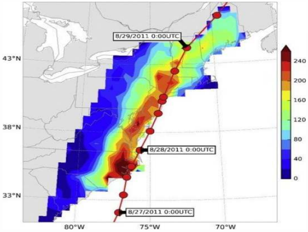
Constrained Counting and Sampling: Bridging the Gap between Theory and Practice

Kuldeep S. Meel

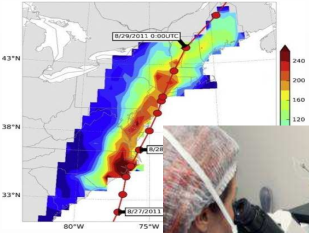
Rice University, USA

www.kuldeepmeel.com
kuldeep@rice.edu

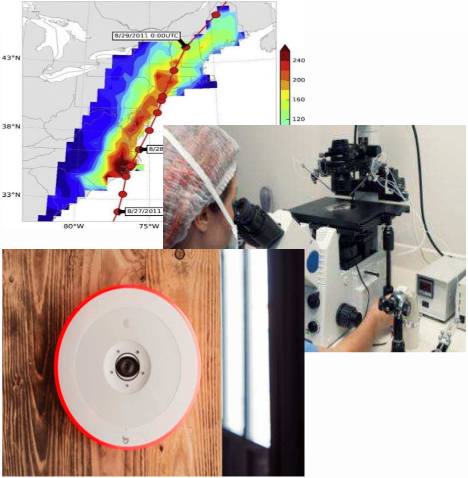
- Data
- Uncertainty



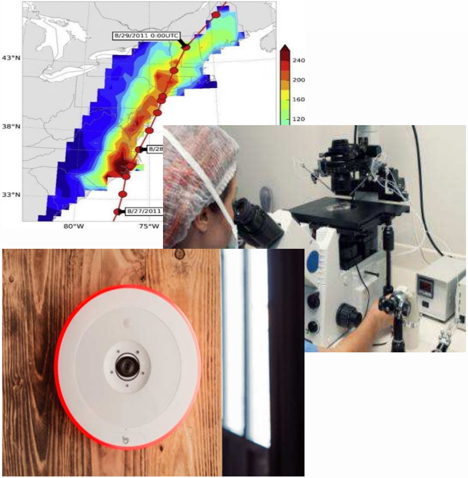
- Data
- Uncertainty



- Data
- Uncertainty

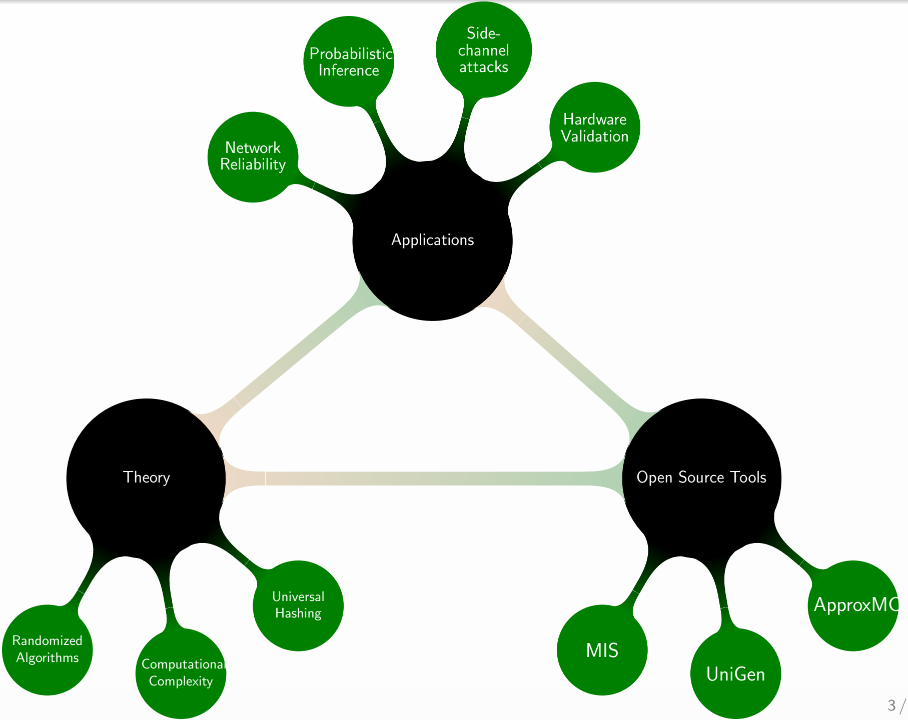


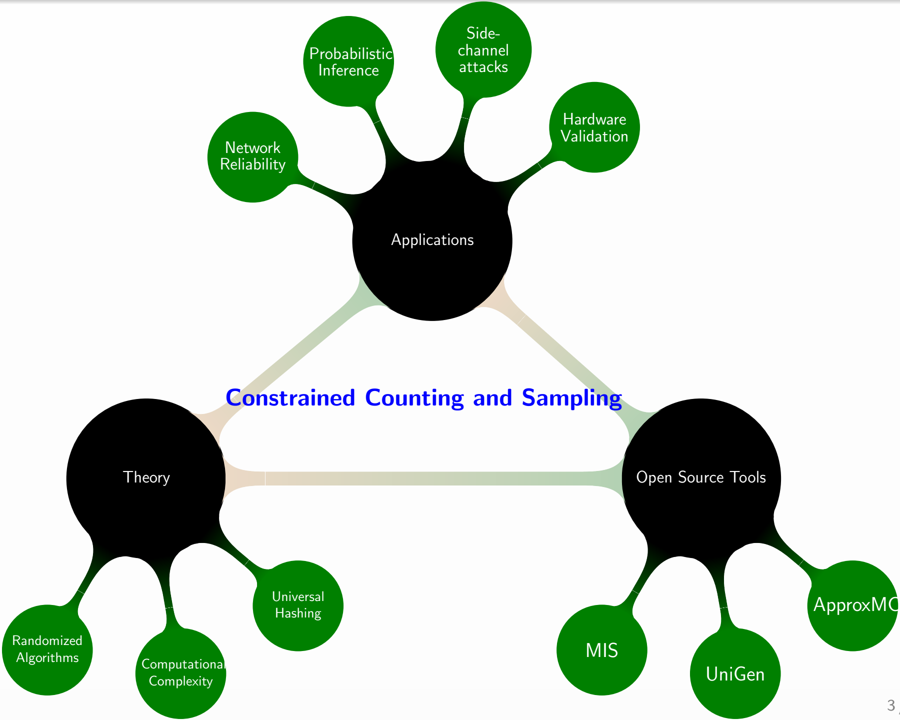
- Data
- Uncertainty



- Data
- Uncertainty

- Scalable
- Guarantees of Accuracy


Applications
Network Reliability
Probabilistic Inference
Side-channel attacks
Hardware Validation
Theory
Randomized Algorithms
Computational Complexity
Universal Hashing
Open Source Tools
MIS
UniGen
ApproxMC



Constrained Counting and Sampling
Applications
Network Reliability
Probabilistic Inference
Side-channel attacks
Hardware Validation
Theory
Randomized Algorithms
Computational Complexity
Universal Hashing
Open Source Tools
MIS
UniGen
ApproxMC

A Tale of Constraints

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

Example: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3)$

$x_1 = 1, x_2 = 1, x_3 = 1$

A Tale of Constraints

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

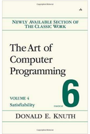
Example: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3)$

$x_1 = 1, x_2 = 1, x_3 = 1$

