Formulas

Modern SAT Solvers: Conflict-Driven Clause Learning (CDCL) paradigm
  • Guess an assignment to subset of variables, if conflict, remember the reason
  • Continue until satisfiable/unsatisfiable

CDCL and XORs: Random XORs are hard for CDCL in theory and practice
  • But there is a polynomial time procedure: Gauss-Jordan Elimination
New Architecture for CNF-XOR Formulas

Modern SAT Solvers: Conflict-Driven Clause Learning (CDCL) paradigm
- Guess an assignment to subset of variables, if conflict, remember the reason
- Continue until satisfiable/unsatisfiable

CDCL and XORs: Random XORs are hard for CDCL in theory and practice
- But there is a polynomial time procedure: Gauss-Jordan Elimination

\[
\begin{array}{|c|c|c|c|}
\hline
\text{level} & \text{dec} & \text{prop} \\
\hline
0 & x_1 & \\
1 & x_3 & \rightarrow x_5 \\
2 & x_4 & \rightarrow x_2, \neg x_5 \\
\hline
\end{array}
\]

CDCL

\[
\begin{array}{|c|c|c|c|}
\hline
x_1 & x_2 & x_3 & \\
\hline
1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 \\
\hline
\end{array}
\]

Gauss-Jordan Elimination
New Architecture for CNF-XOR Formulas

Modern SAT Solvers: Conflict-Driven Clause Learning (CDCL) paradigm
- Guess an assignment to subset of variables, if conflict, remember the reason
- Continue until satisfiable/unsatisfiable

CDCL and XORs: Random XORs are hard for CDCL in theory and practice
- But there is a polynomial time procedure: Gauss-Jordan Elimination

\[
\begin{array}{l}
\text{level} & \text{dec} & \text{prop} \\
0 & x_1 & \\
1 & x_3 & \rightarrow x_5 \\
2 & x_4 & \rightarrow x_2, \neg x_5
\end{array}
\]

\[
\begin{array}{ccc|c}
 x_1 & x_2 & x_3 & \\
1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0
\end{array}
\]

"Delta" Trail
"Delta" Propagations
"Delta" Conflicts

Incremental CDCL  Incremental Gauss-Jordan Elimination
Modern SAT Solvers: Conflict-Driven Clause Learning (CDCL) paradigm
- Guess an assignment to subset of variables, if conflict, remember the reason
- Continue until satisfiable/unsatisfiable

CDCL and XORs: Random XORs are hard for CDCL in theory and practice
- But there is a polynomial time procedure: Gauss-Jordan Elimination

Engineering an efficient CDCL-GJE solver
- Data-structures for efficient propagation and conflict analysis
- Supervised machine learning-guided heuristics
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose $m$ random XORs: $Q_1, Q_2, \ldots, Q_m$
- Solutions in a cell: $F \land Q_1 \land \cdots \land Q_m$

✓ Challenge 1 Need to handle CNF-XOR formulas

Software Development Specialized CDCL-GJE Solver with Data-Driven Heuristics

SAT Solvers $\neq$ SAT oracles: Performance degrades with increase in the size of XORs
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose \( m \) random XORs: \( Q_1, Q_2, \ldots Q_m \)
- Solutions in a cell: \( F \land Q_1 \land \cdots \land Q_m \)

✓ Challenge 1 Need to handle CNF-XOR formulas

Software Development Specialized CDCL-GJE Solver with Data-Driven Heuristics

SAT Solvers \( \neq \) SAT oracles: Performance degrades with increase in the size of XORs

- Pick every \( X_i \) with prob. \( \frac{1}{2} \) and XOR them
- Expected size of each XOR: \( \frac{n}{2} \)
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose m random XORs: \(Q_1, Q_2, \ldots, Q_m\)
- Solutions in a cell: \(F \land Q_1 \land \cdots \land Q_m\)

✓ Challenge 1 Need to handle CNF-XOR formulas

Software Development Specialized CDCL-GJE Solver with Data-Driven Heuristics

SAT Solvers \(\neq\) SAT oracles: Performance degrades with increase in the size of XORs

- Pick every \(X_i\) with prob. \(\frac{1}{2}\) and XOR them
- Expected size of each XOR: \(\frac{n}{2}\)

Challenge 2 Do we have really to pick every variable \(X_i\) with prob \(\frac{1}{2}\)?
Not All Variables Matter Equally

