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

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

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

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

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

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

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

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")

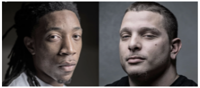
Knuth, 2016: These so-called "SAT solvers" can now routinely find solutions to practical problems that involve millions of variables and were thought until very recently to be hopelessly difficult.

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

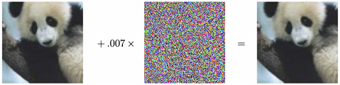
Beyond SAT I: Quantification

```
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;
}
```

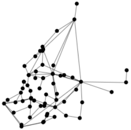
Information Leakage



Fairness



Robustness

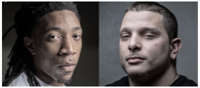


Critical Infrastructure

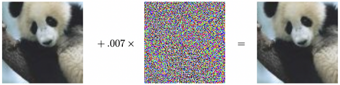
Beyond SAT I: Quantification

```
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;
}
```

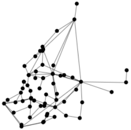
Information Leakage



Fairness



Robustness



Critical Infrastructure

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

Counting

Beyond SAT II: Sampling

Constrained-Random Verification

Configuration Testing

- System is simulated with test vectors
- Constraints represent *relevant* verification scenarios
- **Test vectors**: random solutions of constraints

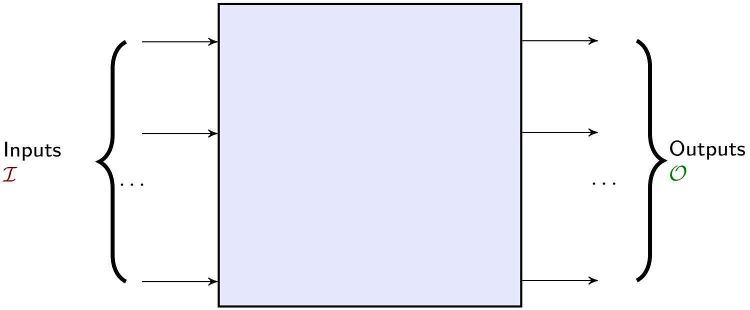
Sampling

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



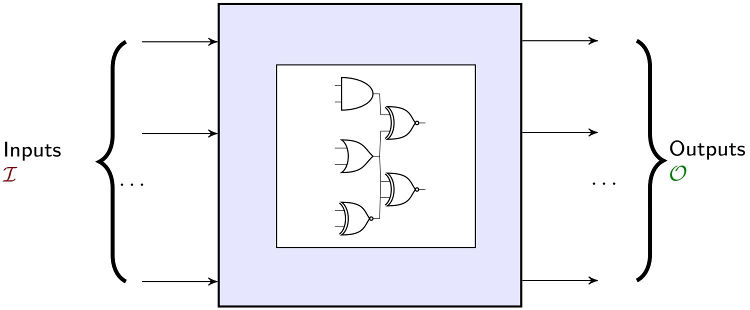
Synthesis

Beyond SAT III: Automated Synthesis

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

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

age	25
capital-gain	4000
occupation	coach



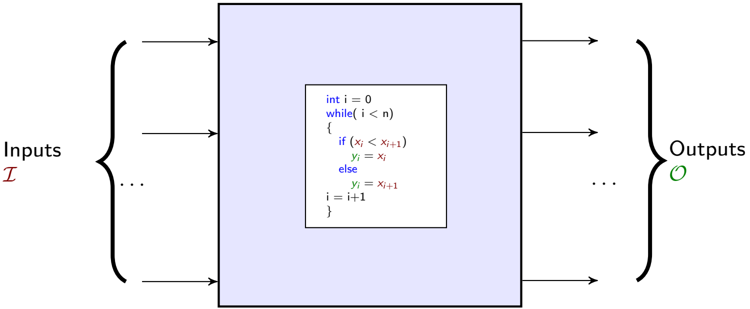
Synthesis

Beyond SAT III: Automated Synthesis

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

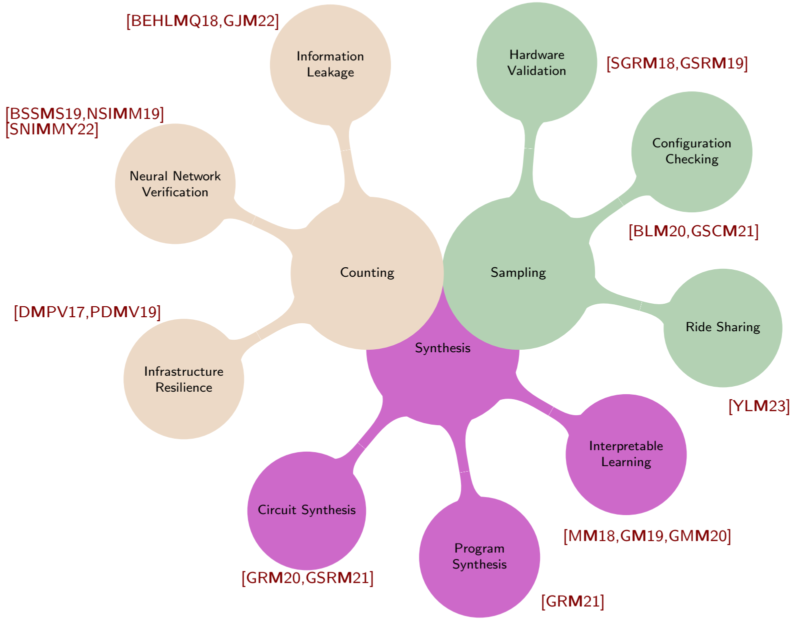
Specification: $\mathcal{P}(\mathcal{I}, \mathcal{O})$