Ernst Schroder, 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."

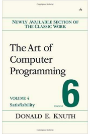
Cook, 1971; Levin, 1973: SAT is NP-complete

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

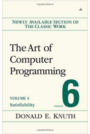
Modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago. (Donald Knuth, 2016)



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: 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[y is sampled] $= \frac{W(y)}{W(F)}$
- $F := (X_1 \vee X_2)$;
  $W[(0,0)] = W[(1,1)] = \frac{1}{6}; W[(1,0)] = W[(0,1)] = \frac{1}{3}$
- $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$

- 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 $\Pr[\text{y is sampled}] = \frac{W(y)}{W(F)}$
- $F := (X_1 \vee X_2)$;
  $W[(0,0)] = W[(1,1)] = \frac{1}{6}; W[(1,0)] = W[(0,1)] = \frac{1}{3}$
- $\mathsf{Sol}(F) = \{(0,1),(1,0),(1,1)\}$
- $W(F) = \frac{1}{3} + \frac{1}{3} + \frac{1}{6} = \frac{5}{6}$

Today's Menu

Network Reliability

Probabilistic Inference

Today's Menu

Network Reliability

Probabilistic Inference

Constrained Counting

Today's Menu

Network Reliability

Probabilistic Inference

Constrained Counting

Hashing Framework

Today's Menu

Network Reliability

Probabilistic Inference

Hardware Validation

Constrained Counting

Constrained Sampling

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?