- Not all variables are required to specify solution space of $F$
  - $F := X_3 \iff (X_1 \lor X_2)$
  - $X_1$ and $X_2$ uniquely determines rest of the variables (i.e., $X_3$)

- $I \subseteq X$ is independent support if it suffices to determine the solution space
  - $\{X_1, X_2\}$ is independent support
Not All Variables Matter Equally

- Not all variables are required to specify solution space of $F$
  - $F := X_3 \iff (X_1 \lor X_2)$
  - $X_1$ and $X_2$ uniquely determines rest of the variables (i.e., $X_3$)

- $I \subseteq X$ is independent support if it suffices to determine the solution space
  - $\{X_1, X_2\}$ is independent support

- Random XORs need to be constructed only over $I$  
  - Typically $I$ is 1-2 orders of magnitude smaller than $X$

[CMV14]
Not All Variables Matter Equally

- Not all variables are required to specify solution space of $F$
  - $F := X_3 \iff (X_1 \lor X_2)$
  - $X_1$ and $X_2$ uniquely determines rest of the variables (i.e., $X_3$)

- $I \subseteq X$ is independent support if it suffices to determine the solution space
  - $\{X_1, X_2\}$ is independent support

- Random XORs need to be constructed only over $I$ [CMV14]

- Typically $I$ is 1-2 orders of magnitude smaller than $X$

Algorithmic procedure to determine $I$?
Not All Variables Matter Equally

- Not all variables are required to specify solution space of $F$
  - $F := X_3 \iff (X_1 \lor X_2)$
  - $X_1$ and $X_2$ uniquely determines rest of the variables (i.e., $X_3$)

- $\mathcal{I} \subseteq X$ is independent support if it suffices to determine the solution space
  - $\{X_1, X_2\}$ is independent support

- Random XORs need to be constructed only over $\mathcal{I}$

- Typically $\mathcal{I}$ is 1-2 orders of magnitude smaller than $X$

Algorithmic procedure to determine $\mathcal{I}$?

- Approach I: $\log n$ calls to SAT solver via reduction to GMUS
  [CMV14]
  Best Student Paper, CP15

- Approach II: $n$ easy calls to SAT solver via Padoa’s theorem
  [SM22]

- Approach II + ApproxMC is up to 100 times faster than Approach I + ApproxMC

SAT Solvers $\neq$ SAT Oracles
Not All Variables Matter Equally

- Not all variables are required to specify solution space of $F$
  - $F := X_3 \iff (X_1 \lor X_2)$
  - $X_1$ and $X_2$ uniquely determines rest of the variables (i.e., $X_3$)

- $I \subseteq X$ is independent support if it suffices to determine the solution space
  - $\{X_1, X_2\}$ is independent support

- Random XORs need to be constructed only over $I$
  
- Typically $I$ is 1-2 orders of magnitude smaller than $X$

Algorithmic procedure to determine $I$?

- Approach I: $\log n$ hard calls to SAT solver via reduction to GMUS
  - [IMMV15]
  - Best Student Paper, CP15

- Approach II: $n$ easy calls to SAT solver via Padoa's theorem
  - [SM22]

- Approach II + ApproxMC is up to 100 × faster than Approach I + ApproxMC

SAT Solvers $\neq$ SAT Oracles
Not All Variables Matter Equally

- Not all variables are required to specify solution space of $F$
  - $F := X_3 \iff (X_1 \lor X_2)$
  - $X_1$ and $X_2$ uniquely determines rest of the variables (i.e., $X_3$)

- $\mathcal{I} \subseteq X$ is independent support if it suffices to determine the solution space
  - $\{X_1, X_2\}$ is independent support

- Random XORs need to be constructed only over $\mathcal{I}$

- Typically $\mathcal{I}$ is 1-2 orders of magnitude smaller than $X$

Algorithmic procedure to determine $\mathcal{I}$?

