Quantitative Automated Reasoning: The Quest for Scalability

Kuldeep S. Meel

Georgia Instiutte of Technology and University of Toronto

Quantitative Automated Reasoning: 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

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

$\models$

satisfies

Automated Reasoning

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

$\models$

satisfies

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

Automated Reasoning

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

$\models$

satisfies

$$\mathcal{M}(\mathcal{I},\mathcal{O}) \qquad \models \qquad \mathcal{P}(\mathcal{I},\mathcal{O})$$

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

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

Automated Reasoning

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

$\models$

satisfies

$$\mathcal{M}(\mathcal{I},\mathcal{O}) \qquad \models \qquad \mathcal{P}(\mathcal{I},\mathcal{O})$$

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: $(X_1 \vee \neg X_2 \vee \neg X_3) \wedge (X_2 \vee \neg X_3)$ $\qquad X_1 = 1, X_2 = 1, X_3 = 1$

Automated Reasoning

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

$\models$

satisfies

$\mathcal{M}(\mathcal{I},\mathcal{O}) \qquad \models \qquad \mathcal{P}(\mathcal{I},\mathcal{O})$

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: $(X_1 \vee \neg X_2 \vee \neg X_3) \wedge (X_2 \vee \neg X_3)$ $\qquad X_1 = 1, X_2 = 1, X_3 = 1$

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

Automated Reasoning

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

$\models$

satisfies

$$\mathcal{M}(\mathcal{I},\mathcal{O}) \qquad \models \qquad \mathcal{P}(\mathcal{I},\mathcal{O})$$

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: $(X_1 \vee \neg X_2 \vee \neg X_3) \wedge (X_2 \vee \neg X_3)$ $\quad X_1 = 1, X_2 = 1, X_3 = 1$

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

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

$\models$

satisfies

$$\mathcal{M}(\mathcal{I}, \mathcal{O}) \qquad \models \qquad \mathcal{P}(\mathcal{I}, \mathcal{O})$$

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: $(X_1 \vee \neg X_2 \vee \neg X_3) \wedge (X_2 \vee \neg X_3)$ $\qquad X_1 = 1, X_2 = 1, X_3 = 1$

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

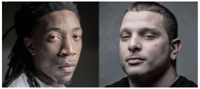
Knuth, 2016: These so-called "SAT solvers" can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

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

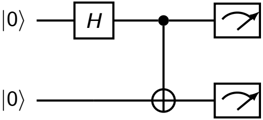
Beyond SAT I: Quantification

```
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;
}
```

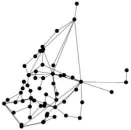
Information Leakage



Fairness



Quantum Simulation

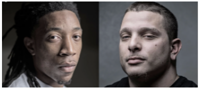


Critical Infrastructure

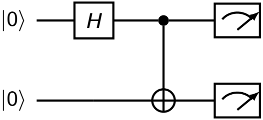
Beyond SAT I: Quantification

```
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;
}
```

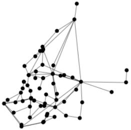
Information Leakage



Fairness



Quantum Simulation

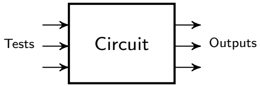


Critical Infrastructure

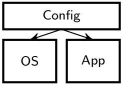
Quantification: How often does $\mathcal{M}$ satisfy $\mathcal{P}$?

Counting

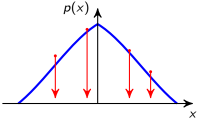
Beyond SAT II: Reasoning about Distributions



Constrained-Random Verification



Configuration Testing



Sampling

- Multiple application domains require *reasoning about distributions*
- Constraints define the structure of relevant input spaces
- **Challenge**: How to efficiently sample from and analyze these distributions?

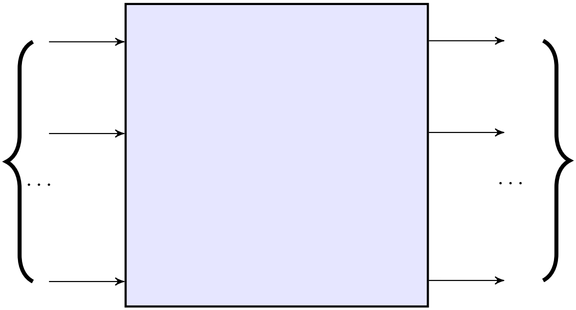
Beyond SAT III: Automated Synthesis

$f(u,v) \geq u;$
$f(u,v) \geq v;$
$f(u,v) = u \vee$
$f(u,v) = v)$

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

age	25
capital-gain	4000
occupation	coach

Inputs
$\mathcal{I}$



Outputs
$\mathcal{O}$

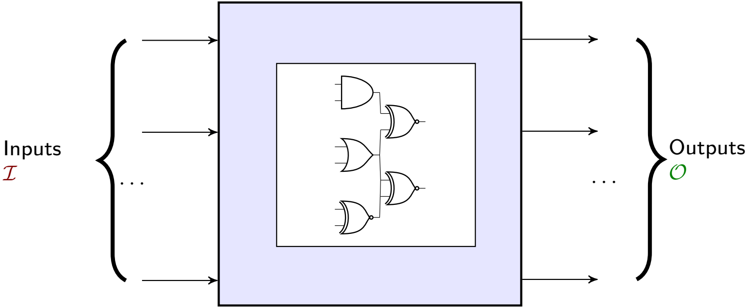
Synthesis

Beyond SAT III: Automated Synthesis

$$\begin{aligned} f(u,v) &\geq u; \\ f(u,v) &\geq v; \\ f(u,v) &= u \vee \\ f(u,v) = v) \end{aligned}$$

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

age	25
capital-gain	4000
occupation	coach



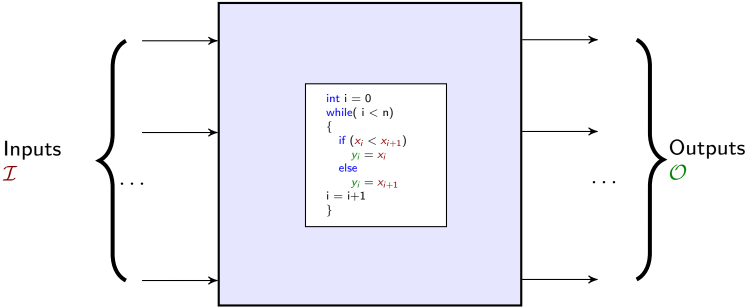
Synthesis

Beyond SAT III: Automated Synthesis

$$f(u,v) \geq u;$$
$$f(u,v) \geq v;$$
$$f(u,v) = u \vee$$
$$f(u,v) = v)$$

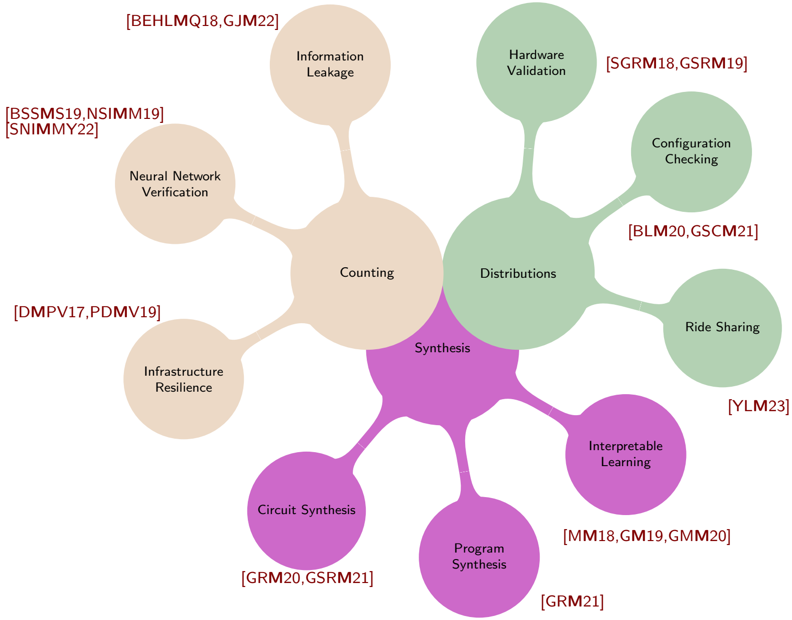
Specification: $\mathcal{P}(\mathcal{I}, \mathcal{O})$