age	25
capital-gain	4000
occupation	coach

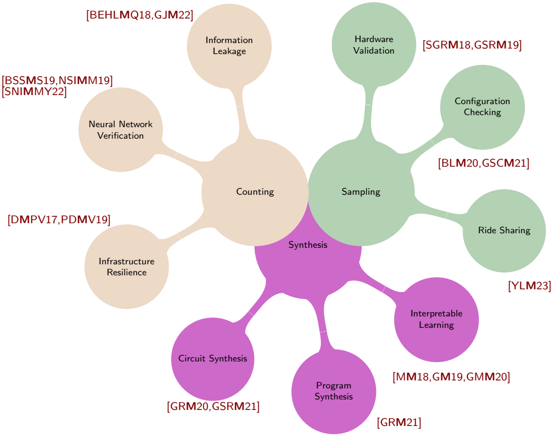


Synthesis

Research Overview



Research Overview



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

Counting

- Given: A Boolean formula F over $X_1, X_2, \cdots X_n$
- $\text{Sol}(F) = \{$ 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 Analysis of AI Systems

Main Course ApproxMC: A Scalable Counting Framework

Dessert Future Outlook

Resilience of Critical Infrastructure Networks

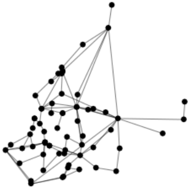
[DMPV17,PDMV19]

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

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19]

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

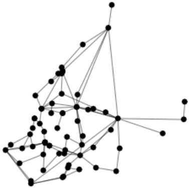


- $G = (V, E)$; set of source nodes S and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[t is disconnected from S]?

Resilience of Critical Infrastructure Networks

[DMPV17,PDMV19]

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 is disconnected from S]?

Constrained Counting

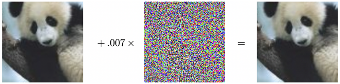
Key Idea: Encode disconnectedness using constraints

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]

Our Focus: Binarized Neural Networks



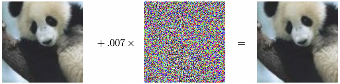
Robustness Quantification

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

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22]

Our Focus: Binarized Neural Networks



Robustness Quantification

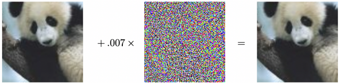
$$\underbrace{\left| \{x : \mathcal{N}(x + \varepsilon) \neq \mathcal{N}(x)\} \right|}_{\text{Encode Symbolically}}$$

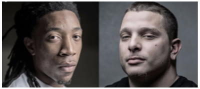
Constrained Counting

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22]

Our Focus: Binarized Neural Networks





Robustness Quantification

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

Fairness Quantification

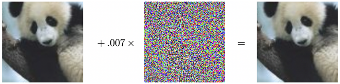
$$\underbrace{\left|\{x :\ \mathcal{N}(x \wedge \textsc{Black}) \neq \mathcal{N}(x \wedge \textsc{White})\}\right|}_{\text{Encode Symbolically}}$$

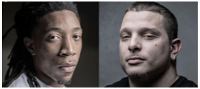
Constrained Counting

Quantitative Analysis of AI Systems

[BSSMS19,NSMIS19,NSMISV22]

Our Focus: Binarized Neural Networks





Robustness Quantification

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

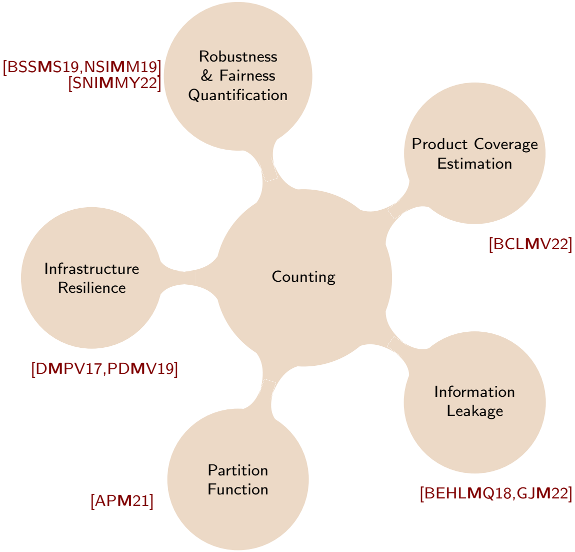
Fairness Quantification

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

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

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

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

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

So Fundamental Yet So Hard

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

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

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

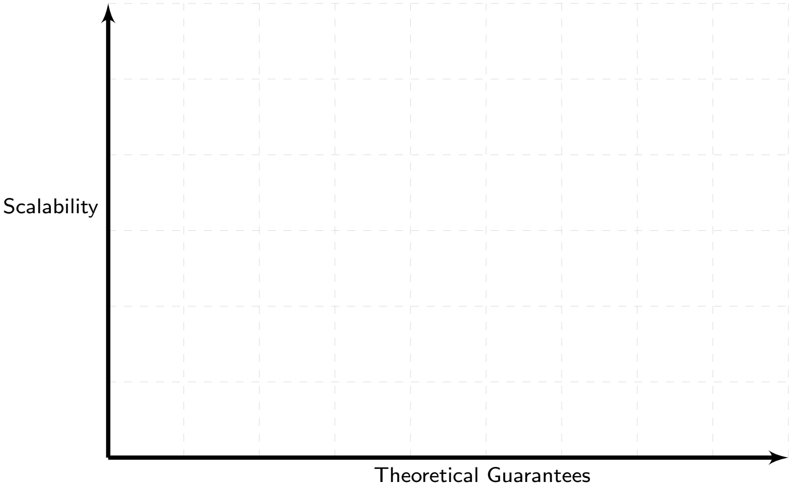
Stoc83, JVV86, BP94: Polynomial calls to SAT oracle suffice