- Approach I: $\log n$ hard calls to SAT solver via reduction to GMUS
  - [IMMV15]
  - Best Student Paper, CP15

- Approach II: $n$ easy calls to SAT solver via Padoa’s theorem
  - [SM22]

Approach II + ApproxMC is up to $100 \times$ faster than Approach I + ApproxMC

SAT Solvers $\neq$ SAT Oracles
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose m random XORs: \( Q_1, Q_2, \ldots Q_m \)
- Solutions in a cell: \( F \land Q_1 \land \cdots \land Q_m \)

✓ **Challenge 1** Need to handle CNF-XOR formulas

**Software Development** Specialized CDCL-GJE Solver with Data-Driven Heuristics

**SAT Solvers \( \neq \) SAT oracles:** Performance degrades with increase in the size of XORs

- Pick every \( X_i \) with prob. \( \frac{1}{2} \) and XOR them
- Expected size of each XOR: \( \frac{n}{2} \)

✓ **Challenge 2** Do we have to really pick every variable \( X_i \) with prob \( \frac{1}{2} \)?

**Algorithmic Engineering** Pick \( X_i \in \mathcal{I} \)
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose m random XORs: $Q_1, Q_2, \ldots, Q_m$
- Solutions in a cell: $F \land Q_1 \cdots \land Q_m$

✓ Challenge 1 Need to handle CNF-XOR formulas

Software Development Specialized CDCL-GJE Solver with Data-Driven Heuristics

SAT Solvers != SAT oracles: Performance degrades with increase in the size of XORs

- Pick every $X_i$ with prob. $\frac{1}{2}$ and XOR them
- Expected size of each XOR: $\frac{n}{2}$

✓ Challenge 2 Do we have to really pick every variable $X_i$ with prob $\frac{1}{2}$?

Algorithmic Engineering Pick $X_i \in \mathcal{I}$

Challenge 3 Do we have to really pick every variable $X_i$ with prob $\frac{1}{2}$?
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose m random XORs: $Q_1, Q_2, \ldots, Q_m$
- Solutions in a cell: $F \land Q_1 \land \cdots \land Q_m$

✓ Challenge 1 Need to handle CNF-XOR formulas

Software Development Specialized CDCL-GJE Solver with Data-Driven Heuristics

SAT Solvers != SAT oracles: Performance degrades with increase in the size of XORs

- Pick every $X_i$ with prob. $\frac{1}{2}$ and XOR them
- Expected size of each XOR: $\frac{n}{2}$

✓ Challenge 2 Do we have to really pick every variable $X_i$ with prob $\frac{1}{2}$?

Algorithmic Engineering Pick $X_i \in I$

Challenge 3 Do we have to really pick every variable $X_i$ with prob $\frac{1}{2}$?

- If we pick with prob $p < \frac{1}{2}$, then no guarantees of 2-wise independence
- $Z_m$: Number of solutions in a randomly chosen cell
- $2$-wise independence $\iff \frac{\text{Var}[Z_m]}{\text{E}[Z_m]} \leq 1 \iff$ Concentration bounds
Beyond 2-wise Independence

Open problem (2013-19): Sparse \((p < \frac{1}{2})\) XORs that work in theory and practice

Theorem (Log-Sparse XORs suffice)

If we pick \(m\)-th XOR with \(p^m = \log m\), we have

\[
\text{Var}[Z_m] \leq 1.
\]

Improvement of \(p\) from \(m/2\) to \(\log m\)

\[
\text{Var}[Z_m] \leq 1 + \left| \text{Sol}(F) \right| - 1 \cdot \frac{X_{\sigma_1 \in \text{Sol}(F)} X_{\sigma_2 \in \text{Sol}(F)} w = d(\sigma_1, \sigma_2)}
\approx \text{collision probability} \right| \left\{ r(w, p^m) \right\}
\leq 1 + \sum_{w=0}^{n} \left| C_F(w) r(w, p^m) \right|
\]

Earlier Attempts \cite{EGSS14,ZCSE16,AD17,ATD18}

\[\text{Rewrite } X_{\sigma_1 \in \text{Sol}(F)} X_{\sigma_2 \in \text{Sol}(F)} w = d(\sigma_1, \sigma_2) = \{ \sigma_1, \sigma_2 \in \text{Sol}(F) \} = C_F(w)\]

Isoperimetric Inequalities: Possible to bound \(C_F(w)\) if bound on \(\left| \text{Sol}(F) \right|\) is known

Barrier: But \(\left| \text{Sol}(F) \right|\) can be arbitrarily large

Key Idea: In the context of \(Z_m\), it suffices to assume \(\left| \text{Sol}(F) \right| < 2^m + u\) for small \(u\).
Beyond 2-wise Independence

Open problem (2013-19): Sparse \((p < \frac{1}{2})\) XORs that work in theory and practice