age	25
capital-gain	4000
occupation	coach

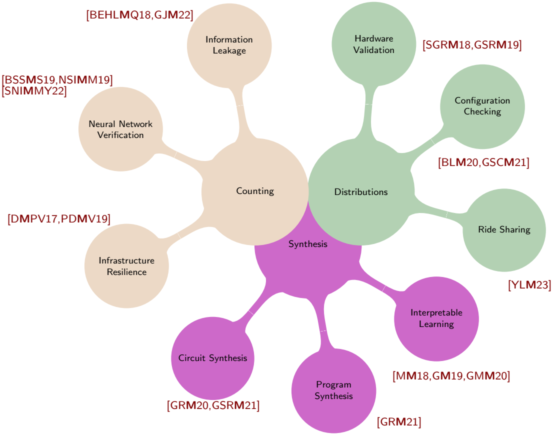


Synthesis

Research Overview



Research Overview



Artificial Intelligence	AAAI:25×, IJCAI:13×, NeurIPS: 6×, KR:2×
Formal Methods/Constraints	CAV:11×, SAT:10×, CP:9×,TACAS: 3×, DAC: 3×, ICCAD: 2×, DATE:
Logic/Databases	PODS:7×, LICS:2×, LPAR:4×,
Software Engineering	ICSE:2×, FSE: 2×, CCS:1×

Research Program's Output

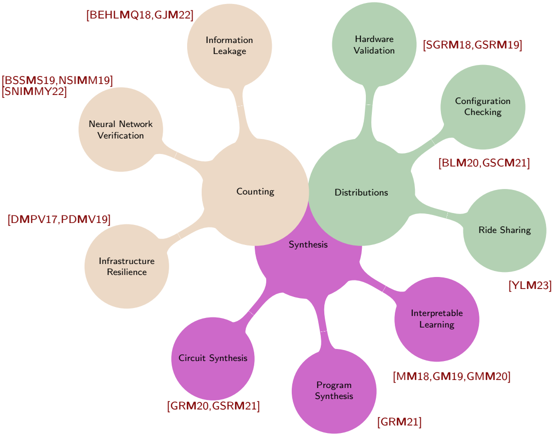
Recent Research Paper Recognition

- Best/Distinguished Paper Awards: CAV-23, CAV-24, and ICLP-24
- Best Paper Award Nomination: ICCAD-22 and DATE-23
- Research Highlights: CACM (2023), ACM SIGMOD (2022)
- Knuth's Article on Distinct Elements (and media coverage)

Job Placement Highlights

- Teodora Baluta (PhD) → Assistant Professor, Georgia Tech
- Priyanka Golia (PhD) → Assistant Professor, CISPA (and IIT Delhi, India)
  ACM India Doctoral Dissertation Award 2024
- Tim van Bremen (post-doc) → Assistant Professor, NTU Singapore
- Anna Latour (post-doc) → Assistant Professor, TU Delft
- Gunjan Kumar (post-doc) → Assistant Professor, IIT Kanpur
- Lawqueen Kanesh (post-doc) → Assistant Professor, IIT Jodhpur
- Yong Lai (post-doc) → Associate Professor, Jilin University

Research Overview



Artificial Intelligence	AAAI:25×, IJCAI:13×, NeurIPS: 6×, KR:2×
Formal Methods/Constraints	CAV:11×, SAT:10×, CP:9×,TACAS: 3×, DAC: 3×, ICCAD: 2×, DATE:
Logic/Databases	PODS:7×, LICS:2×, LPAR:4×,
Software Engineering	ICSE:2×, FSE: 2×, CCS:1×

Today's Focus: Counting

Counting

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

Counting

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

Counting

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

- Generalization to arbitrary weights
  - Given weight function (implicitly represented) W: $\{0,1\}^n \mapsto [0,1]$
  - $W(F) = \Sigma_{y \in \mathrm{Sol}(F)} W(y)$
  - (Weighted) Counting: Determine $W(F)$

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

Today's Menu

Appetizer Applications

- Critical Infrastructure Resilience
- Quantitative Verification of AI Systems
- Quantum Simulation

Main Course ApproxMC: A Scalable Counting Framework

Dessert Future Outlook

Resilience of Critical Infrastructure Networks

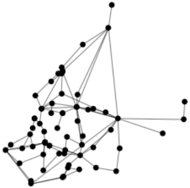
[DMPV17,PDMV19, KM23]

Can we predict the likelihood of a blackout due to natural disaster?

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19, KM23]

Can we predict the likelihood of a blackout due to natural disaster?

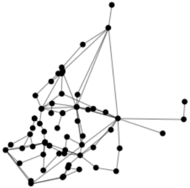


- $G = (V, E)$; set of source nodes S and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[t is disconnected from S]?

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19, KM23]

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 $g : E \to [0, 1]$
- Compute Pr[t is disconnected from S]?

Constrained Counting

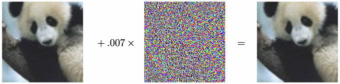
Key Idea: Encode disconnectedness using constraints

Impact: The first theoretically sound estimates of resilience in power transmission networks of ten medium sized cities in US

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22,YTSM25]

Our Focus: Binarized Neural Networks



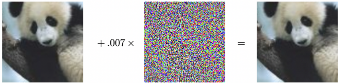
Robustness Quantification

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

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22,YTSM25]

Our Focus: Binarized Neural Networks



Robustness Quantification

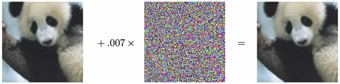
$$\underbrace{\left|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\right|}_{\text{Encode Symbolically}}$$

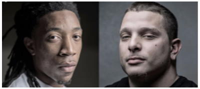
$\boxed{\text{Constrained Counting}}$

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22,YTSM25]

Our Focus: Binarized Neural Networks





Robustness Quantification

$$\underbrace{\left|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\right|}_{\text{Encode Symbolically}}$$

Fairness Quantification

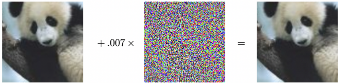
$$\underbrace{\left|\{x : \ \mathcal{N}(x \wedge \textsc{Black}) \neq \mathcal{N}(x \wedge \textsc{White})\}\right|}_{\text{Encode Symbolically}}$$

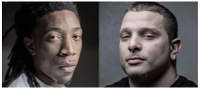
$\boxed{\text{Constrained Counting}}$

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22,YTSM25]

Our Focus: Binarized Neural Networks





Robustness Quantification

$$\underbrace{\left|\{x : \mathcal{N}(x+\varepsilon) \neq \mathcal{N}(x)\}\right|}_{\text{Encode Symbolically}}$$

Fairness Quantification

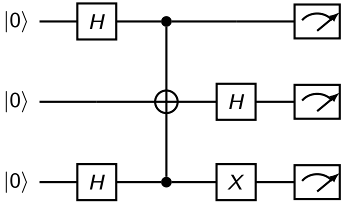
$$\underbrace{\left|\{x :\ \mathcal{N}(x \wedge \textsc{Black}) \neq \mathcal{N}(x \wedge \textsc{White})\}\right|}_{\text{Encode Symbolically}}$$

$\boxed{\text{Constrained Counting}}$

Impact: The first scalable technique for rigorous and certified quantification of robustness and fairness of Binarized Neural Networks

Quantum Circuit Simulation via Model Counting

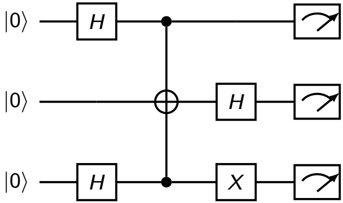
[MBL24]



- Quantum state: superposition of 2^n basis states with amplitudes α_i
- Measurement collapses to classical state with probability $|\alpha_i|^2$
- Challenge: Exponential state space makes simulation intractable

Quantum Circuit Simulation via Model Counting

[MBL24]

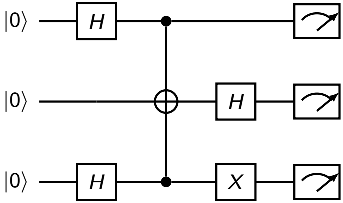


- Quantum state: superposition of 2^n basis states with amplitudes α_i
- Measurement collapses to classical state with probability $|\alpha_i|^2$
- Challenge: Exponential state space makes simulation intractable

