Beyond NP Revolution

Kuldeep S. Meel

National University of Singapore

CAALM Workshop

Turing, 1950: "Opinions may vary as to the complexity which is suitable in the child machine. One might try to make it as simple as possible consistent with the general principles. Alternatively one might have a complete system of logical inference "built in". In the latter case the store would be largely occupied with definitions and propositions. The propositions would have various kinds of status, e.g., well-established facts, conjectures, mathematically proved theorems, statements given by an authority, expressions having the logical form of proposition but not a belief-value"

Aristotle's Syllogisms

- All men are mortal
- Socrates is a man

Socrates is a mortal

Boole's Symbolic Logic

Boole's insight: Aristotle's syllogisms are about *classes* of objects, which can be treated *algebraically*.

> *"If an adjective, as 'good', is employed as a term of description, let us represent by a letter, as y, all things to which the description 'good' is applicable, i.e., 'all good things', or the class of 'good things'. Let it further be agreed that by the combination xy shall be represented that class of things to which the name or description represented by x and y are simultaneously applicable. Thus, if x alone stands for 'white' things and y for 'sheep', let xy stand for 'white sheep'.*

Boolean Satisfiability (SAT); Given a Boolean expression, using "and" ($\wedge$) "or", ($\vee$) and "not" ($\neg$), *is there a satisfying solution* (an assignment of 0's and 1's to the variables that makes the expression equal 1)?

Example:

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_3 \vee x_1 \vee x_4)$$

Solution: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$

History:

- William Stanley Jevons, 1835-1882: "I have given much attention, therefore, to lessening both the manual and mental labour of the process, and I shall describe several devices which may be adopted for saving trouble and risk of mistake."
- Ernst Schröder, 1841-1902: "Getting a handle on the consequences of any premises, or at least the fastest method for obtaining these consequences, seems to me to be one of the noblest, if not the ultimate goal of mathematics and logic."

History:

- William Stanley Jevons, 1835-1882: "I have given much attention, therefore, to lessening both the manual and mental labour of the process, and I shall describe several devices which may be adopted for saving trouble and risk of mistake."
- Ernst Schröder, 1841-1902: "Getting a handle on the consequences of any premises, or at least the fastest method for obtaining these consequences, seems to me to be one of the noblest, if not the ultimate goal of mathematics and logic."
- Cook, 1971, Levin, 1973: Boolean Satisfiability is NP-complete.

History:

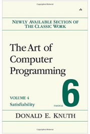
- William Stanley Jevons, 1835-1882: "I have given much attention, therefore, to lessening both the manual and mental labour of the process, and I shall describe several devices which may be adopted for saving trouble and risk of mistake."
- Ernst Schröder, 1841-1902: "Getting a handle on the consequences of any premises, or at least the fastest method for obtaining these consequences, seems to me to be one of the noblest, if not the ultimate goal of mathematics and logic."
- Cook, 1971, Levin, 1973: Boolean Satisfiability is NP-complete.
- Clay Institute, 2000: $1M Award!

Algorithmic Boolean Reasoning: Early History

- Davis and Putnam, 1958: "Computational Methods in The Propositional calculus", unpublished report to the NSA
- Davis and Putnam, JACM 1960: "A Computing procedure for quantification theory"
- Davis, Logemman, and Loveland, CACM 1962: "A machine program for theorem proving"
- Marques-Silva and Sakallah 1996, Zhang et al. 2001, Een and Sorensson 2003, Simon and Audemard 2009, Liang et al 2016
  CDCL = conflict-driven clause learning
  - Smart but cheap branching heuristics
  - Quick detection of unit clauses
  - Conflict Driven Clause Learning
  - Restarts

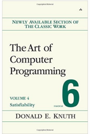
(Laurent Simon's talk will give a behind the scenes peek into SAT revolution)

Modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago. (Donald Knuth, 2016)



The Tale of Triumph of SAT Solvers

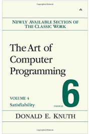
Modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago. (Donald Knuth, 2016)



Industrial usage of SAT Solvers: Hardware Verification, Planning, Genome Rearrangement, Telecom Feature Subscription, Resource Constrained Scheduling, Noise Analysis, Games, $\cdots$

The Tale of Triumph of SAT Solvers

Modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago. (Donald Knuth, 2016)



Industrial usage of SAT Solvers: Hardware Verification, Planning, Genome Rearrangement, Telecom Feature Subscription, Resource Constrained Scheduling, Noise Analysis, Games, $\cdots$

Now that SAT is "easy", it is time to look beyond satisfiability

Constrained Counting and Sampling

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$

- $\mathsf{Sol}(F) = \{$ solutions of F $\}$

Constrained Counting and Sampling

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$

- $\mathsf{Sol}(F) = \{$ solutions of F $\}$

- Constrained Counting: Determine $|\mathsf{Sol}(F)|$
- Constrained Sampling: Randomly sample from $\mathsf{Sol}(F)$ such that $\Pr[\text{y is sampled}] = \frac{1}{|\mathsf{Sol}(F)|}$

Constrained Counting and Sampling

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
  - Weight Function W: $\{0,1\}^n \mapsto [0,1]$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- $W(F) = \Sigma_{y \in \mathsf{Sol}(F)} W(y)$
- Constrained Counting: Determine $W(F)$
- Constrained Sampling: Randomly sample from $\mathsf{Sol}(F)$ such that Pr[y is sampled] $= \frac{W(y)}{W(F)}$