**Theorem (Log-Sparse XORs suffice)**

If we pick \(m\)-th XOR with \(p_m = \frac{\log m}{m}\), we have

\[
\frac{\text{Var}[Z_m]}{\text{E}[Z_m]} \leq 1.1
\]

**Improvement of \(p\) from \(\frac{m/2}{m}\) to \(\frac{\log m}{m}\)**
Beyond 2-wise Independence

Open problem (2013-19): Sparse \((p < \frac{1}{2})\) XORs that work in theory and practice

Theorem (Log-Sparse XORs suffice)

If we pick \(m\)-th XOR with \(p_m = \frac{\log m}{m}\), we have \(\frac{\text{Var}[Z_m]}{\text{E}[Z_m]} \leq 1.1\)

Improvement of \(p\) from \(\frac{m/2}{m}\) to \(\frac{\log m}{m}\)

\[
\frac{\text{Var}[Z_m]}{\text{E}[Z_m]} \leq 1 + |\text{Sol}(F)|^{-1} \cdot \sum_{\sigma_1 \in \text{Sol}(F)} \sum_{\sigma_2 \in \text{Sol}(F)} \sum_{w=d(\sigma_1,\sigma_2)} \approx \text{collision probability} \cdot r(w, p_m)
\]
Beyond 2-wise Independence

Open problem (2013-19): Sparse \( (p < \frac{1}{2}) \) XORs that work in theory and practice

**Theorem (Log-Sparse XORs suffice)**

If we pick \( m \)-th XOR with \( p_m = \frac{\log m}{m} \), we have \( \frac{\text{Var}[Z_m]}{\text{E}[Z_m]} \leq 1.1 \)

**Improvement of \( p \) from \( \frac{m/2}{m} \) to \( \frac{\log m}{m} \)**

\[
\frac{\text{Var}[Z_m]}{\text{E}[Z_m]} \leq 1 + \frac{1}{|\text{Sol}(F)|} \cdot \sum_{\sigma_1 \in \text{Sol}(F)} \sum_{\sigma_2 \in \text{Sol}(F)} \sqrt{r(w, p_m)} \approx \text{collision probability} \leq 1 + \sum_{w=0}^{n} \binom{n}{w} r(w, p_m)
\]

**Earlier Attempts**

[EGSS14, ZCSE16, AD17, ATD18]
Beyond 2-wise Independence

Open problem (2013-19): Sparse ($p < \frac{1}{2}$) XORs that work in theory and practice

Theorem (Log-Sparse XORs suffice)

If we pick $m$-th XOR with $p_m = \frac{\log m}{m}$, we have $\frac{\text{Var}[Z_m]}{\mathbb{E}[Z_m]} \leq 1.1$

Improvement of $p$ from $\frac{m/2}{m}$ to $\frac{\log m}{m}$

$$\frac{\text{Var}[Z_m]}{\mathbb{E}[Z_m]} \leq 1 + \frac{1}{|\text{Sol}(F)|} \cdot \sum_{\sigma_1 \in \text{Sol}(F)} \sum_{\sigma_2 \in \text{Sol}(F)} \sum_{w = d(\sigma_1, \sigma_2)} r(w, p_m) \approx \text{collision probability} \leq 1 + \sum_{w=0}^{n} \binom{n}{w} r(w, p_m)$$

Earlier Attempts

[EGSS14, ZCSE16, AD17, ATD18]

Rewrite

$$\sum_{\sigma_1 \in \text{Sol}(F)} \sum_{\sigma_2 \in \text{Sol}(F)} r(w, p_m) = \sum_{w=0}^{n} C_F(w) r(w, p_m)$$

$$C_F(w) = |\{\sigma_1, \sigma_2 \in \text{Sol}(F) | d(\sigma_1, \sigma_2) = w\}|$$

Isoperimetric Inequalities: Possible to bound $C_F(w)$ if bound on $|\text{Sol}(F)|$ is known

Barrier: But $|\text{Sol}(F)|$ can be arbitrarily large
Beyond 2-wise Independence

Open problem (2013-19): Sparse \((p < \frac{1}{2})\) XORs that work in theory and practice

**Theorem (Log-Sparse XORs suffice)**

If we pick \(m\)-th XOR with \(p_m = \frac{\log m}{m}\), we have \(\frac{\text{Var}[Z_m]}{E[Z_m]} \leq 1.1\)