- Key Insight: Encode quantum computation as weighted counting problem
  - Boolean formula F encodes valid computation paths
  - Weight function W encodes quantum amplitudes

Quantum Circuit Simulation via Model Counting

[MBL24]



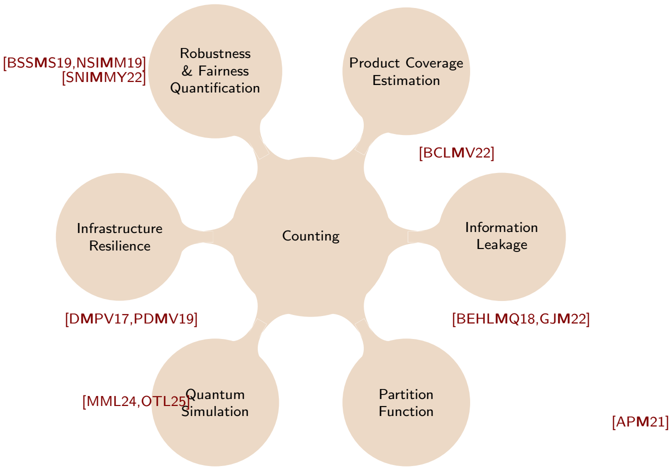
- Quantum state: superposition of 2^n basis states with amplitudes α_i
- Measurement collapses to classical state with probability $|\alpha_i|^2$
- Challenge: Exponential state space makes simulation intractable

- Key Insight: Encode quantum computation as weighted counting problem
  - Boolean formula F encodes valid computation paths
  - Weight function W encodes quantum amplitudes

- $\Pr[\text{measuring outcome } x] = |W(F \wedge \text{outcome} = x)|^2$

Impact: State of the art approach for large classes of circuits, enabling verification of quantum algorithms on classical hardware

Applications across Computer Science



Impact: Counting-based approach is now the state of the art for all these applications

So Fundamental Yet So Hard

Valiant, 1979: Counting exactly is **#P-hard**

So Fundamental Yet So Hard

Valiant, 1979: Counting exactly is **#P-hard**

Stockmeyer, 1983: Probably Approximately Correct (PAC) aka (ε, δ)-guarantees

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

Stoc83, JVV86, BP94: Polynomial calls to SAT oracle suffice

So Fundamental Yet So Hard

Valiant, 1979: Counting exactly is **#P-hard**

Stockmeyer, 1983: Probably Approximately Correct (PAC) aka (ε, δ)-guarantees

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

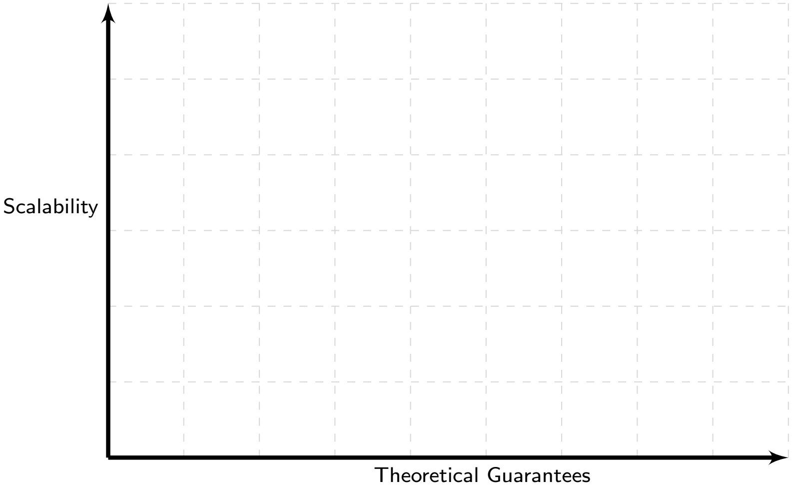
Stoc83, JVV86, BP94: Polynomial calls to SAT oracle suffice

- Not practical

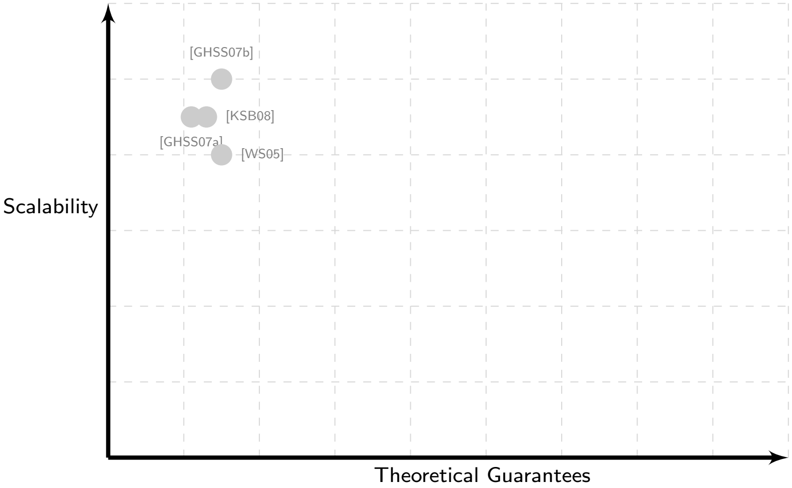
SAT Solver $\neq$ SAT Oracle

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

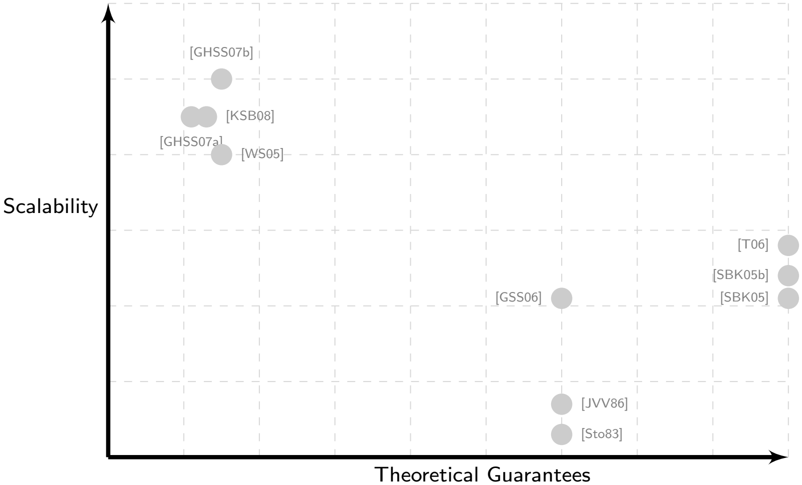
Snapshot from 2012



Snapshot from 2012

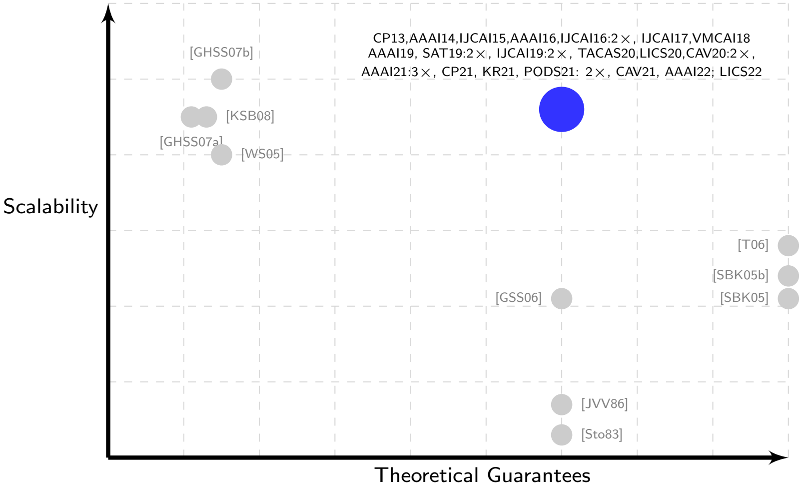


Snapshot from 2012



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 Singapore

How many people in Singapore like coffee?

- Population of Singapore $= 5.6$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx 5.6$M)

Counting in Singapore

How many people in Singapore like coffee?

- Population of Singapore $= 5.6$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx 5.6$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 5.6M/50

Counting in Singapore

How many people in Singapore like coffee?

- Population of Singapore = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx 5.6$M)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 5.6M/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 Singapore

How many people in Singapore like coffee?

- Population of Singapore = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx$ 5.6M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 5.6M/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

Counting in Singapore

How many people in Singapore like coffee?