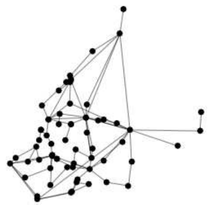
Can we predict likelihood of a region facing blackout?

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]?

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 $|E|$
- $W(\pi) = \Pr(\pi)$



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

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] $= \sum_{\pi_{s,t}} W(\pi_{s,t})$

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 $|E|$
- $W(\pi) = \Pr(\pi)$
- $\pi_{s,t}$: configuration where s and t are disconnected
  - Represented as a solution to set of constraints over edge variables
- Pr[s and t are disconnected] $= \sum_{\pi_{s,t}} W(\pi_{s,t})$

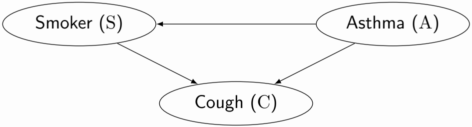
Constrained Counting

(D**M**PV, AAAI 17)

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

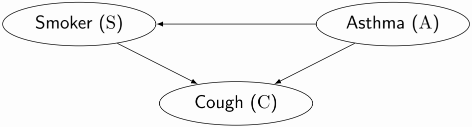
$\Longrightarrow$



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

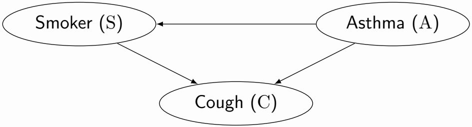
$\Longrightarrow$



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

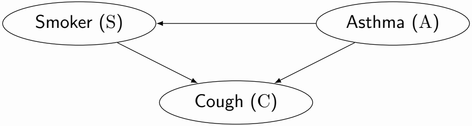
$\Longrightarrow$



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

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



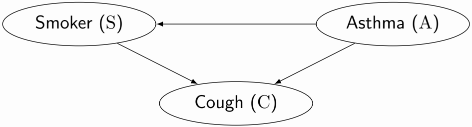
$$\Pr[\text{Asthma}(\mathrm{A}) \mid \text{Cough}(\mathrm{C})] = \frac{\Pr[\mathrm{A} \cap \mathrm{C}]}{\Pr[\mathrm{C}]}$$

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

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

$\Longrightarrow$



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

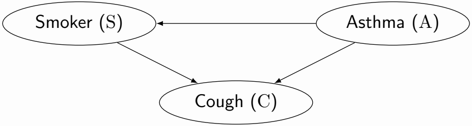
$$F = \mathrm{A} \wedge \mathrm{C}$$

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

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

$\Longrightarrow$



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

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

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

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

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

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?

- ExactCount(F, W): Compute $W(F)$?
  - #P-complete (Valiant 1979)

- 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 $W : \{0,1\}^n \rightarrow \mathbb{Q}^{\geq 0}$

Boolean Formula F'

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

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

(CF**M**V, 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

(CF**M**V, IJCAI15)

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

Counting in Singapore

How many people in Singapore like coffee?

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

Counting in Singapore

How many people in Singapore like coffee?

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

Counting in Singapore

How many people in Singapore like coffee?

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

How many people in Singapore like coffee?

- Population of NUS $= 5.6$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 5.6$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 5.6M/50
  - If only 5 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- NP Query: Find a person who likes coffee

Counting in Singapore

How many people in Singapore like coffee?

- Population of NUS = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 5.6$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 5.6M/50
  - If only 5 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- NP Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person y

Counting in Singapore

How many people in Singapore like coffee?