**Improvement of \(p\) from \(\frac{m/2}{m}\) to \(\frac{\log m}{m}\)**

\[
\frac{\text{Var}[Z_m]}{E[Z_m]} \leq 1 + \frac{|\text{Sol}(F)|^{-1}}{|\text{Sol}(F)|} \cdot \sum_{\sigma_1 \in \text{Sol}(F)} \sum_{\sigma_2 \in \text{Sol}(F)} \sum_{w=d(\sigma_1, \sigma_2)} r(w, p_m) \approx \text{collision probability}
\]

\[
\leq 1 + \sum_{w=0}^{n} \binom{n}{w} r(w, p_m)
\]

Earlier Attempts

[EGSS14, ZCSE16, AD17, ATD18]

Rewrite

\[
\sum_{\sigma_1 \in \text{Sol}(F)} \sum_{\sigma_2 \in \text{Sol}(F)} r(w, p_m) = \sum_{w=0}^{n} C_F(w) r(w, p_m)
\]

\[
C_F(w) = |\{\sigma_1, \sigma_2 \in \text{Sol}(F) \mid d(\sigma_1, \sigma_2) = w\}|
\]

**Isoperimetric Inequalities:** Possible to bound \(C_F(w)\) if bound on \(|\text{Sol}(F)|\) is known

**Barrier:** But \(|\text{Sol}(F)|\) can be arbitrarily large

**Key Idea:** In the context of \(Z_m\), It suffices to assume \(|\text{Sol}(F)| < 2^{m+u}\) for small \(u\).
Challenges in Pursuit of Scalability

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

2-wise Independent Hash Functions

- Choose m random XORs: \( Q_1, Q_2, \ldots, Q_m \)
- Solutions in a cell: \( F \land Q_1 \land \cdots \land Q_m \)

✓ Challenge 1 Need to handle CNF-XOR formulas

Software Development Specialized CDCL-GJE Solver with Data-Driven Heuristics

SAT Solvers \( \neq \) SAT oracles: Performance degrades with increase in the size of XORs

- Pick every \( X_i \) with prob. \( \frac{1}{2} \) and XOR them
- Expected size of each XOR: \( \frac{n}{2} \)

✓ Challenge 2 Do we have really to pick every variable \( X_i \) with prob \( \frac{1}{2} \)?

Algorithmic Engineering Pick \( X_i \in \mathcal{I} \)

✓ Challenge 3 Do we have really to pick every variable \( X_i \) with prob \( \frac{1}{2} \)?

Theoretical Advances Pick m-th XOR with \( p_m = \frac{\log m}{m} \)
In the Pursuit of Scalability

<table>
<thead>
<tr>
<th>Theoretical Advances</th>
<th>Algorithmic Engineering</th>
<th>Software Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse Hashing</td>
<td>Duality</td>
<td>Phase Transition</td>
</tr>
<tr>
<td>SAT-20, LICS-20</td>
<td>CP-19</td>
<td>IJCAI-16, 17, 19</td>
</tr>
<tr>
<td></td>
<td>CP-18, IJCAI-19, PODS-21, 22</td>
<td>CP-20</td>
</tr>
<tr>
<td>Indep Supp</td>
<td>Chain Formulas</td>
<td>Symmetry</td>
</tr>
<tr>
<td>Constraints-16, ICCAD-22</td>
<td>IJCAI-15</td>
<td>TACAS-20, AAAI-21</td>
</tr>
<tr>
<td></td>
<td>NeurIPS-20</td>
<td></td>
</tr>
<tr>
<td>CNF-XOR</td>
<td>Pseudo-Boolean</td>
<td>MaxSAT-XOR</td>
</tr>
<tr>
<td>AAAI-19, CAV-20</td>
<td>CP-21</td>
<td>KR-21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hardware Accelerator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAT-21</td>
</tr>
</tbody>
</table>
Reliability of Critical Infrastructure Networks

Figure: Plantersville, SC

Timeout = 1000 seconds
Timeout = 1000 seconds
Timeout = 1000 seconds

Impact: The first theoretically sound estimates of resilience in power transmission networks of ten medium sized cities in US
ApproxMC: Progress over the years

Time taken (seconds) for an instance

- **2016**: 3552.16
- **2019**: 32.83
- **2020**: 19.59
- **2022**: 0.15

A speedup of 20,000× over 2016

1896 benchmarks from diverse applications
Another Iteration of Virtuous Cycle

**B. Cook, 2022:** Virtuous cycle: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. Around and around.
Another Iteration of Virtuous Cycle