- Population of Singapore = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx 5.6$M)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 5.6M/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 Singapore

How many people in Singapore like coffee?

- Population of Singapore = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx 5.6$M)

- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 5.6M/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

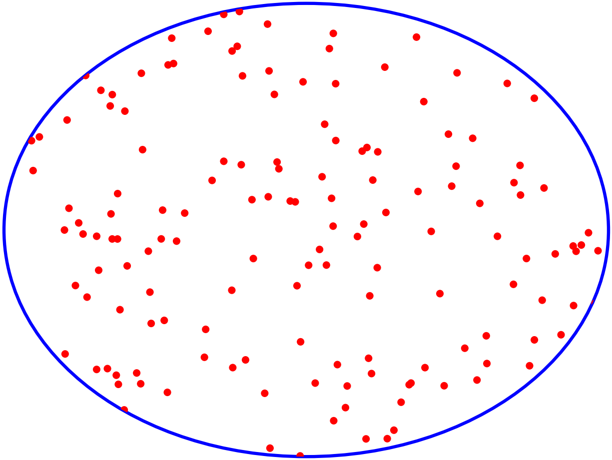
Counting in Singapore

How many people in Singapore like coffee?

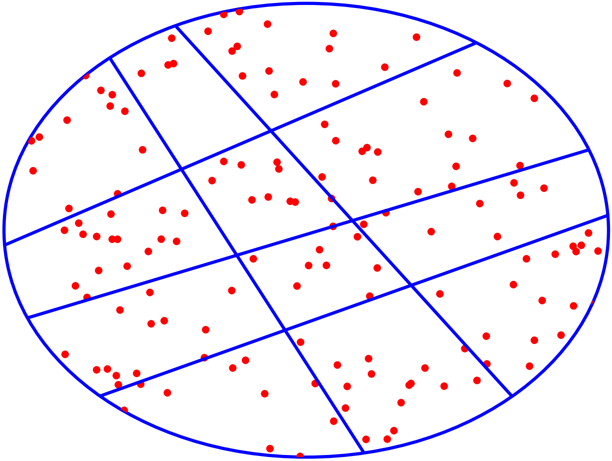
- Population of Singapore = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n \approx 5.6$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 5.6M/50
    - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in 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 – $\mathcal{O}(n)$ or $\mathcal{O}(\log n)$?

As Simple as Counting Dots

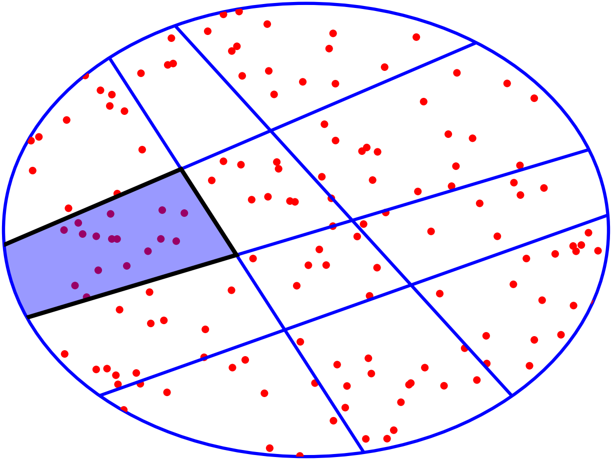


As Simple as Counting Dots



As Simple as Counting Dots

Pick a random cell



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

Challenges

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

Challenge 2 How many cells?

Challenges

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

- Designing function h : assignments $\rightarrow$ cells (hashing)

Challenges

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

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

$\boxed{\text{2-wise Independent Hashing}}$ [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

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

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

Challenges

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

$\boxed{\text{Random XOR-based Hash Functions}}$ [CW77]

Challenge 2 How many cells?

Challenge 2: How many cells?

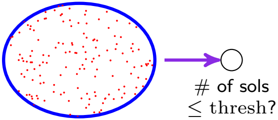
[CMV13,CMV16]

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

Challenge 2: How many cells?

[CMV13,CMV16]

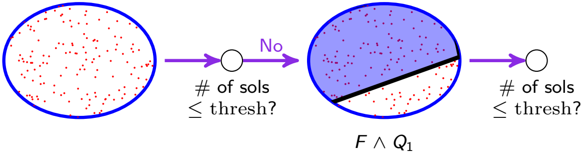
- A cell is small if it has $\approx \mathrm{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

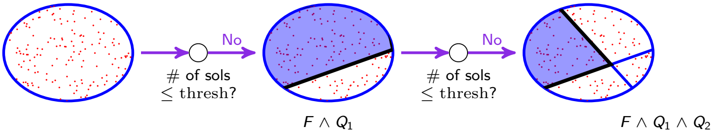
[CMV13,CMV16]

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



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

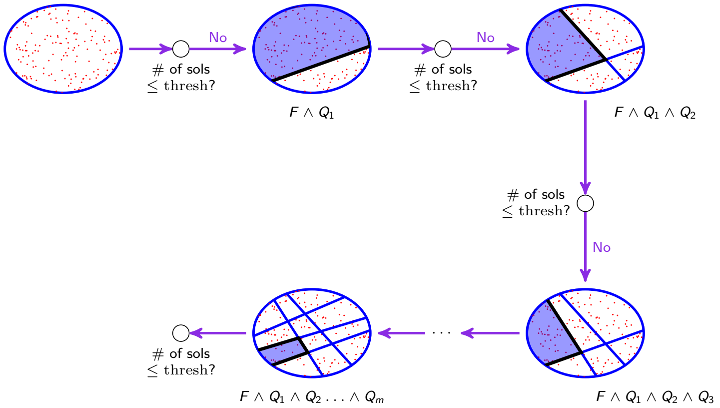
- A cell is small if it has $\approx \mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

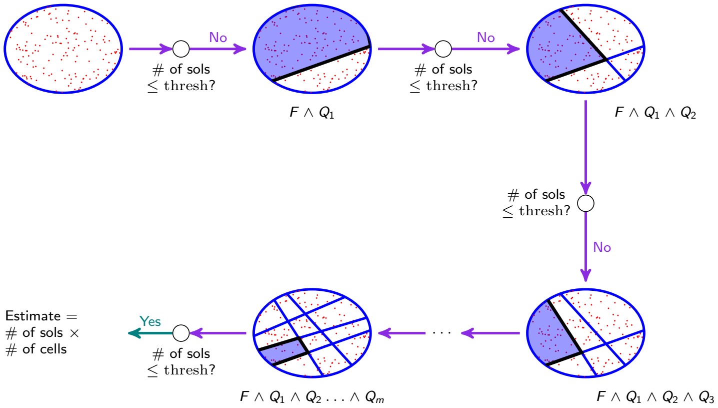
- A cell is small if it has $\approx \mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

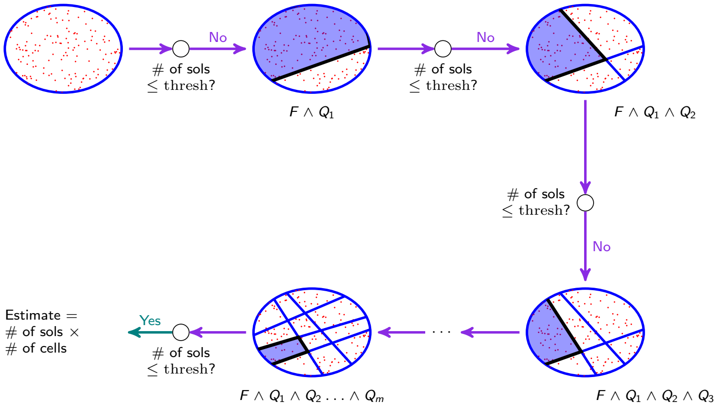
- A cell is small if it has $\approx \mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

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

ApproxMC makes $O(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta} \cdot \log n)$ SAT queries.