- Population of NUS $= 5.6$M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 5.6$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 5.6M/50
  - If only 5 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- NP Query: Find a person who likes coffee
- A SAT solver can answer queries like:
  - Q1: Find a person who likes coffee
  - Q2: Find a person who likes coffee and is not person y
- Attempt #2: Enumerate every person who likes coffee

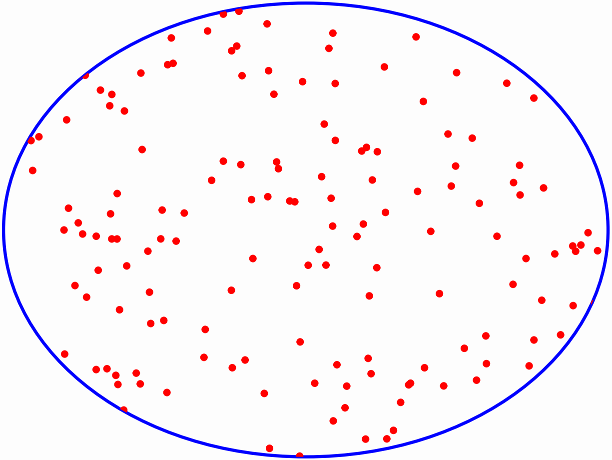
Counting in Singapore

How many people in Singapore like coffee?

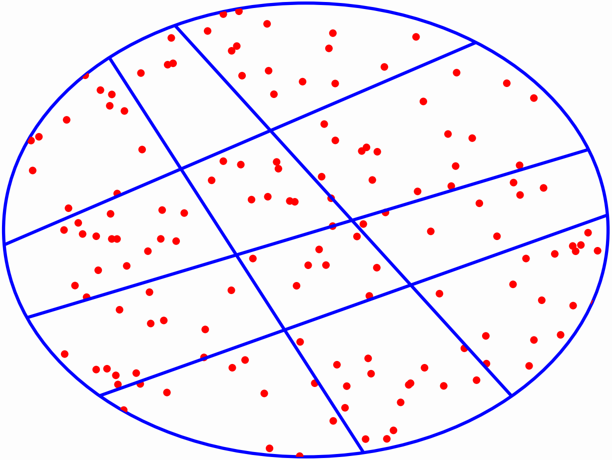
- Population of NUS = 5.6M
- Assign every person a unique ($n =$) 23 bit identifier ($2^n = 5.6$M)
- Attempt #1: Pick 50 people and count how many of them like coffee and multiple by 5.6M/50
  - If only 5 people like coffee, it is unlikely that we will find anyone who likes coffee in our sample of 50
- NP 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 $\mathcal{O}(\log n)$?

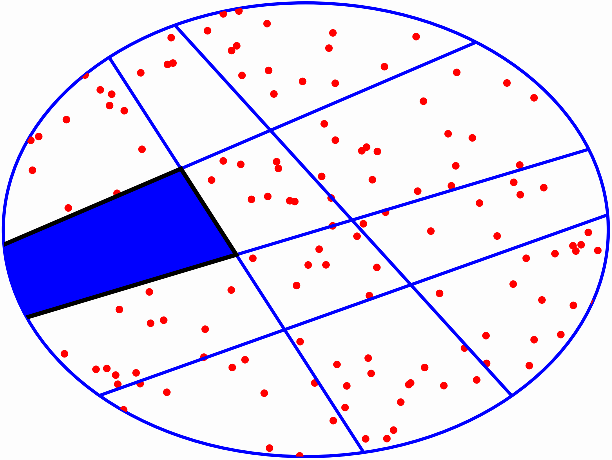
As Simple as Counting Dots



As Simple as Counting Dots



Pick a random cell



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

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: $\mathsf{Sol}(F) \cap \{y \mid h(y) = \alpha\}$

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

  $\boxed{\text{Universal Hashing (Carter and Wegman 1977)}}$

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 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

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

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

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

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

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

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

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

- Solutions in a cell: $F \wedge Q_1 \cdots \wedge Q_m$
- Performance of state of the art SAT solvers degrade with increase in the size of XORs (SAT Solvers != SAT oracles)
- Two orders of magnitude reduction in the size of XORs by embedding formula into smaller dimension ("Independent Support")

(IMMV CP15, Best Student Paper) (IMMV Constraints16, Invited Paper)

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 less than $\mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions

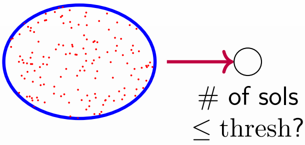
Question 2: How many cells?