B. Cook, 2022: **Virtuous cycle**: ...application areas drives more investment in foundational tools, while improvements in the foundational tools drive further applications. Around and around.
**Generalizability**

**Union of Sets** ApproxMC is Fully Polynomial Randomized Approximation Scheme (FPRAS) – fundamentally different from the Monte-Carlo based FPRAS

- IJCAI-19 Sister Conferences Best Paper Award Track [MSV19]
Generalizability

**Union of Sets** ApproxMC is Fully Polynomial Randomized Approximation Scheme (FPRAS) – fundamentally different from the Monte-Carlo based FPRAS

- IJCAI-19 Sister Conferences Best Paper Award Track [MSV19]

**Streaming** Counting over a stream: Distinct Elements

Example: How many unique customers visit website?
Fundamental problem in Databases

- CACM Research Highlights [PVBM21]
- ACM SIGMOD 2022 Research Highlight
- “Best of PODS 2021” by ACM TODS
**Generalizability**

**Union of Sets**  ApproxMC is Fully Polynomial Randomized Approximation Scheme (FPRAS) – fundamentally different from the Monte-Carlo based FPRAS

- IJCAI-19 Sister Conferences Best Paper Award Track

**Streaming**  Counting over a stream: Distinct Elements

Example: How many unique customers visit website?

Fundamental problem in Databases

- CACM Research Highlights
- ACM SIGMOD 2022 Research Highlight
- “Best of PODS 2021” by ACM TODS

**Unsatisfiable Subsets**  Count minimal subsets of clauses that are unsatisfiable.

Diagnosis metric for systems

- “Best Papers of CAV-20” by FMSD
In Pursuit of Scalability

Counting over the years
In Pursuit of Scalability

Counting over the years

Sampling over the years

Synthesis over the years

ICCAD-21 & DATE-23 Best Paper Award Nomination
Where do we go from here?
Where do we go from here?

The Quest for Scalability is Endless

Today’s Counters/Samplers/Synthesis Engines $\approx$ SAT Solvers in early 2000s

Industrial Practice: $100 \times$ Speedup
The Pursuit of Scalability

Mission 2028: 100× Speedup for Counting, Sampling, and Synthesis

Challenge Problems (for Counting)

Civil Engineering  Rigorous resilience estimation for power grid of Los Angeles
Quantitative Evaluation  Binarized neural network with 1M neurons
Software Engineering  Information Flow analysis of programs with 10K lines of code

Technical Directions (for Counting)

Theoretical Advances  Native reasoning over expressive theories (*Beyond SMT*)
Algorithmic Engineering  Machine Learning-guided heuristic design
Software Development  Hardware-accelerator aware tools
The Pursuit of Scalability

Mission 2028: $100 \times$ Speedup for Counting, Sampling, and Synthesis

Challenge Problems (for Counting)

Civil Engineering  Rigorous resilience estimation for power grid of Los Angeles
Quantitative Evaluation  Binarized neural network with 1M neurons
Software Engineering  Information Flow analysis of programs with 10K lines of code

Technical Directions (for Counting)

Theoretical Advances  Native reasoning over expressive theories (*Beyond SMT*)
Algorithmic Engineering  Machine Learning-guided heuristic design
Software Development  Hardware-accelerator aware tools

Certification: Approximate count is “correct” or the distribution generated is correct

- Applications to verification of probabilistic programming
- Building on recent advances in distribution testing
It Takes a Village

Research Group

Durgesh Agrawal
Lorenzo Ciampiconi
Priyanka Golia
Gunjan Kumar
Yash Pote
Tim van Bremen
Teodora Baluta
Alexis de Colnet
Rahul Gupta
Lawqueen Kanesh
Shubham Sharma
Jiong Yang
Jaroslav Bendik
Paulius Dilkas
Yacine Izza
Yong Lai
Mate Soos
Suweit Yang
Bhavishya
Bishwamittra Ghosh
Md Mohimenul Kabir
Anna Latour
Arijit Shaw