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

ApproxMC: Early Years (2013-16)

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

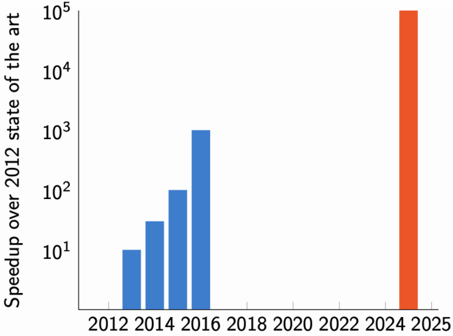
2019: CP-13 paper selected as one of the 25 papers across 25 years of CP conference

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

Closing Slide from Seminar at NUS in Feb 2017

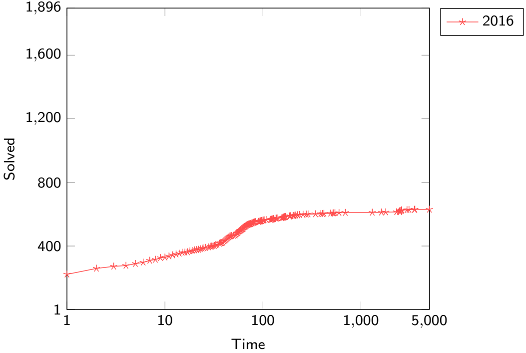
Mission 2025: Constrained Counting and Sampling Revolution



Requires combinations of ideas from theory, statistics and systems

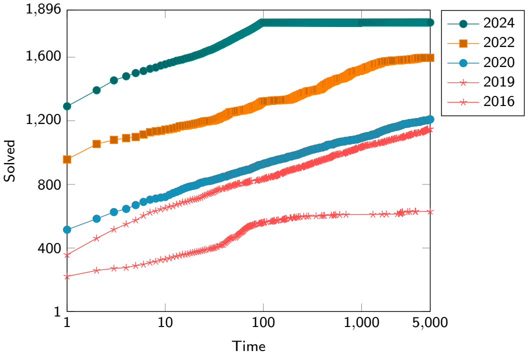
2025 Target: 100× speedup over 2016

ApproxMC: In Pursuit of Scalability



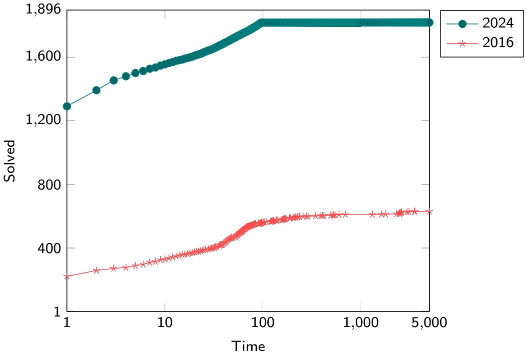
1896 instances from diverse applications
All experiments on the same hardware

ApproxMC: In Pursuit of Scalability



1896 instances from diverse applications
All experiments on the same hardware

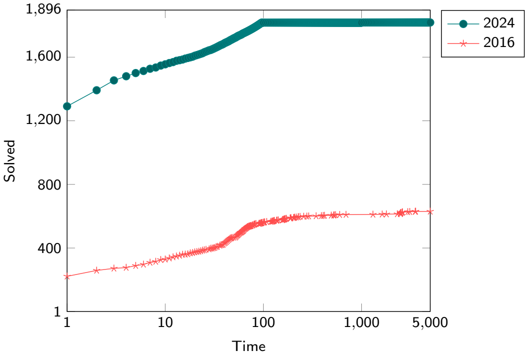
ApproxMC: In Pursuit of Scalability



1896 instances from diverse applications
All experiments on the same hardware

2016 630 instances, each in $\leq$ 5000 seconds
2024 1291 instances, each in $\leq$ 1 second

ApproxMC: In Pursuit of Scalability



1896 instances from diverse applications
All experiments on the same hardware

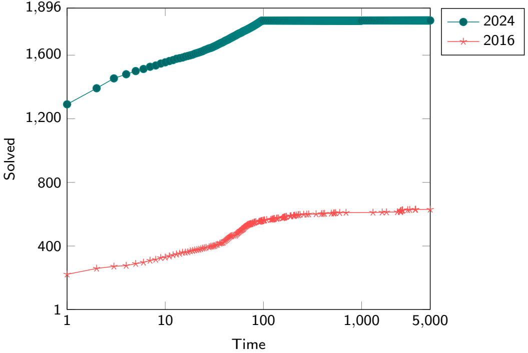
2016 630 instances, each in $\leq$ 5000 seconds
2024 1291 instances, each in $\leq$ 1 second

Time taken (seconds) for an instance

2016: 3552.16 2019: 32.83 2020: 19.59 2022: 0.15 2024: 0.02

A speedup of 100,000$\times$ over 2016

ApproxMC: In Pursuit of Scalability



1896 instances from diverse applications
All experiments on the same hardware

2016 630 instances, each in $\leq$ 5000 seconds
2024 1291 instances, each in $\leq$ 1 second

Time taken (seconds) for an instance

2016: 3552.16 2019: 32.83 2020: 19.59 2022: 0.15 2024: 0.02

A speedup of 100,000$\times$ over 2016

Still provides (ε, δ)-guarantees

In Pursuit of Scalability

Theoretical Advances	Sparse hashing SAT-20 LICS-20	Duality CP-19	DNF CP-18,IJCAI-19 PODS-21,22	Phase Transition IJCAI-16,17,19 CP-20
Algorithmic Engineering	Indep Supp Constraints-16 ICCAD-22, DAC-24	Chain Formulas IJCAI-15 NeurIPS-20, SAT-25	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23, CAV-25
Software Development	CNF-XOR AAAI-19 CAV-20, CAV-24	Pseudo-Boolean CP-21 AAAI-24, AAAI-25 SAT-25	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21

In Pursuit of Scalability

Theoretical Advances	Sparse hashing SAT-20 LICS-20	Duality CP-19	DNF CP-18,IJCAI-19 PODS-21,22	Phase Transition IJCAI-16,17,19 CP-20
Algorithmic Engineering	Indep Supp Constraints-16 ICCAD-22, DAC-24	Chain Formulas IJCAI-15 NeurIPS-20, SAT-25	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23, CAV-25
Software Development	CNF-XOR AAAI-19 CAV-20	Pseudo-Boolean CP-21 AAAI-24, AAAI-25 SAT-25	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21

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: $F \wedge Q_1 \cdots \wedge Q_m$

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: $F \wedge Q_1 \cdots \wedge Q_m$

Need to handle $\underbrace{F}_{\text{CNF}} \wedge \underbrace{Q_1 \cdots \wedge Q_m}_{\text{XOR}}$

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: $F \wedge Q_1 \cdots \wedge Q_m$

Need to handle $\underbrace{F}_{\text{CNF}} \wedge \underbrace{Q_1 \cdots \wedge Q_m}_{\text{XOR}}$

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

SAT Solvers != 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

level	dec		prop
0	x_1		
1	x_3	$\rightarrow$	x_5
2	x_4	$\rightarrow$	$x_2, \neg x_5$

CDCL

$$\left.\begin{array}{ccc|c} x_1 & x_2 & x_3 & \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{array}\right.$$

Gauss-Jordan Elimination

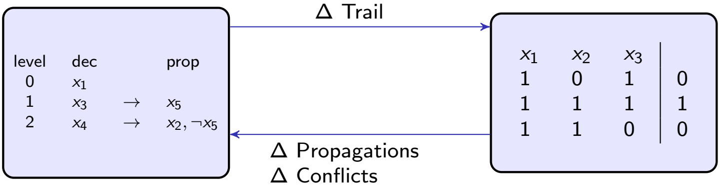
New Architecture for CNF-XOR Formulas

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



Incremental CDCL

Incremental Gauss-Jordan Elimination

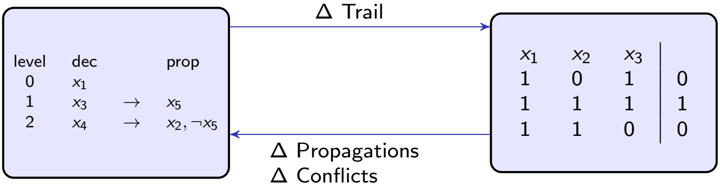
New Architecture for CNF-XOR Formulas

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



Incremental CDCL — Incremental Gauss-Jordan Elimination

Engineering an efficient CDCL-GJE solver [SM19; SGM20]

- 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: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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

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: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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: $\frac{n}{2}$

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: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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 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
  - $F := X_3 \iff (X_1 \vee X_2)$
  - 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

Not All Variables Matter Equally

- 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)
- $\mathcal{I} \subseteq X$ 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 $\mathcal{I}$ [CMV14]
- Typically $\mathcal{I}$ is 1-2 orders of magnitude smaller than X

Not All Variables Matter Equally

- 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)
- $\mathcal{I} \subseteq X$ 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 $\mathcal{I}$ [CMV14]
- Typically $\mathcal{I}$ is 1-2 orders of magnitude smaller than X

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

Not All Variables Matter Equally

- 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)
- $\mathcal{I} \subseteq X$ 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 $\mathcal{I}$ [CMV14]
- 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 [IMMV15]
  Best Student Paper, CP15

Not All Variables Matter Equally

- 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)
- $\mathcal{I} \subseteq X$ 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 $\mathcal{I}$ [CMV14]
- 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

Not All Variables Matter Equally

- 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)
- $\mathcal{I} \subseteq X$ 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 $\mathcal{I}$ [CMV14]
- 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; SM24]

Approach II + ApproxMC is up to 100× 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: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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 $X_i \in \mathcal{I}$

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: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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** $\mathbf{\frac{1}{2}}$?