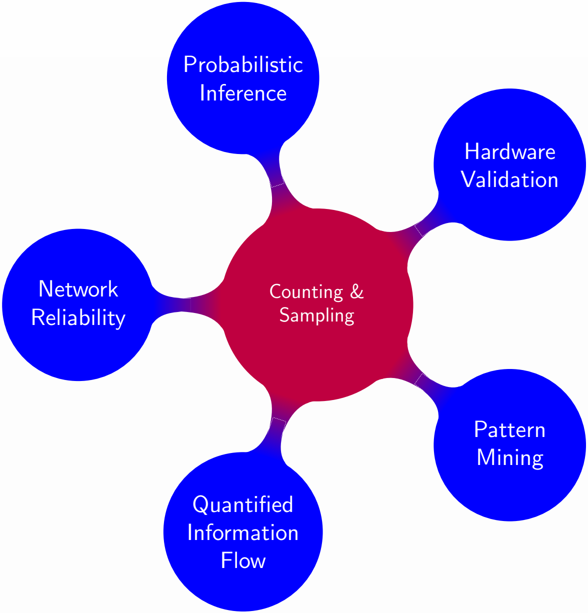
Constrained Counting and Sampling

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
  - Weight Function W: $\{0,1\}^n \mapsto [0,1]$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- $W(F) = \Sigma_{y \in \mathsf{Sol}(F)} W(y)$
- Constrained Counting: Determine $W(F)$
- Constrained Sampling: Randomly sample from $\mathsf{Sol}(F)$ such that $\Pr[\text{y is sampled}] = \frac{W(y)}{W(F)}$
- Given
  - $F := (X_1 \vee X_2)$
  - $W[(0,0)] = W[(1,1)] = \frac{1}{6}; W[(1,0)] = W[(0,1)] = \frac{1}{3}$
- $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$

Constrained Counting and Sampling

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
  - Weight Function W: $\{0,1\}^n \mapsto [0,1]$
- $\mathsf{Sol}(F) = \{$ solutions of F $\}$
- $W(F) = \Sigma_{y \in \mathsf{Sol}(F)} W(y)$
- Constrained Counting: Determine $W(F)$
- Constrained Sampling: Randomly sample from $\mathsf{Sol}(F)$ such that $\Pr[\text{y is sampled}] = \frac{W(y)}{W(F)}$
- Given
  - $F := (X_1 \vee X_2)$
  - $W[(0,0)] = W[(1,1)] = \frac{1}{6}; W[(1,0)] = W[(0,1)] = \frac{1}{3}$
- $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$
- $W(F) = \frac{1}{3} + \frac{1}{3} + \frac{1}{6} = \frac{5}{6}$

Applications across Computer Science



Today's Menu

Network Reliability

Probabilistic Inference

Part I Constrained Counting

Today's Menu

Network Reliability

Probabilistic Inference

Part I Constrained Counting Hashing Framework

Can we reliably predict the effect of natural disasters on critical infrastructure such as power grids?

Can we reliably predict the effect of natural disasters on critical infrastructure such as power grids?

Can we predict likelihood of a region facing blackout?

Reliability of Critical Infrastructure Networks

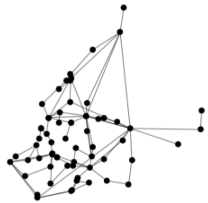


Figure: Plantersville, SC

- $G = (V, E)$; source node: s and terminal node t
- failure probability $g : E \rightarrow [0, 1]$
- Compute Pr[s and t are disconnected]?

Reliability of Critical Infrastructure Networks

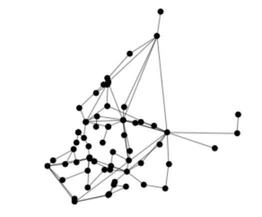


Figure: Plantersville, SC

- $G = (V, E)$; source node: s and terminal node t
- failure probability $g : E \rightarrow [0, 1]$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a $0/1$ vector of size $|E|$
- $W(\pi) = \Pr(\pi)$

Reliability of Critical Infrastructure Networks

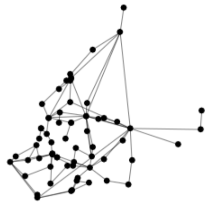


Figure: Plantersville, SC

- $G = (V, E)$; source node: s and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a 0/1 vector of size $|E|$
- $W(\pi) = \Pr(\pi)$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables

Reliability of Critical Infrastructure Networks

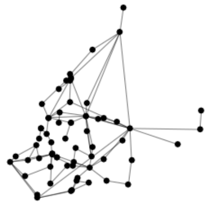


Figure: Plantersville, SC

- $G = (V, E)$; source node: s and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a 0/1 vector of size $|E|$
- $W(\pi) = \Pr(\pi)$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables
- Pr[s and t are disconnected] $= \sum_{\pi_{s,t}} W(\pi_{s,t})$