- Not practical

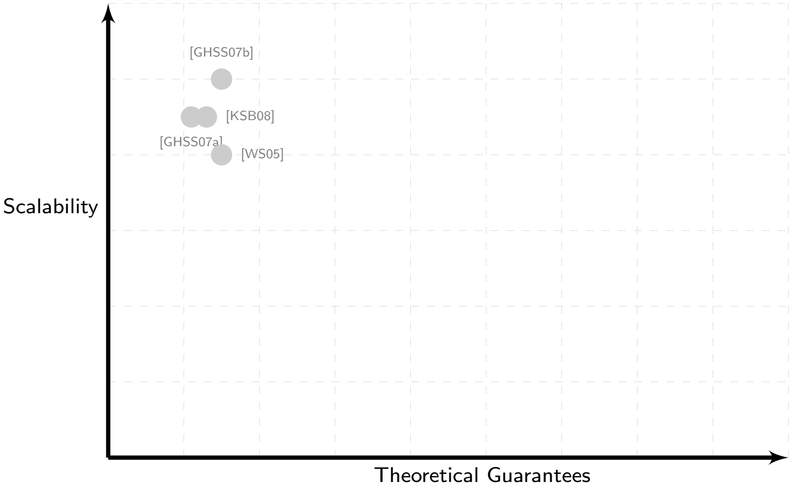
SAT Solver $\neq$ SAT Oracle

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

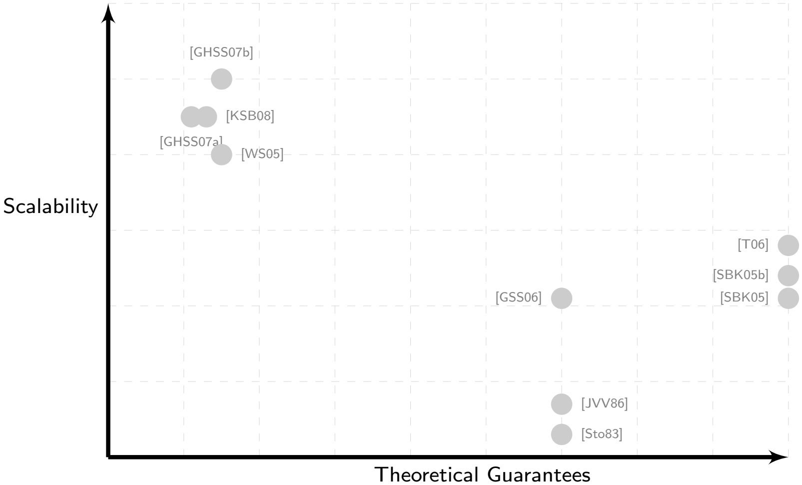
Snapshot from 2012



Snapshot from 2012

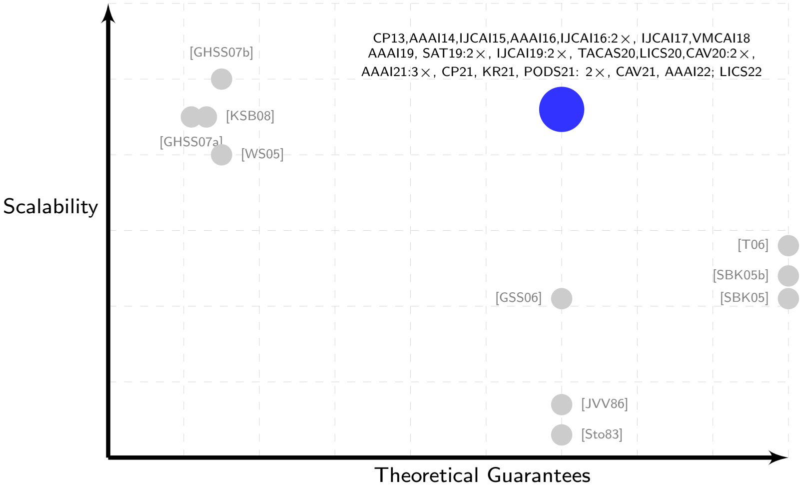


Snapshot from 2012



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 Atlanta

How many people in Atlanta like coffee?

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

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 ($2^n \approx 6.1$M)
- 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

Counting in Atlanta

How many people in Atlanta like coffee?