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: $Q_1, Q_2, \dots Q_m$
- Solutions in a cell: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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** $\mathbf{\frac{1}{2}}$?

- 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 $\implies \frac{\mathsf{Var}[Z_m]}{\mathsf{E}[Z_m]} \leq 1 \implies$ Concentration bounds

Beyond 2-wise Independence

[MA20]

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

Beyond 2-wise Independence

[MA20]

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* $\frac{\mathsf{Var}[Z_m]}{\mathsf{E}[Z_m]} \leq 1.1$

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

Beyond 2-wise Independence

[MA20]

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 $\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1.1$

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

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

Beyond 2-wise Independence

[MA20]

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 $\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1.1$

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

$$\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1 + |\mathrm{Sol}(F)|^{-1} \cdot \sum_{\sigma_1 \in \mathrm{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathrm{Sol}(F) \\ w = d(\sigma_1, \sigma_2)}} \overbrace{r(w, p_m)}^{\approx \text{ collision probability}} \quad \underbrace{\leq 1 + \sum_{w=0}^{n} \binom{n}{w} r(w, p_m)}_{\text{Earlier Attempts}}$$

[EGSS14,ZCSE16,AD17,ATD18]

Beyond 2-wise Independence

[MA20]

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 $\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1.1$

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

$$\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1 + |\mathrm{Sol}(F)|^{-1} \cdot \sum_{\sigma_1 \in \mathrm{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathrm{Sol}(F) \\ w = d(\sigma_1,\sigma_2)}} \overbrace{r(w, p_m)}^{\approx \text{ collision probability}} \qquad \underbrace{\leq 1 + \sum_{w=0}^{n} \binom{n}{w} r(w, p_m)}_{\text{Earlier Attempts}}$$

[EGSS14,ZCSE16,AD17,ATD18]

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

$$C_F(w) = \left| \{\sigma_1, \sigma_2 \in \mathrm{Sol}(F) \mid d(\sigma_1, \sigma_2) = w\} \right|$$

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

Barrier: But $|\mathrm{Sol}(F)|$ can be arbitrarily large

Beyond 2-wise Independence

[MA20]

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 $\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1.1$

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

$$\frac{\mathrm{Var}[Z_m]}{\mathrm{E}[Z_m]} \leq 1 + |\mathrm{Sol}(F)|^{-1} \cdot \sum_{\sigma_1 \in \mathrm{Sol}(F)} \sum_{\substack{\sigma_2 \in \mathrm{Sol}(F) \\ w = d(\sigma_1, \sigma_2)}} \overbrace{r(w, p_m)}^{\approx \text{ collision probability}} \quad \underbrace{\leq 1 + \sum_{w=0}^{n} \binom{n}{w} r(w, p_m)}_{\text{Earlier Attempts}}$$

[EGSS14,ZCSE16,AD17,ATD18]

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

$$C_F(w) = \left| \{\sigma_1, \sigma_2 \in \mathrm{Sol}(F) \mid d(\sigma_1, \sigma_2) = w\} \right|$$

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

Barrier: But $|\mathrm{Sol}(F)|$ can be arbitrarily large

Key Idea: In the context of Z_m, It suffices to assume $|\mathrm{Sol}(F)| < 2^{m+u}$ 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: $F \wedge Q_1 \cdots \wedge Q_m$

✓**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 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** $\mathbf{\frac{1}{2}}$?

Theoretical Advances Pick m-th XOR with $p_m = \frac{\log m}{m}$

In the Pursuit of Scalability

Theoretical Advances	Sparse hashing SAT-20 LICS-20	Duality CP-19	DNF CP-18,IJCAI-19 PODS-21,22	Phase Transition IJCAI-16,17,19 CP-20
Algorithmic Engineering	Indep Supp Constraints-16 ICCAD-22, DAC-24	Chain Formulas IJCAI-15 NeurIPS-20, SAT-25	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23, CAV-25
Software Development	CNF-XOR AAAI-19 CAV-20, CAV-24	Pseudo-Boolean CP-21 AAAI-24, AAAI-25 SAT-25	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21

Reliability of Critical Infrastructure Networks

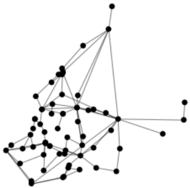
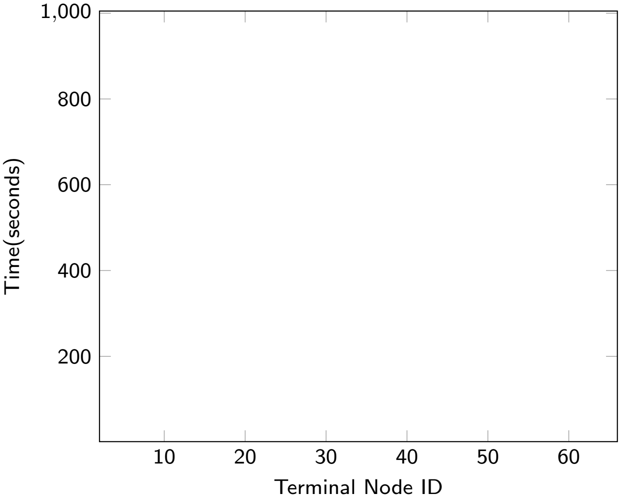


Figure: Plantersville, SC



Timeout = 1000 seconds

Reliability of Critical Infrastructure Networks

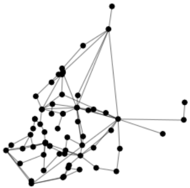
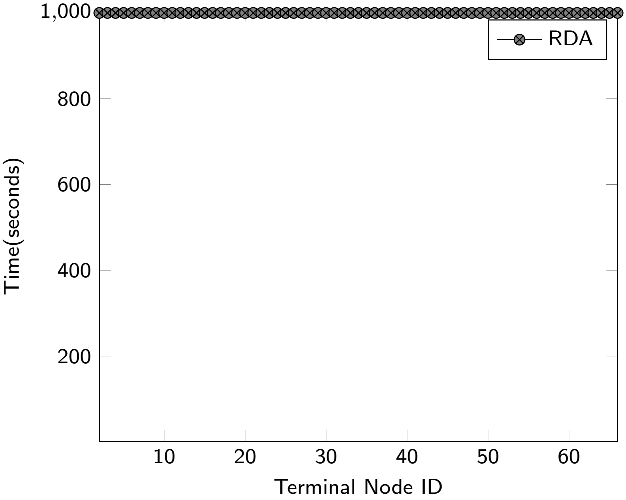


Figure: Plantersville, SC



Timeout = 1000 seconds

Reliability of Critical Infrastructure Networks

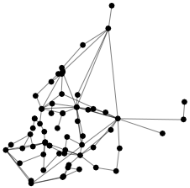
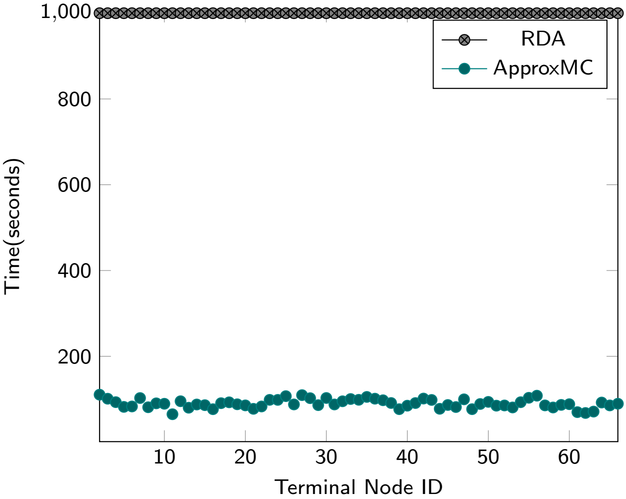