Reliability of Critical Infrastructure Networks

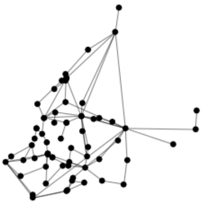


Figure: Plantersville, SC

Constrained Counting

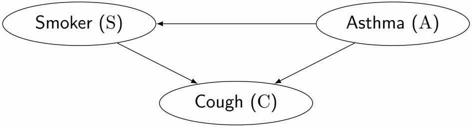
- $G = (V, E)$; source node: s and terminal node t
- failure probability $g : E \to [0, 1]$
- Compute Pr[s and t are disconnected]?
- π : Configuration (of network) denoted by a $0/1$ vector of size $|E|$
- $W(\pi) = \Pr(\pi)$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables
- Pr[s and t are disconnected] $= \sum_{\pi_{s,t}} W(\pi_{s,t})$

(DMPV, AAAI 17, RESS 2018)

Probabilistic Models

Patient	Cough	Smoker	Asthma
Alice	1	0	0
Bob	0	0	1
Randee	1	0	0
Tova	1	1	1
Azucena	1	0	0
Georgine	1	1	0
Shoshana	1	0	1
Lina	0	0	1
Hermine	1	1	1

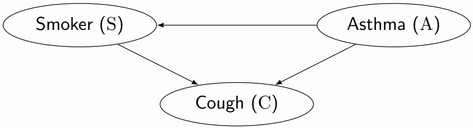
$\Longrightarrow$



Probabilistic Models

Patient	Cough	Smoker	Asthma
Alice	1	0	0
Bob	0	0	1
Randee	1	0	0
Tova	1	1	1
Azucena	1	0	0
Georgine	1	1	0
Shoshana	1	0	1
Lina	0	0	1
Hermine	1	1	1

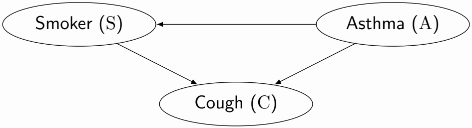
$\Longrightarrow$



Probabilistic Models

Patient	Cough	Smoker	Asthma
Alice	1	0	0
Bob	0	0	1
Randee	1	0	0
Tova	1	1	1
Azucena	1	0	0
Georgine	1	1	0
Shoshana	1	0	1
Lina	0	0	1
Hermine	1	1	1

$\Longrightarrow$

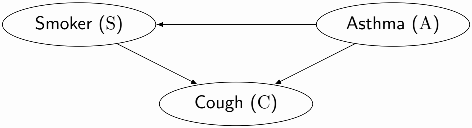


$$\Pr[\text{Asthma}(\mathrm{A}) \mid \text{Cough}(\mathrm{C})] = \frac{\Pr[\mathrm{A} \cap \mathrm{C}]}{\Pr[\mathrm{C}]}$$

Probabilistic Models

Patient	Cough	Smoker	Asthma
Alice	1	0	0
Bob	0	0	1
Randee	1	0	0
Tova	1	1	1
Azucena	1	0	0
Georgine	1	1	0
Shoshana	1	0	1
Lina	0	0	1
Hermine	1	1	1

$\Longrightarrow$



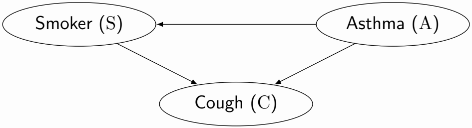
$$\Pr[\text{Asthma}(\mathrm{A}) \mid \text{Cough}(\mathrm{C})] = \frac{\Pr[\mathrm{A} \cap \mathrm{C}]}{\Pr[\mathrm{C}]}$$

$$F = \mathrm{A} \wedge \mathrm{C}$$

Probabilistic Models

Patient	Cough	Smoker	Asthma
Alice	1	0	0
Bob	0	0	1
Randee	1	0	0
Tova	1	1	1
Azucena	1	0	0
Georgine	1	1	0
Shoshana	1	0	1
Lina	0	0	1
Hermine	1	1	1

$\Longrightarrow$



$$\Pr[\text{Asthma}(\mathrm{A}) \mid \text{Cough}(\mathrm{C})] = \frac{\Pr[\mathrm{A} \cap \mathrm{C}]}{\Pr[\mathrm{C}]}$$

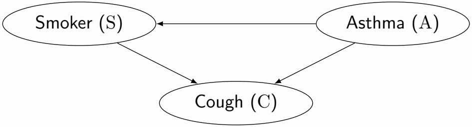
$$F = \mathrm{A} \wedge \mathrm{C}$$

$$\text{Sol}(F) = \{(\mathrm{A}, \mathrm{C}, \mathrm{S}), (\mathrm{A}, \mathrm{C}, \bar{\mathrm{S}})\}$$

Probabilistic Models

Patient	Cough	Smoker	Asthma
Alice	1	0	0
Bob	0	0	1
Randee	1	0	0
Tova	1	1	1
Azucena	1	0	0
Georgine	1	1	0
Shoshana	1	0	1
Lina	0	0	1
Hermine	1	1	1

$\Longrightarrow$



$$\Pr[\text{Asthma}(\mathrm{A}) \mid \text{Cough}(\mathrm{C})] = \frac{\Pr[\mathrm{A} \cap \mathrm{C}]}{\Pr[\mathrm{C}]}$$

$$F = \mathrm{A} \wedge \mathrm{C}$$

$$\mathsf{Sol}(F) = \{(\mathrm{A}, \mathrm{C}, \mathrm{S}), (\mathrm{A}, \mathrm{C}, \bar{\mathrm{S}})\}$$

$$\Pr[\mathrm{A} \cap \mathrm{C}] = \Sigma_{y \in \mathsf{Sol}(F)} W(y) = W(F)$$

Constrained Counting

(Roth, 1996)