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

- 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 ($2^n \approx 6.1M$)

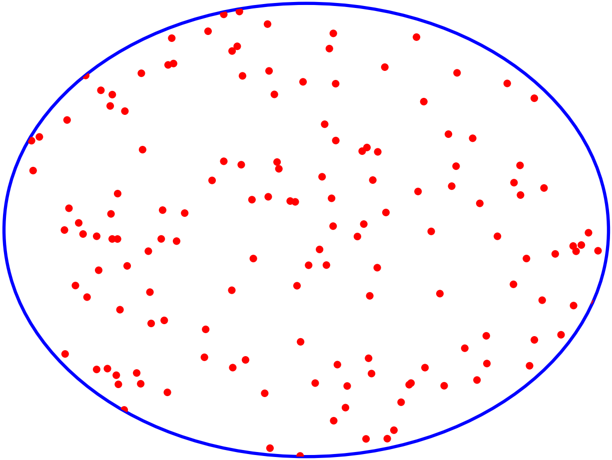
- Attempt #1: Pick 50 people and count how many of them like coffee and multiply by 6.1M/50
  - If only 1000 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

- 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

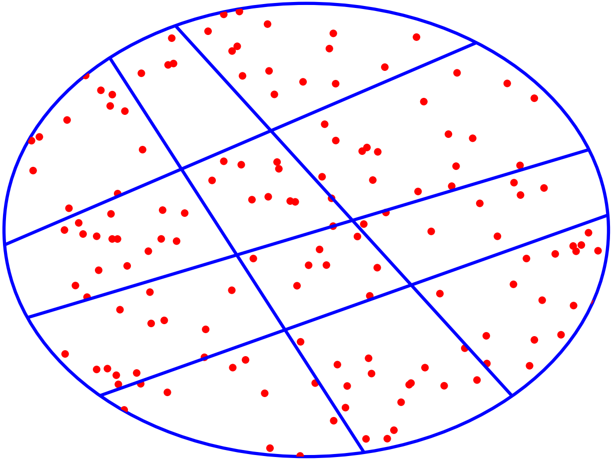
- Attempt #2: Enumerate every person who likes coffee
  - Potentially 2^n queries

Can we do with lesser # of SAT queries – $\mathcal{O}(n)$ or $\mathcal{O}(\log n)$?

As Simple as Counting Dots

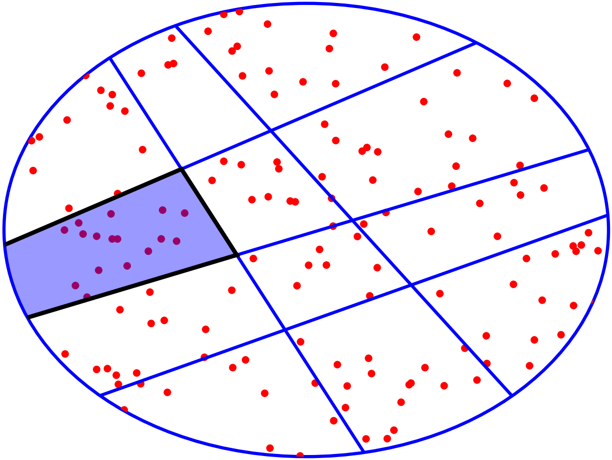


As Simple as Counting Dots



As Simple as Counting Dots

Pick a random cell



Estimate = Number of solutions in a cell × 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

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 $h : \{0,1\}^n \to \{0,1\}^m$, choose m random XORs
- Pick every X_i with prob. $\frac{1}{2}$ and XOR them
  - $X_1 \oplus X_3 \oplus X_6 \cdots \oplus X_{n-2}$
- 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?

Random XOR-based Hash Functions [CW77]

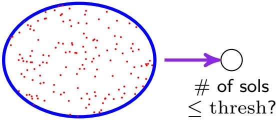
Challenge 2 How many cells?

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

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

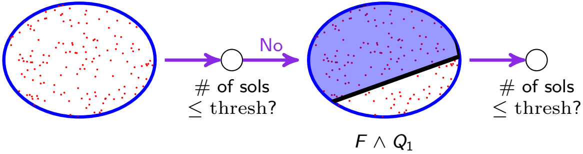
Challenge 2: How many cells? [CMV13,CMV16]

- 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]

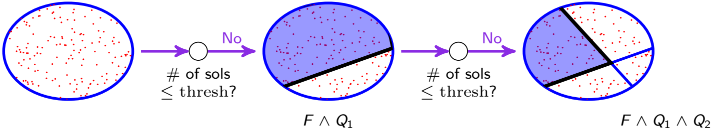
- A cell is small if it has $\approx \mathrm{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

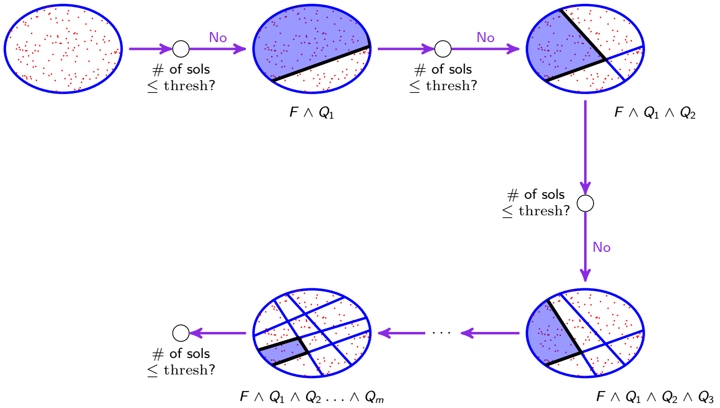
[CMV13,CMV16]

- 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]

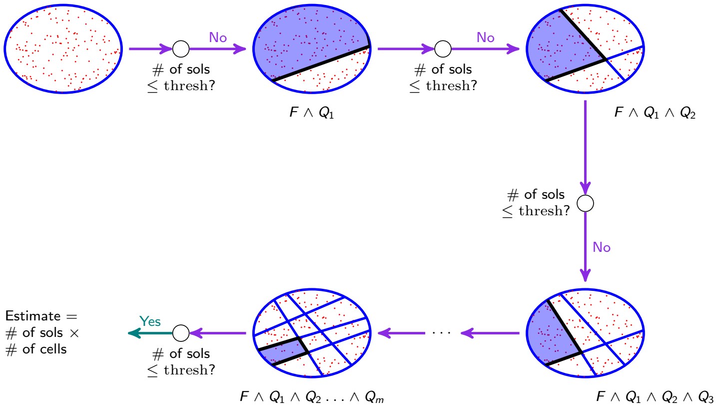
- A cell is small if it has $\approx \text{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



Challenge 2: How many cells?