Collaborators

S. Akshay (IITB, IN)
Rajkishore Barik(Intel, US)
Fabrizio Biondi (CS, FR)
Sourav Chakraborty(ISIK, IN)
Zheng Leong Chua(IP, SG)
A. Dileep(IITD, India)
Michael A. Enescu(Inria, FR)
Sutanu Gayen(NUS, SG)
Alexey Ignatiev(ULisboa, PT)
Mohan S. Kankanhalli(NUS, SG)
Axel Legay (UCL, BE)
Sharad Malik(Princeton, US)
John M. Mellor-Crummey(Rice, US)
Karthik Murthy(Rice, US)
Sri Raj Paul(Rice, US)
Nicolas Prevot(London, UK)
Ammar F. Sabili(NUS, SG)
Jonathan Scarlett(NUS, SG)
Shweta Shinde(ETH, CH)
Harold Soh(NUS, SG)
N. V. Vinodchandran(UNL, US)
Yaqi Xie(NUS, SG)

Alyas Almaawi(UtAustion, US)
Debabrota Basu(Chalmers, US)
Kian Ming Adam Chai(DSO, SG)
Supratik Chakraborty (IITB, IN)
Tiago Cogumbreiro(Rice, US)
Jeffrey M. Dudek(Rice, US)
Daniel J. Fremont(UCB, US)
Stephan Gocht (Lund U., SE)
Alexander Ivrii(IBM, IL)
Sarfraz Khurshid(UtAustion, US)
Massimo Lupascu(NUS, SG)
Dmitry Malioutov(IBM, US)
Rakesh Mistry(IITB, IN)
Nina Narodytska(VMware, US)
Jean Quilbeuf(Inria, FR)
Vivek Sarkar(Rice, US)
Sanjit A. Seshia(UCB, US)
Aditya A. Shrotri(Rice, US)
Muhammad Usman(UtAustion, US)
Kaiyuan Wang(Google, US)
Ziwei Xu(NUS, SG)
Eduard Baranov(UCLouvain, BE)
Arnab Bhattacharya (NUS, SG)
Diptarka Chakraborty(NUS, SG)
Davin Choo(DSO, SG)
Vincent Derkinderen(KUL, BE)
Leonardo Duenas-Osorio (Rice, US)
Dror Fried (Open U., IL)
Annelie Heuser(CNRS, FR)
Saurabh Joshi(IITH, IN)
Raghat Kulkarni(CMI, IN)
Deepak Majeti(Rice, US)
Joao Marques-Silva(ANITI, FR)
M. Mohammadalitajrishi(Polymlt, CA)
Roger Paredes(Rice, US)
Gilles Pesant(Polymlt, CA)
Subhajit Roy (IITK, IN)
Prateek Saxena(NUS, SG)
Shiqi Shen(NUS, SG)
Friedrich Slivovsky(TU Wien, AT)
Moshe Y. Vardi(Rice, US)
Wenxi Wang(UtAustion, US)
Roland H. C. Yap(NUS, SG)

Funding Agencies: National Research Foundation, Ministry of Education, Defense Service Organization
Industrial Support: Grab Taxiholdings, Amazon, Microsoft Research
Counting, Sampling, and Synthesis

```c
PC2 (char[] SP, char[] UI) {
    match = true;
    for (int i=0; i<UI.length(); i++) {
        if (SP[i] != UI[i]) match=false;
        else match = match;
    }
    if (match) return Yes;
    else return No;
}
```

These slides are available at tinyurl.com/meel-talk
Detailed Future Directions

Applications: Infrastructure Resilience, Information Leakage, Prob. Databases, Configuration Testing, Partition Function, BNN Verification

Theoretical Advances
- Formula-based Sparse-XORs
- Revisiting FPRAS
- Parameterized Complexity
  - DNF, Minimal Solutions, Chain formula
  - Permanent, Automata, Linear Extensions
  - Addition of XORs
- Streaming
- Synthesis
- Entropy
- Algorithmic Engineering
  - Incremental
  - Bit-vectors
  - Heuristic
  - Distributed
  - SMT Synthesis
  - Beyond Qualitative Synthesis
  - Software Development
  - Tighter Integration
  - Hybrid Constraints
  - XOR Handling
  - Accelerators
  - Knowledge Compilation
  - Certification
  - Distribution
  - Counting

SMT Synthesis
- SMT Formula Learning
- Optimal Functions, Approximate Synthesis

Software Development
- Multiple Queries
- Callbacks
- PB-XOR, BNN-XOR, MaxSAT-XOR, ASP-XOR
- GPU
- SMT, Portfolio
- Certificate for Approximation