Prior Work

Strong guarantees but poor scalability

- Exact counters (Birnbaum and Lozinskii 1999, Jr. and Schrag 1997, Sang et al. 2004, Thurley 2006)
- Hashing-based approach (Stockmeyer 1983, Jerrum Valiant and Vazirani 1986)

Weak guarantees but impressive scalability

- Bounding counters (Gomes et al. 2007,Kroc, Sabharwal, and Selman 2008, Gomes, Sabharwal, and Selman 2006, Kroc, Sabharwal, and Selman 2008)
- Sampling-based techniques (Wei and Selman 2005, Rubinstein 2012, Gogate and Dechter 2011)

Prior Work

Strong guarantees but poor scalability

- Exact counters (Birnbaum and Lozinskii 1999, Jr. and Schrag 1997, Sang et al. 2004, Thurley 2006)
- Hashing-based approach (Stockmeyer 1983, Jerrum Valiant and Vazirani 1986)

Weak guarantees but impressive scalability

- Bounding counters (Gomes et al. 2007,Kroc, Sabharwal, and Selman 2008, Gomes, Sabharwal, and Selman 2006, Kroc, Sabharwal, and Selman 2008)
- Sampling-based techniques (Wei and Selman 2005, Rubinstein 2012, Gogate and Dechter 2011)

How to bridge this gap between theory and practice?

Constrained Counting

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
  - Weight Function W: $\{0,1\}^n \mapsto [0,1]$
- ExactCount(F, W): Compute $W(F)$?
  - #P-complete (Valiant 1979)

- Given
  - Boolean variables $X_1, X_2, \cdots X_n$
  - Formula F over $X_1, X_2, \cdots X_n$
  - Weight Function W: $\{0,1\}^n \mapsto [0,1]$
- ExactCount(F, W): Compute $W(F)$?
  - #P-complete (Valiant 1979)
- ApproxCount($F, W, \varepsilon, \delta$): Compute C such that

$$\Pr[\frac{W(F)}{1+\varepsilon} \leq C \leq W(F)(1+\varepsilon)] \geq 1-\delta$$

From Weighted to Unweighted Counting

Boolean Formula F and weight function $W : \{0,1\}^n \to \mathbb{Q}^{\geq 0}$

Boolean Formula F'

$$W(F) = c(W) \times |\mathsf{Sol}(F')|$$

- Key Idea: Encode weight function as a set of constraints

From Weighted to Unweighted Counting

Boolean Formula F and weight function $W : \{0,1\}^n \to \mathbb{Q}^{\geq 0}$

Boolean Formula F'

$$W(F) = c(W) \times |\mathsf{Sol}(F')|$$

- Key Idea: Encode weight function as a set of constraints
- Caveat: $|F'| = O(|F| + |W|)$

(CFMV, IJCAI15)

From Weighted to Unweighted Counting

Boolean Formula F and weight function $W : \{0,1\}^n \to \mathbb{Q}^{\geq 0}$

Boolean Formula F'

$$W(F) = c(W) \times |\mathsf{Sol}(F')|$$

- Key Idea: Encode weight function as a set of constraints
- Caveat: $|F'| = O(|F| + |W|)$

(CFMV, IJCAI15)

How do we estimate $|\mathsf{Sol}(F')|$?

Counting in Chennai

How many people in Chennai like coffee?

- Population of Chennai $= 7.1$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)

Counting in Chennai

How many people in Chennai like coffee?

- Population of Chennai $= 7.1$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 7.1M/50

Counting in Chennai

How many people in Chennai like coffee?

- Population of Chennai = 7.1M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 7.1M/50
  - If only 5 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50

Counting in Chennai

How many people in Chennai like coffee?

- Population of Chennai $= 7.1$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 7.1M/50
  - If only 5 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 Chennai

How many people in Chennai like coffee?

- Population of Chennai = 7.1M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 7.1M/50
  - If only 5 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 Chennai

How many people in Chennai like coffee?

- Population of Chennai = 7.1M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 7.1M/50
  - If only 5 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

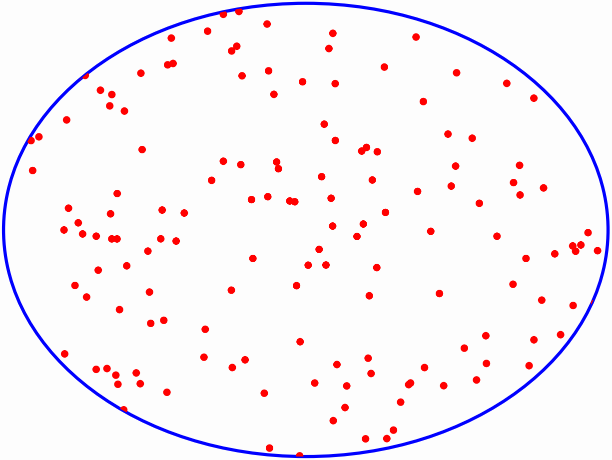
Counting in Chennai

How many people in Chennai like coffee?