[CMV13,CMV16]

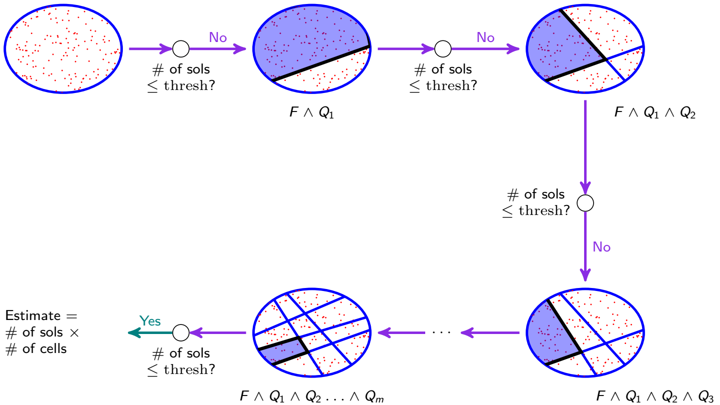
- A cell is small if it has $\approx \text{thresh} = 5(1+\frac{1}{\varepsilon})^2$ solutions
- Many solutions $\implies$ Many cells & Fewer solutions $\implies$ Fewer cells



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 $\Longrightarrow$ Many cells & Fewer solutions $\Longrightarrow$ 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: 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

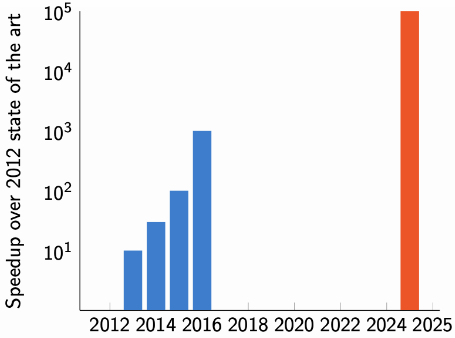
2019: CP-13 paper selected as one of the 25 papers across 25 years of CP conference

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

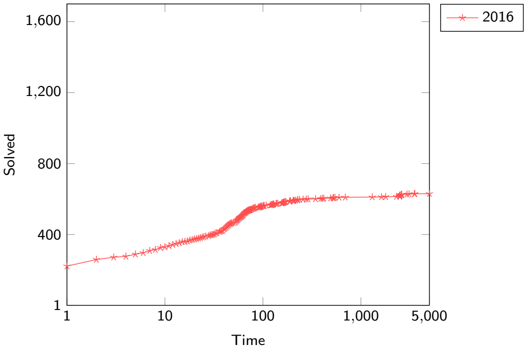
Mission 2025: Constrained Counting and Sampling Revolution



Requires combinations of ideas from theory, statistics and systems

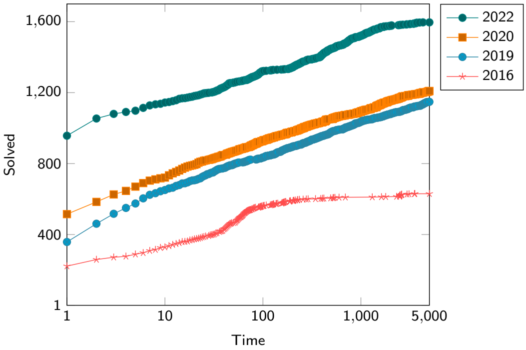
2025 Target: 100× speedup over 2016

ApproxMC: In Pursuit of Scalability



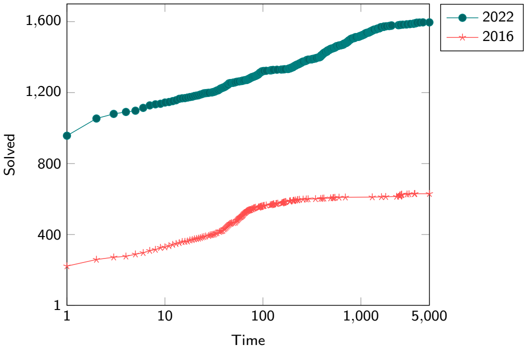
1896 instances from diverse applications
All experiments on 2022 hardware

ApproxMC: In Pursuit of Scalability



1896 instances from diverse applications
All experiments on 2022 hardware

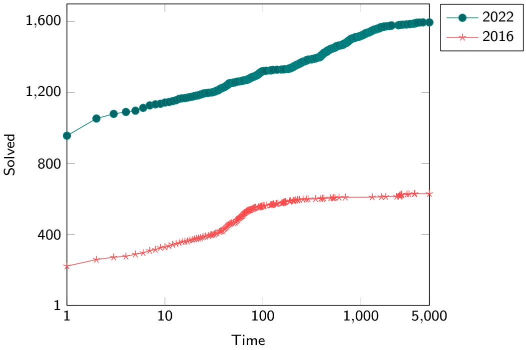
ApproxMC: In Pursuit of Scalability



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

2016 630 instances, each in $\leq$ 5000 seconds
2022 950 instances, each in $\leq$ 1 second