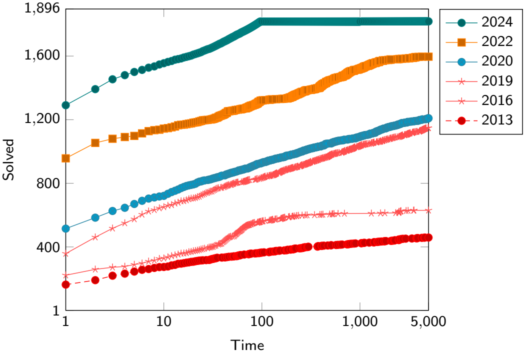
Figure: Plantersville, SC



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



1896 benchmarks from diverse applications

Time taken (seconds) for an instance

2016: 3552.16 2019: 32.83 2020: 19.59 2022: 0.15 2024: 0.02

A speedup of 100,000× over 2016

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.

SharpTNI: Counting and Sampling Parsimonious Transmission Networks under a Weak Bottleneck

Palash Sashittal[1] and Mohammed El-Kebir[2*]

Check before You Change: Preventing Correlated Failures in Service Updates

Ennan Zhai[†], Ang Chen[‡], Ruzica Piskac[○], Mahesh Balakrishnan[§,*]
Bingchuan Tian[♮], Bo Song[•], Haoliang Zhang[•]

Automating the Development of Chosen Ciphertext Attacks

Gabrielle Beck, Maximilian Zinkus, and Matthew Green,
Johns Hopkins University

Static Evaluation of Noninterference using Approximate Model Counting

Ziqiao Zhou, Zhiyun Qian, Michael K. Reiter, Yinqian Zhang

A Study of the Learnability of Relational Properties

Model Counting Meets Machine Learning (MCML)

Muhammad Usman, University of Texas at Austin, USA, muhammadusman@utexas.edu
Wenxi Wang, University of Texas at Austin, USA, wenxiw@utexas.edu
Marko Vasic, University of Texas at Austin, USA, vasic@utexas.edu
Kaiyuan Wang[*], Haris Vikalo, Sarfraz Khurshid

Quantifying Software Reliability via Model-Counting

Samuel Teuber[(✉)] and Alexander Weigl

IN SEARCH FOR A SAT-FRIENDLY BINARIZED NEURAL NETWORK ARCHITECTURE

Nina Narodytska, **Hongce Zhang[*]**

Quantifying the Efficacy of Logic Locking Methods

Joseph Sweeney, Deepali Garg, Lawrence Pileggi

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.

SharpTNI: Counting and Sampling Parsimonious Transmission Networks under a Weak Bottleneck

Palash Sashittal[1] and Mohammed El-Kebir[2*]

Check before You Change: Preventing Correlated Failures in Service Updates

Ennan Zhai[†], Ang Chen[‡], Ruzica Piskac[○], Mahesh Balakrishnan[§,*]
Bingchuan Tian[♮], Bo Song[•], Haoliang Zhang[•]

Automating the Development of Chosen Ciphertext Attacks

Gabrielle Beck, Maximilian Zinkus, and Matthew Green,
Johns Hopkins University

Model Counting Competition

MC-2024

Part of SAT 2024

- Tracks: **Model Counting**, **Weighted Model Counting**, **Weighted Model Counting (negative weights)**, **Projected Model Counting**, **Projected Weighted Model Counting**
(description, StarExec Community, format, winners, report [tba], slides, solvers, competition instances, submitted benchmarks

MC-2023

Part of SAT 2023

- Tracks: **Model Counting**, **Weighted Model Counting**, **Projected Model Counting**, **Projected Weighted Model Counting**
(description, StarExec Community, format, winners, report, slides, solvers, competition instances, submitted benchmarks)

MC-2022

Part of FLoC Olympic Games

- Tracks: **Model Counting**, **Weighted Model Counting**, **Projected Model Counting**, **Projected Weighted Model Counting**
(description, StarExec Community, format, winners, report, slides, solvers, competition instances, submitted benchmarks)

MC-2021

- Tracks: **Model Counting**, **Weighted Model Counting**, and **Projected Model Counting**
(description, StarExec Community, format, winners, report, slides, solvers, competition instances, submitted benchmarks)

MC-2020

The results of each Model Counting Competition are documented in a report.

- Tracks: **Model Counting**, **Weighted Model Counting**, and **Projected Model Counting**
(description, format, winners, report, slides, solvers, competition instances, submitted benchmarks)

Workshops

- 2024 Workshop on Counting, Sampling, and Synthesis (in person event)
(program)
- 2023 Workshop on Counting and Sampling (in person event)
(program)
- 2022 Workshop on Counting and Sampling (in person event)
(program, abstracts, slides)
- Workshop 2021
(program, abstracts, slides , recordings)
- 2020 Model Counting Workshop
(program, abstracts, slides , recordings)

Static Evaluation of Noninterference using Approximate Model Counting

Ziqiao Zhou Zhiyun Qian Michael K. Reiter Yinqian Zhang

A Study of the Learnability of Relational Properties

Model Counting Meets Machine Learning (MCML)

Muhammad Usman, University of Texas at Austin, USA, muhammadusman@utexas.edu
Wenxi Wang, University of Texas at Austin, USA, wenxiw@utexas.edu
Marko Vasic, University of Texas at Austin, USA, vasic@utexas.edu
Kaiyuan Wang*
Haris Vikalo
Sarfraz Khurshid

Quantifying Software Reliability via Model-Counting

Samuel Teuber[(✉)] and Alexander Weigl

IN SEARCH FOR A SAT-FRIENDLY BINARIZED NEURAL NETWORK ARCHITECTURE

Nina Narodytska Hongce Zhang*

Quantifying the Efficacy of Logic Locking Methods

Joseph Sweeney, Deepali Garg, Lawrence Pileggi

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 [**M**SV19]
- Praise by Donald Knuth and coverage in popular media (Quanta Magazine) [CV**M**22]

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 [**M**SV19]
- Praise by Donald Knuth and coverage in popular media (Quanta Magazine) [CV**M**22]

Streaming Counting over a stream: Distinct Elements
Example: How many unique customers visit website?
Fundamental problem in Databases

- CACM 2023 Research Highlights [PVB**M**21]
- ACM SIGMOD 2022 Research Highlight

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 [**M**SV19]
- Praise by Donald Knuth and coverage in popular media (Quanta Magazine) [CV**M**22]

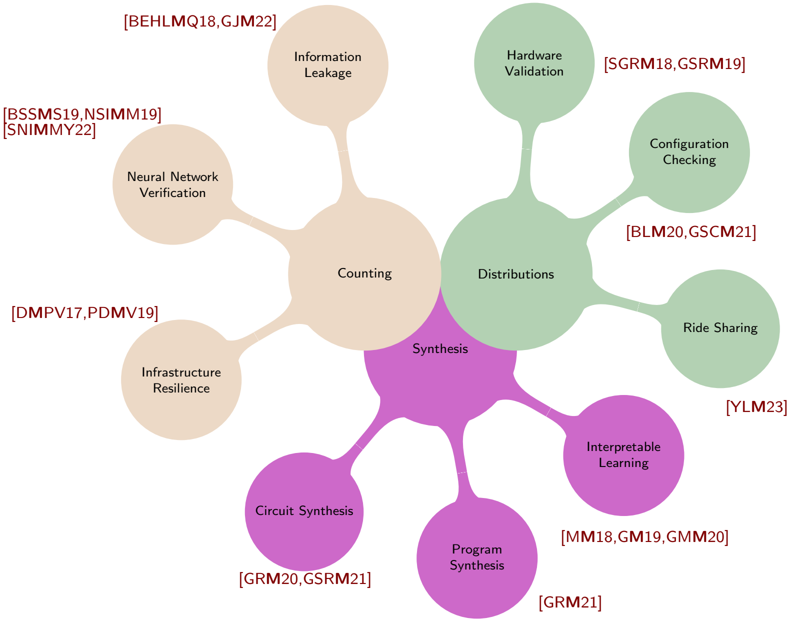
Streaming Counting over a stream: Distinct Elements
Example: How many unique customers visit website?
Fundamental problem in Databases