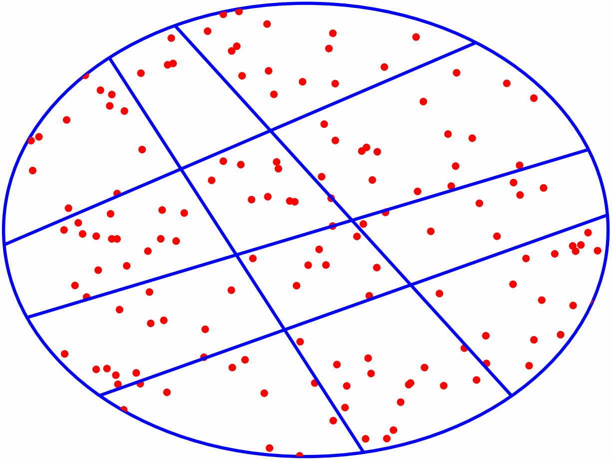
- Population of Chennai = 7.1M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 7.1$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 7.1M/50
  - If only 5 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 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

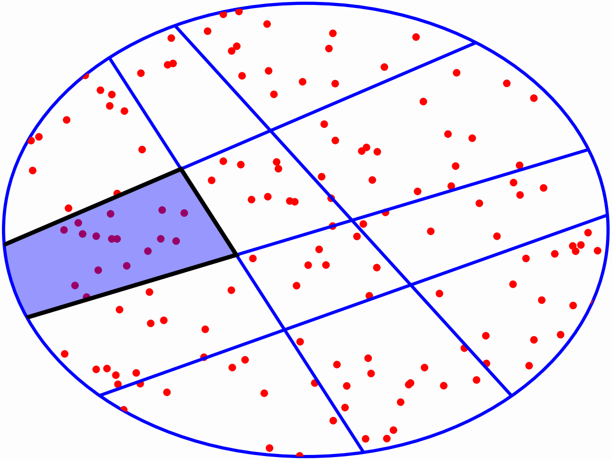


As Simple as Counting Dots



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?

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)
- Solutions in a cell α: $\mathsf{Sol}(F) \cap \{y \mid h(y) = \alpha\}$

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)
- Solutions in a cell α: $\mathsf{Sol}(F) \cap \{y \mid h(y) = \alpha\}$
- 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)
- Solutions in a cell α: $\mathsf{Sol}(F) \cap \{y \mid h(y) = \alpha\}$
- Deterministic h unlikely to work
- Choose h randomly from a large family H of hash functions

  Universal Hashing (Carter and Wegman 1977)

2-Universal Hashing

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

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

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

2-Universal Hashing

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

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

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

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

2-Universal Hash Functions

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

2-Universal Hash Functions

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

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

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

2-Universal Hash Functions

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

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

- Solutions in a cell: $F \wedge Q_1 \cdots \wedge Q_m$
- Performance of state of the art SAT solvers degrade with increase in the size of XORs (SAT Solvers != SAT oracles)

Improved Universal Hash Functions

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

Improved Universal Hash Functions

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

Improved Universal Hash Functions

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

Improved Universal Hash Functions

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

Algorithmic procedure to determine I?

Improved Universal Hash Functions

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

Algorithmic procedure to determine I?

- FP^{NP} procedure via reduction to Minimal Unsatisfiable Subset

Improved Universal Hash Functions

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

Algorithmic procedure to determine I?

- FP^{NP} procedure via reduction to Minimal Unsatisfiable Subset
- Two orders of magnitude runtime improvement
  (IMMV CP15, Best Student Paper) (IMMV Constraints16, Invited Paper)

Challenges

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

- Independent Support-based 2-Universal Hash Functions

Challenge 2 How many cells?

Question 2: How many cells?

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

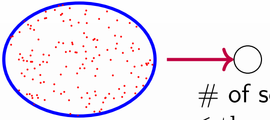
Question 2: How many cells?

- A cell is small if it has $\text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\text{thresh}}$

Question 2: How many cells?

- A cell is small if it has $\text{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\text{thresh}}$
  - Check for every $m = 0, 1, \cdots n$ if the number of solutions $\leq \text{thresh}$

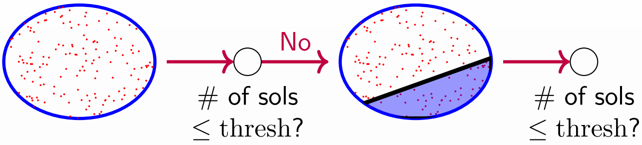
ApproxMC(F, ε, δ)