Time taken (seconds) for an instance

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

A speedup of 20,000$\times$ 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	Chain Formulas IJCAI-15 NeurIPS-20	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23
Software Development	CNF-XOR AAAI-19 CAV-20	Pseudo-Boolean CP-21	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	Chain Formulas IJCAI-15 NeurIPS-20	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23
Software Development	CNF-XOR AAAI-19 CAV-20	Pseudo-Boolean CP-21	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

x_1	x_2	x_3	
1	0	1	0
1	1	1	1
1	1	0	0

Gauss-Jordan Elimination

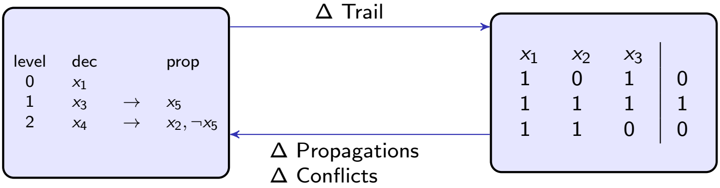
New Architecture for CNF-XOR Formulas

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

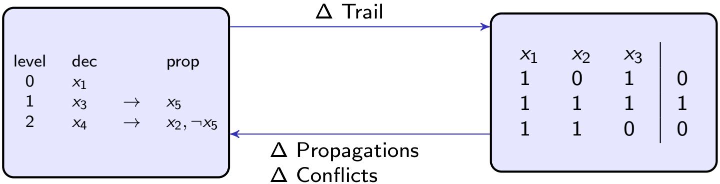
New Architecture for CNF-XOR Formulas

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]

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: $Q_1, Q_2, \dots Q_m$
- 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** $\mathbf{\frac{1}{2}}$?

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** $\mathbf{\frac{1}{2}}$?

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

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

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

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

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

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

[EGSS14,ZCSE16,AD17,ATD18]

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

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

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

Barrier: But $|\mathsf{Sol}(F)|$ 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	Chain Formulas IJCAI-15 NeurIPS-20	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23
Software Development	CNF-XOR AAAI-19 CAV-20	Pseudo-Boolean CP-21	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21

Reliability of Critical Infrastructure Networks

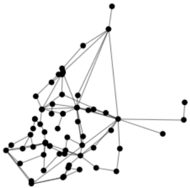
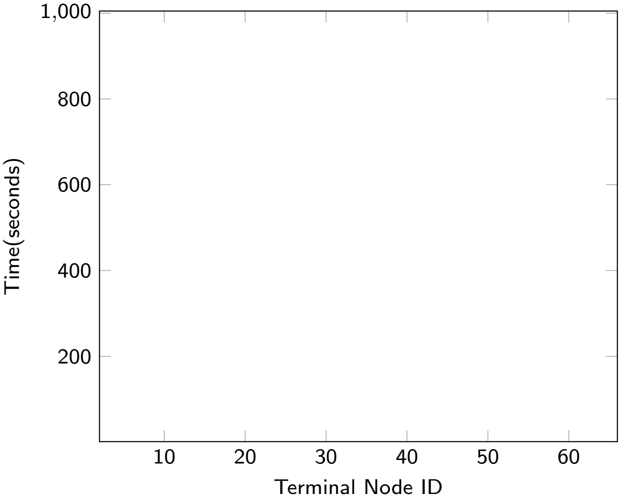


Figure: Plantersville, SC



Timeout = 1000 seconds

Reliability of Critical Infrastructure Networks

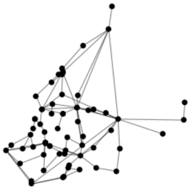
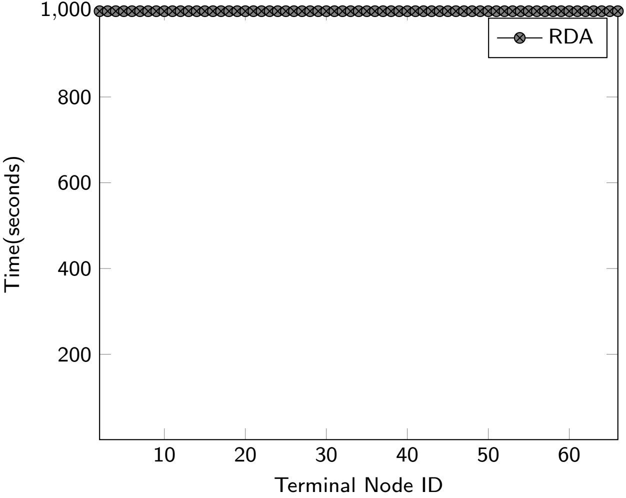


Figure: Plantersville, SC



Timeout = 1000 seconds

Reliability of Critical Infrastructure Networks

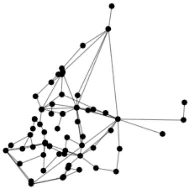
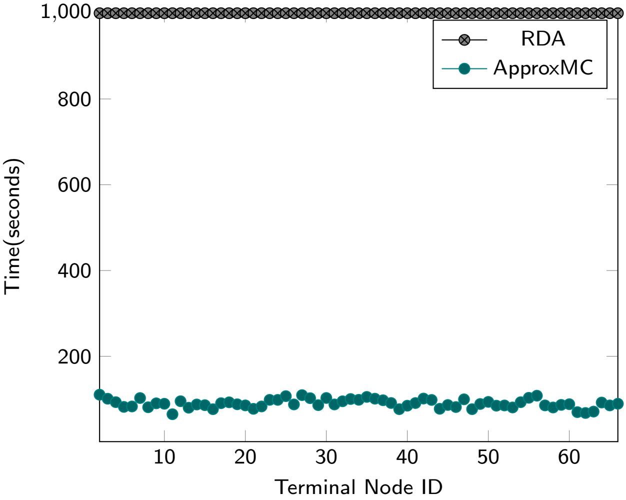