- CACM 2023 Research Highlights [PVB**M**21]
- ACM SIGMOD 2022 Research Highlight

Unsatisfiable Subsets Count minimal subsets of clauses that are unsatisfiable.
Diagnosis metric for systems

- ICLP-24 Best Paper Award [K**M**24]

Counting, Sampling, and Synthesis



In Pursuit of Scalability

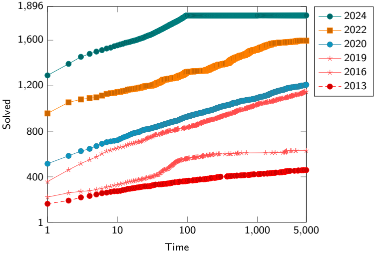


Figure: Counting over the years

In Pursuit of Scalability

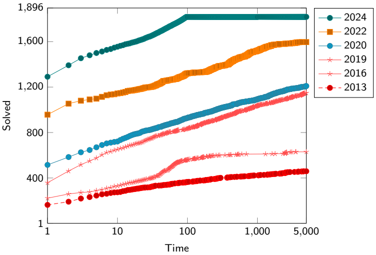
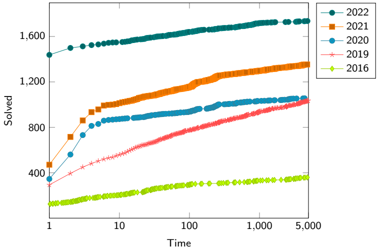
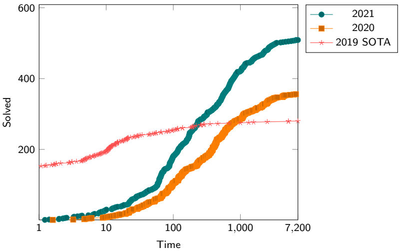


Figure: Counting over the years



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

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**.

Today's Quantitative Automated Reasoning tools

$$\approx$$

Qualitative Reasoning tools (SAT/SMT) in early 2000s

Where do we go from here?

The Quest for Scalability is Endless

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**.

Today's Quantitative Automated Reasoning tools

$$\approx$$

Qualitative Reasoning tools (SAT/SMT) in early 2000s

The Time is Now

- LLMs enable rapid system creation $\rightarrow$ Critical need for verification
- The achilees heel of formal methods adoption: formalization of complex systems
- LLMs excel at enabling rapid formalization of complex systems

This is the perfect time to do automated reasoning!

Industrial Practice: 100$\times$ Speedup

The Pursuit of Scalablity

Mission 2030: 100$\times$ Speedup for Quantitative Reasoning Tasks

Challenge Problems (for Counting)

Civil Engineering Rigorous resilience estimation for power grid of Los Angeles

AI Safety Quantized neural network with 1M neurons

Quantum Simulation Quantum Simultion for 1000 qubits.

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 Scalablity

Mission 2030: 100× Speedup for Quantitative Reasoning Tasks

Challenge Problems (for Counting)

Civil Engineering Rigorous resilience estimation for power grid of Los Angeles

AI Safety Quantized neural network with 1M neurons

Quantum Simulation Quantum Simultion for 1000 qubits.

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
- Preliminary Work: AAAI-19, NeurIPS-20, NeurIPS-21, CP-22, NeurIPS-22, ICML-24, ICLR-25

It Takes a Village

Research Group

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

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)
Moshe Y. Vardi(Rice,US)
Wenxi Wang(UTAustin,US)
Roland H. C. Yap(NUS, SG)

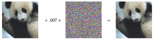
Alyas Almaawi(UTAustin,US)
Debabrota Basu(Chalmers,US)
Kian Ming Adam Chai(DSO, SG)
Supratik Chakraborty (IITB, IN)
Tiago Cogumbreiro(Rice,US)
Jeffrey M. Dudek(Rice,US)
Daniel J. Fremont(UCB,US)
Stephan Gocht (Lund U., SE)
Alexander Ivrii(IBM, IL)
Sarfraz Khurshid(UTAustin,US)
Massimo Lupascu(NUS, SG)
Dmitry Malioutov(IBM,US)
Rakesh Mistry(IITB, IN)
Nina Narodytska(VMware,US)
Aduri Pavan(ISU,US)
Jean Quilbeuf(Inria, FR)
Vivek Sarkar(Rice,US)
Sanjit A. Seshia(UCB,US)
Aditya A. Shrotri(Rice,US)
Yong Kiam Tan (NTU, SG)
N. V. Vinodchandran(UNL,US)
Yaqi Xie(NUS, SG)

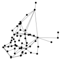
Eduard Baranov(UCLouvain, BE)
Arnab Bhattacharya (NUS, SG)
Diptarka Chakraborty(NUS,SG)
Davin Choo(DSO, SG)
Vincent Derkinderen(KUL, BE)
Leonardo Duenas-Osorio (Rice,US)
Dror Fried (Open U., IL)
Annelie Heuser(CNRS, FR)
Saurabh Joshi(IITH, IN)
Raghav Kulkarni(CMI, IN)
Deepak Majeti(Rice,US)
Joao Marques-Silva(ANITI, FR)
M.Mohammadalitajrishi(Polymtl,CA)
Roger Paredes(Rice,US)
Gilles Pesant(Polymtl,CA)
Subhajit Roy (IITK, IN)
Prateek Saxena(NUS, SG)
Shiqi Shen(NUS, SG)
Friedrich Slivovsky(TU Wien, AT)
Muhammad Usman(UTAustin,US)
Kaiyuan Wang(Google,US)
Ziwei Xu(NUS, SG)

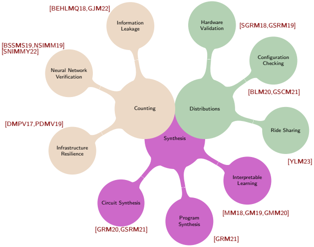
Funding Agencies: National Research Foundation, Ministry of Education, Defense Service Organization, NSERC
Industrial Support: Grab Taxiholdings, Amazon, Microsoft Research

Quantitative Automated Reasoning

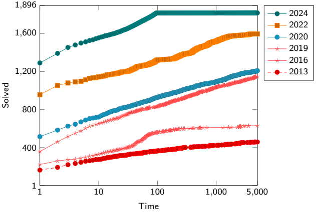
```
PC2 (char[] SP, char[] UI) {
        match = true;
        for (int i=0; i<UI.length(); i++) {
                if (SP[i] != UI[i]) match=false;
                else match = match;
        }
        if match return Yes;
        else return No;
}
```







Theoretical Advances	Sparse hashing SAT-20 LICS-20	Duality CP-19	DNF CP-18,IJCAI-19 PODS-21,22	Phase Transition IJCAI-16,17,19 CP-20
Algorithmic Engineering	Indep Supp Constraints-16 ICCAD-22, DAC-24	Chain Formulas IJCAI-15 NeurIPS-20, SAT-25	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23, CAV-25
Software Development	CNF-XOR AAAI-19 CAV-20, CAV-24	Pseudo-Boolean CP-21 AAAI-24, AAAI-25 SAT-25	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21



These slides are available at https://www.cs.toronto.edu/~meel/talks.html

Detailed Future Directions

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

Theoretical Advances

Formula-based Sparse-XORs DNF, Minimal Solutions, Chain formula
Revisiting FPRAS Permanent, Automata, Linear Extensions
Parameterized Complexity Addition of XORs
Streaming Delphic Sets
Synthesis A theory of learning from relations
Entropy Reduction in the number of queries

Algorithmic Engineering

Incremental Incremental Counting Queries
Bit-vectors Partitioning; Independent Support
Heuristic ML-guided heuristic synthesis
Distributed Streaming techniques
SMT Synthesis SMT Formula Learning
Beyond Qualititative Synthesis Optimal Functions, Approximate Synthesis

Software Development

Tighter Integration Multiple Queries
Hybrid Constraints Callbacks
XOR Handling PB-XOR, BNN-XOR, MaxSAT-XOR, ASP-XOR
Accelerators GPU
Knowledge Compilation SMT, Portfolio

Certification

Distribution Probabilistic Programming Equivalence
Counting Certificate for Approximation