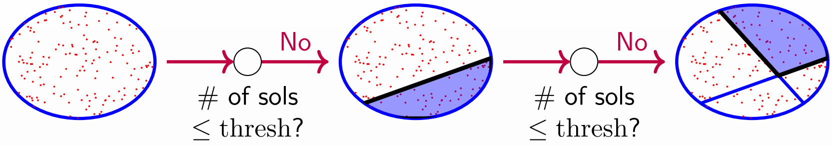


\# of sols

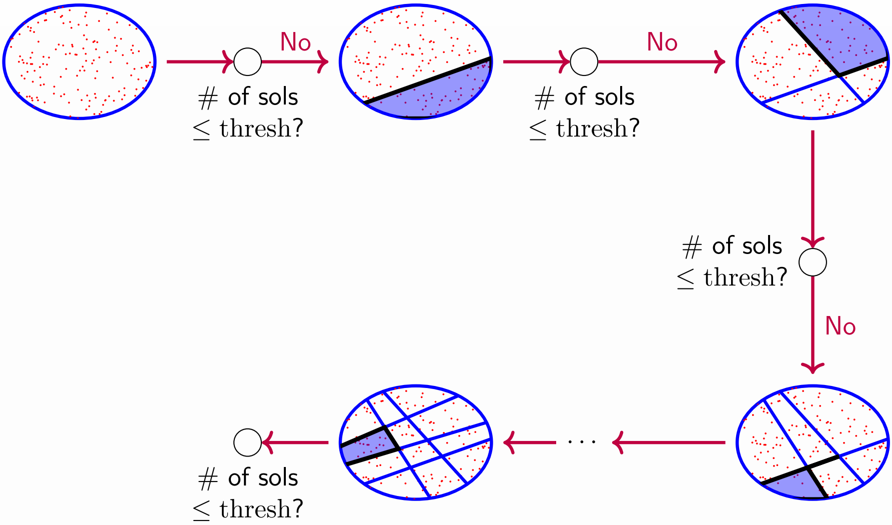
$\leq$ thresh?

ApproxMC(F, ε, δ)

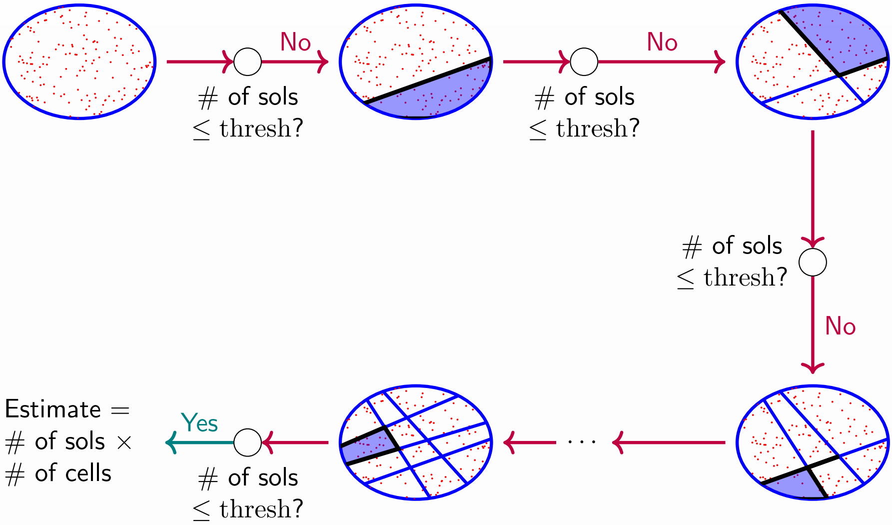



No
# of sols
$\leq$ thresh?
No
# of sols
$\leq$ thresh?

ApproxMC(F, ε, δ)



ApproxMC(F, ε, δ)



ApproxMC(F, ε, δ)

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

ApproxMC(F, ε, δ)

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

ApproxMC(F, ε, δ)

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

ApproxMC(F, ε, δ)

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \mathrm{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \mathrm{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \mathrm{thresh}$
- Stop at the first m where Query m returns YES and return estimate as $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m) \times 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES
  - Logarithmic search (# of SAT calls: $\mathcal{O}(\log n)$)
- **Will this work? Will the "m" where we stop be close to m^*?**
  - Challenge Query i and Query j are not independent
  - Independence crucial to analysis (Stockmeyer 1983, $\cdots$)

ApproxMC(F, ε, δ)

- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Query 1: Is $\#(F \wedge Q_1) \leq \mathrm{thresh}$
  - Query 2: Is $\#(F \wedge Q_1 \wedge Q_2) \leq \mathrm{thresh}$
  - $\cdots$
  - Query n: Is $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_n) \leq \mathrm{thresh}$
- Stop at the first m where Query m returns YES and return estimate as $\#(F \wedge Q_1 \wedge Q_2 \cdots \wedge Q_m) \times 2^m$
- Observation: $\#(F \wedge Q_1 \cdots \wedge Q_i \wedge Q_{i+1}) \leq \#(F \wedge Q_1 \cdots \wedge Q_i)$
  - If Query i returns YES, then Query $i+1$ must return YES
  - Logarithmic search (# of SAT calls: $\mathcal{O}(\log n)$)
- **Will this work? Will the "m" where we stop be close to m^*?**
  - Challenge Query i and Query j are not independent
  - Independence crucial to analysis (Stockmeyer 1983, $\cdots$)
  - Key Insight: The probability of making a bad choice of Q_i is very small for $i \ll m^*$

(CMV, IJCAI16)

Taming the Curse of Dependence

Let $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\text{thresh}}$ $(m^* = \log(\frac{|\mathsf{Sol}(F)|}{\text{thresh}}))$

Lemma (1)

ApproxMC (F, ε, δ) *terminates with* $m \in \{m^* - 1, m^*\}$ *with probability* $\geq$ *0.8*

Lemma (2)

For $m \in \{m^* - 1, m^*\}$*, estimate obtained from a randomly picked cell lies within a tolerance of* ε *of* $|\mathsf{Sol}(F)|$ *with probability* $\geq$ *0.8*