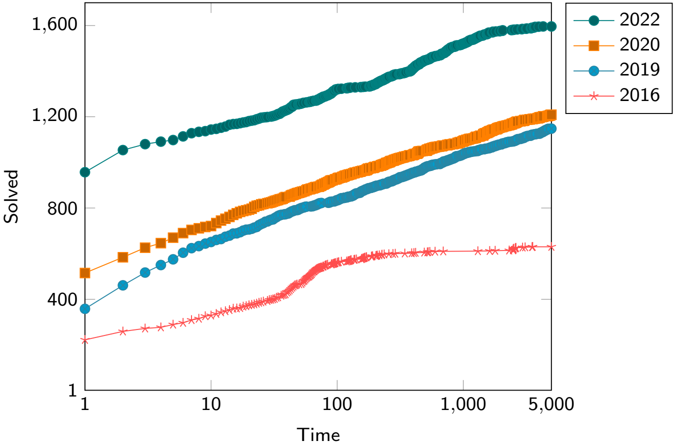
Figure: Plantersville, SC



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

A speedup of 20,000× 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 (MC-2020)

Competition

- Tracks: **Model Counting**, **Weighted Model Counting**, and **Projected Model Counti**
  (description, format, winners, report, slides, solvers, competition instances, submitte

Workshop

- Workshop 2020
  (program, abstracts, slides , recordings)

Model Counting (MC-2021)

Competition

- Tracks: **Model Counting**, **Weighted Model Counting**, and **Projected Model Counti**
  (description, StarExec Community, format, winners, report [tba], slides, solvers, com

Workshop on Counting and Sampling

- Workshop 2021
  (program, abstracts, slides , recordings)

Model Counting (MC-2022)

Competition

Part of FLoC Olympic Games

- Tracks: **Model Counting**, **Weighted Model Counting**, **Projected Model Counting**, **I**
  (description, StarExec Community, format, winners, report [tba], slides, solvers, com

Workshop on Counting and Sampling

- Workshop 2022 (in person event)
  (program, abstracts, slides)

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]

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]

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

- CACM Research Highlights [PVB**M**21]
- 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 [**M**SV19]

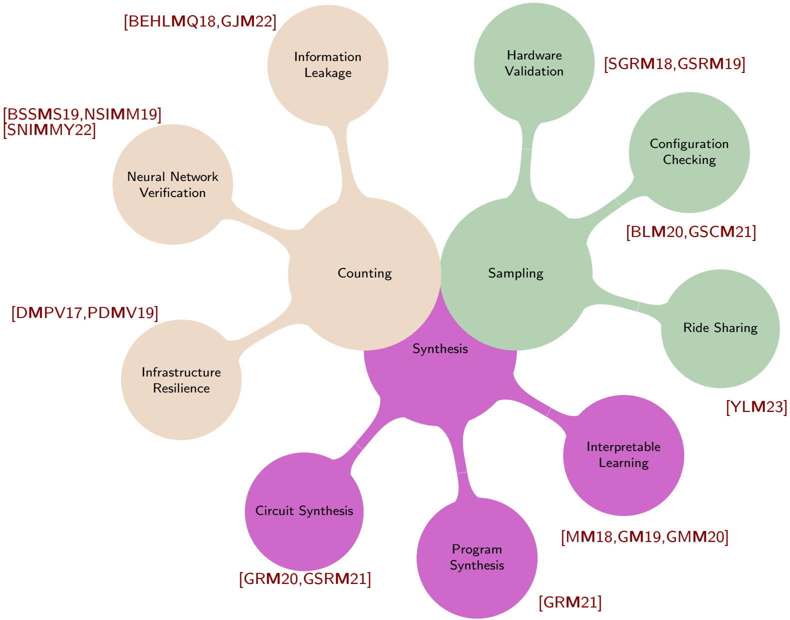
Streaming Counting over a stream: Distinct Elements
Example: How many unique customers visit website?
Fundamental problem in Databases

- CACM Research Highlights [PVB**M**21]
- ACM SIGMOD 2022 Research Highlight
- "Best of PODS 2021" by ACM TODS

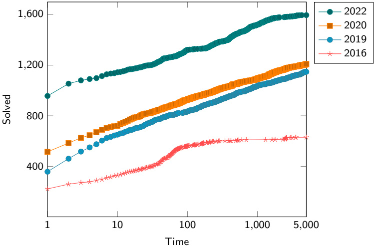
Unsatisfiable Subsets Count minimal subsets of clauses that are unsatisfiable.
Diagnosis metric for systems

- "Best Papers of CAV-20" by FMSD [B**M**20]

Counting, Sampling, and Synthesis

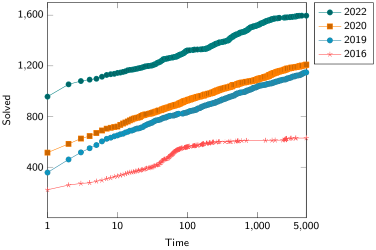


In Pursuit of Scalability

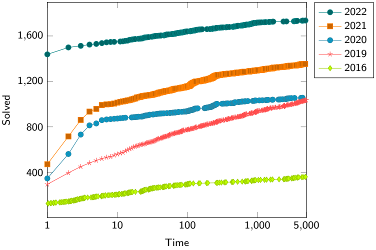


Counting over the years

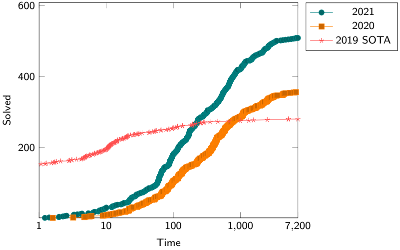
In Pursuit of Scalability



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 Scalablity

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

The Pursuit of Scalablity

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
- Preliminary Work: AAAI-19, NeurIPS-20, NeurIPS-21, CP-22, NeurIPS-22

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, Shubham Sharma, Mate Soos, Arijit Shaw, Tim van Bremen, Jiong Yang, Suwei Yang

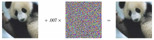
Collaborators

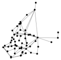
S. Akshay (IITB, IN)
Alyas Almaawi(UTAustin,US)
Eduard Baranov(UCLouvain, BE)
Rajkishore Barik(Intel,US)
Debabrota Basu(Chalmers,US)
Arnab Bhattacharya (NUS, SG)
Fabrizio Biondi (CS, FR)
Kian Ming Adam Chai(DSO, SG)
Diptarka Chakraborty(NUS,SG)
Sourav Chakraborty(ISIK, IN)
Supratik Chakraborty (IITB, IN)
Davin Choo(DSO, SG)
Zheng Leong Chua(IP, SG
Tiago Cogumbreiro(Rice,US)
Vincent Derkinderen(KUL, BE)
A. Dileep(IITD, India)
Jeffrey M. Dudek(Rice,US)
Leonardo Duenas-Osorio (Rice,US)
Michael A. Enescu(Inria, FR)
Daniel J. Fremont(UCB,US)
Dror Fried (Open U., IL)
Sutanu Gayen(NUS, SG)
Stephan Gocht (Lund U., SE)
Annelie Heuser(CNRS, FR)
Alexey Ignatiev(ULisboa, PT)
Alexander Ivrii(IBM, IL)
Saurabh Joshi(IITH, IN)
Mohan S. Kankanhalli(NUS, SG
Sarfraz Khurshid(UTAustin,US)
Raghav Kulkarni(CMI, IN)
Axel Legay (UCL, BE)
Massimo Lupascu(NUS, SG)
Deepak Majeti(Rice,US)
Sharad Malik(Princeton,US)
Dmitry Malioutov(IBM,US)
Joao Marques-Silva(ANITI, FR)
John M.Mellor-Crummey(Rice,US)
Rakesh Mistry(IITB, IN)
M.Mohammadalitajrishi(Polymtl,CA)
Karthik Murthy(Rice,US)
Nina Narodytska(VMware,US)
Roger Paredes(Rice,US)
Sri Raj Paul(Rice,US)
Aduri Pavan(ISU,US)
Gilles Pesant(Polymtl,CA)
Nicolas Prevot(London, UK)
Jean Quilbeuf(Inria, FR)
Subhajit Roy (IITK, IN)
Ammar F. Sabili(NUS, SG)
Vivek Sarkar(Rice,US)
Prateek Saxena(NUS, SG)
Jonathan Scarlett(NUS, SG)
Sanjit A. Seshia(UCB,US)
Shiqi Shen(NUS, SG)
Shweta Shinde(ETH,CH)
Aditya A. Shrotri(Rice,US)
Friedrich Slivovsky(TU Wien, AT)
Harold Soh(NUS, SG)
Muhammad Usman(UTAustin,US)
Moshe Y. Vardi(Rice,US)
N. V. Vinodchandran(UNL,US)
Kaiyuan Wang(Google,US)
Wenxi Wang(UTAustin,US)
Yaqi Xie(NUS, SG)
Ziwei Xu(NUS, SG)
Roland H. C. Yap(NUS, SG)

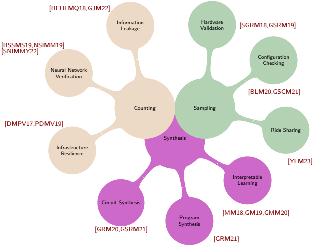
Funding Agencies: National Research Foundation, Ministry of Education, Defense Service Organization
Industrial Support: Grab Taxiholdings, Amazon, Microsoft Research

Counting, Sampling, and Synthesis

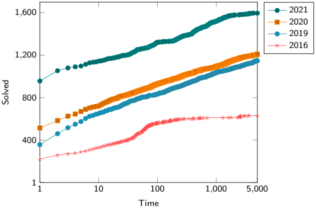
```
PC2 (char[] SP, char[] UI) {
        match = true;
        for (int i=0; i<UI.length(); i++) {
                if (SP[i] != UI[i]) match=false;
                else match = match;
        }
        if match return Yes;
        else return No;
}
```







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	Chain Formulas IJCAI-15 NeurIPS-20	Symmetry TACAS-20, AAAI-21	Prob. Caching IJCAI-19 AAAI-23
Software Development	CNF-XOR AAAI-19, CAV-20	Pseudo-Boolean CP-21	MaxSAT-XOR KR-21	Hardware Accelerator SAT-21



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 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