ApproxMC(F, ε, δ)

Theorem (Correctness)

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

Theorem (Complexity)

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

- *Prior work required $\mathcal{O}(\frac{\boldsymbol{n} \log \boldsymbol{n} \log(\frac{1}{\delta})}{\varepsilon})$ calls to SAT oracle (Stockmeyer 1983)*

ApproxMC(F, ε, δ)

Theorem (Correctness)

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

Theorem (Complexity)

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

- *Prior work required $\mathcal{O}(\frac{\boldsymbol{n} \log \boldsymbol{n} \log(\frac{1}{\delta})}{\varepsilon})$ calls to SAT oracle (Stockmeyer 1983)*

Theorem (FPRAS for DNF; (MSV, FSTTCS 17; CP 18, Invited Paper))

If F is a DNF formula, then ApproxMC is FPRAS – fundamentally different from the only other known FPRAS for DNF (Karp, Luby 1983)

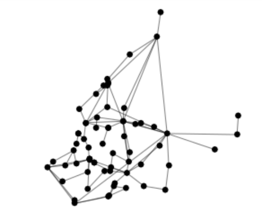
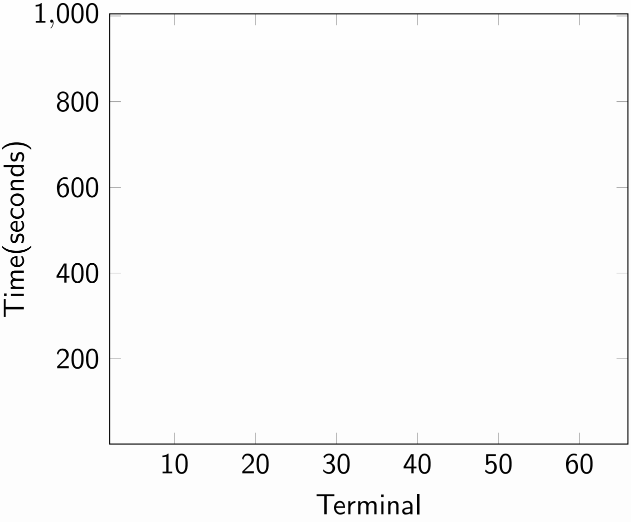


Figure: Plantersville, SC

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

Timeout = 1000 seconds



(DMPV, AAAI17)

Reliability of Critical Infrastructure Networks

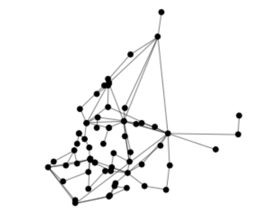
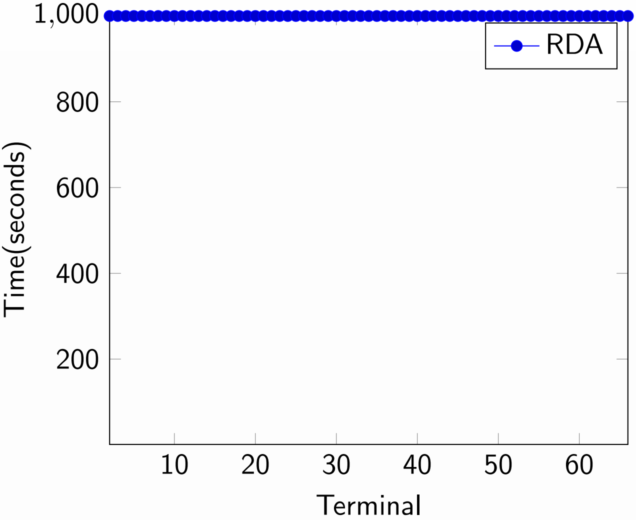


Figure: Plantersville, SC

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

Timeout = 1000 seconds



(DMPV, AAAI17)

Reliability of Critical Infrastructure Networks

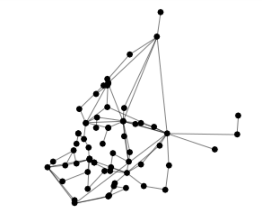
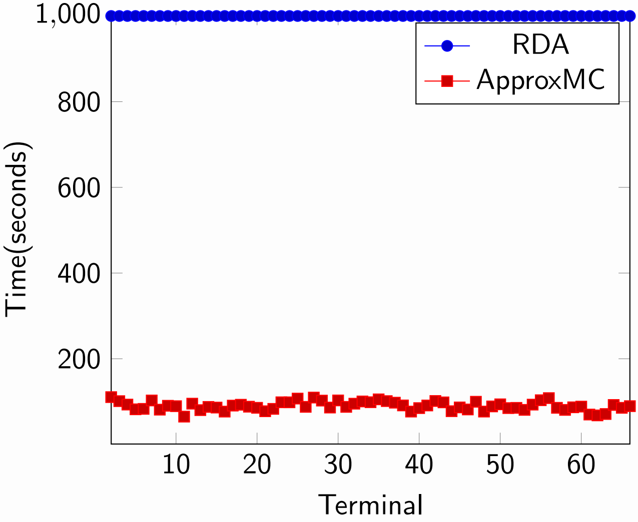


Figure: Plantersville, SC

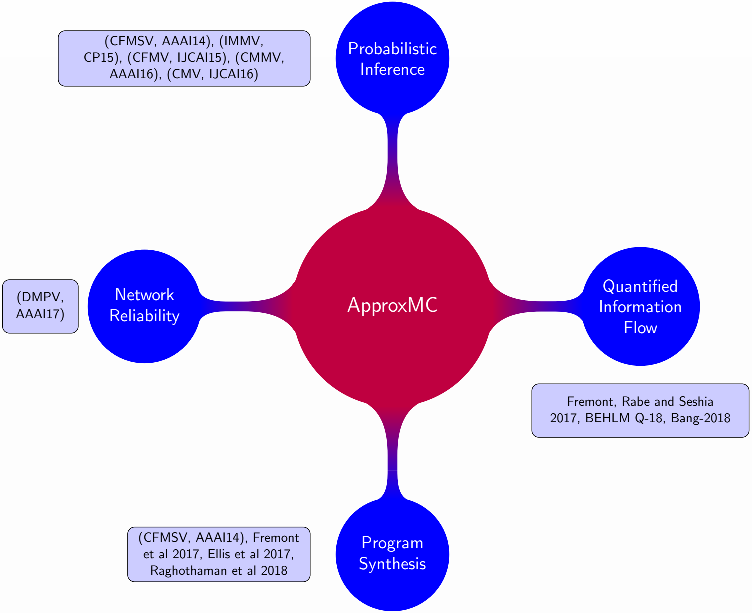
- $G = (V, E)$; source node: s
- Compute Pr[t is disconnected]?

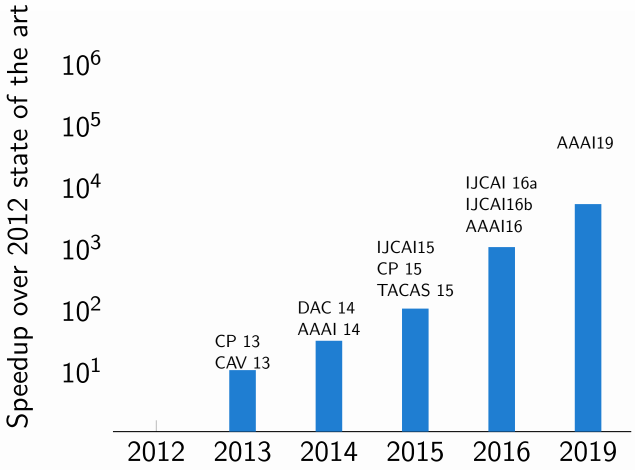
Timeout = 1000 seconds



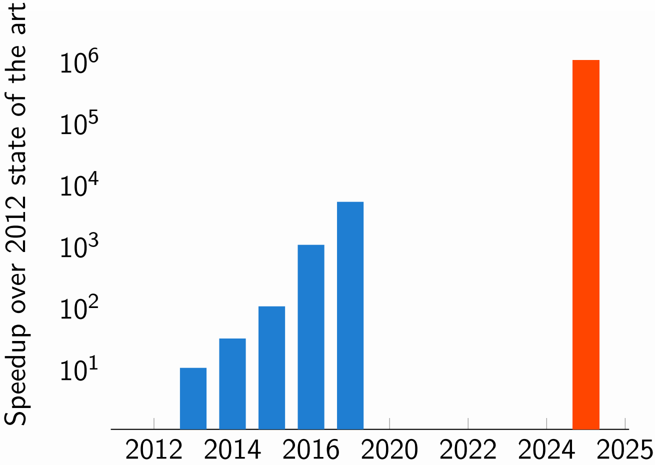
(DMPV, AAAI17)

Beyond Network Reliability




Speedup over 2012 state of the art
10^6
10^5
10^4
10^3
10^2
10^1
CP 13
CAV 13
DAC 14
AAAI 14
IJCAI15
CP 15
TACAS 15
IJCAI 16a
IJCAI16b
AAAI16
AAAI19
2012
2013
2014
2015
2016
2019

Mission 2025: Constrained Counting Revolution



Requires combinations of ideas from theory, statistics and systems

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration
- Verification of counting (CM, AAAI19)

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration
- Verification of counting (CM, AAAI19)
- Designing hardware accelerators – similar to advances in deep learning

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration
- Verification of counting (CM, AAAI19)
- Designing hardware accelerators – similar to advances in deep learning

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration
- Verification of counting (CM, AAAI19)
- Designing hardware accelerators – similar to advances in deep learning
- Understanding and applying counting to real world use-cases

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration
- Verification of counting (CM, AAAI19)
- Designing hardware accelerators – similar to advances in deep learning
- Understanding and applying counting to real world use-cases

We can only see a short distance ahead but we can see plenty there that needs to be done (Turing, 1950)

Mission 2025: Constrained Counting Revolution

- Extending to SMT (CMMV, AAAI16)
- Tighter integration between solvers and algorithms (SM, AAAI19)
- Handling weighted distributions: Connections to theory of integration
- Verification of counting (CM, AAAI19)
- Designing hardware accelerators – similar to advances in deep learning
- Understanding and applying counting to real world use-cases

We can only see a short distance ahead but we can see plenty there that needs to be done (Turing, 1950)

We are hiring: interns, research assistants PhD students, and postdocs. Visit www.comp.nus.edu.sg/ meel for details on how to apply.