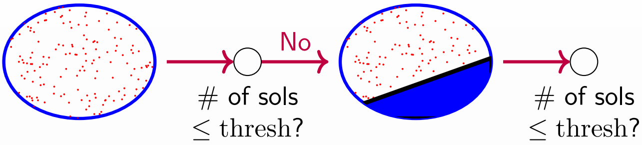
- A cell is small if it has less than $\mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$

Question 2: How many cells?

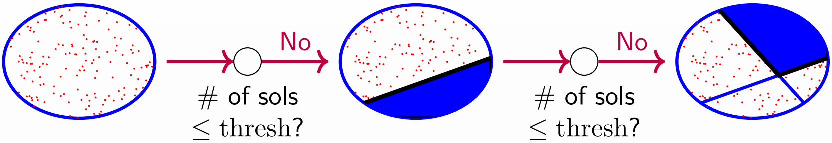
- A cell is small if it has less than $\mathrm{thresh} = 5(1 + \frac{1}{\varepsilon})^2$ solutions
- We want to partition into 2^{m^*} cells such that $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$
  - Check for every $m = 0, 1, \cdots n$ if the number of solutions $\leq \mathrm{thresh}$

ApproxMC(F, ε, δ)

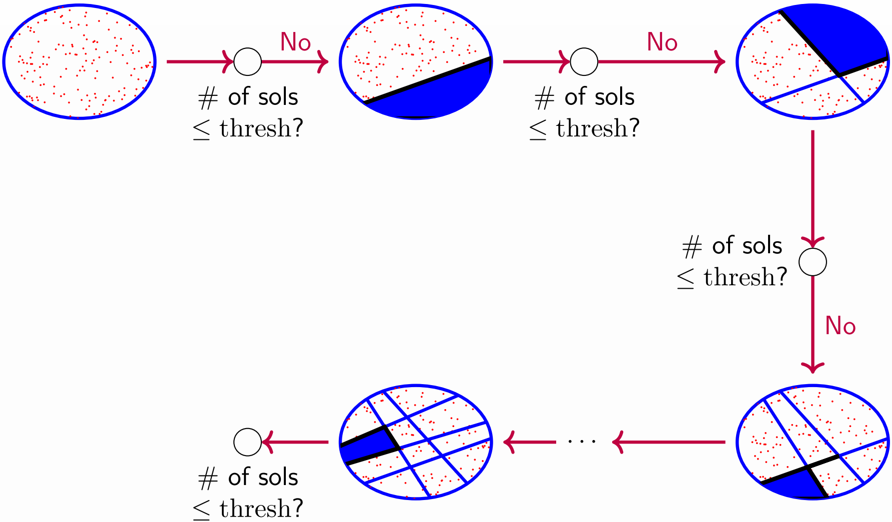



No
# of sols
$\leq$ thresh?
# of sols
$\leq$ thresh?

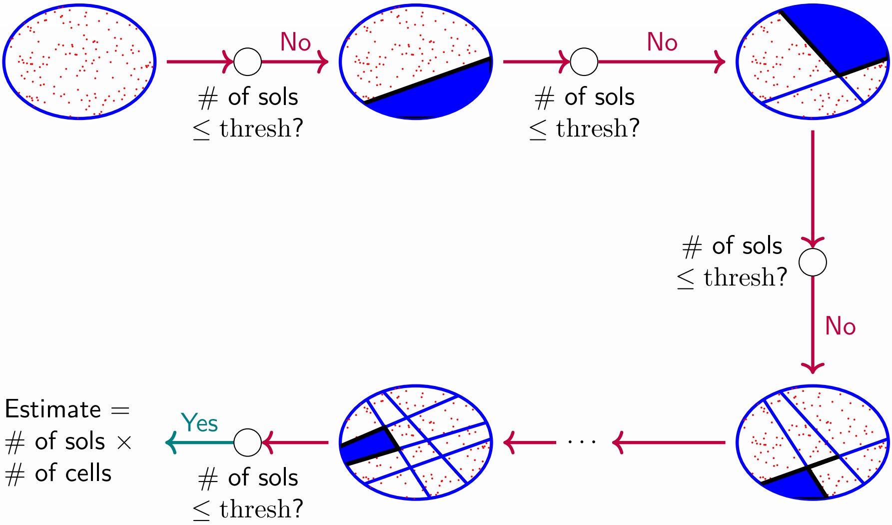
ApproxMC(F, ε, δ)



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)

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)

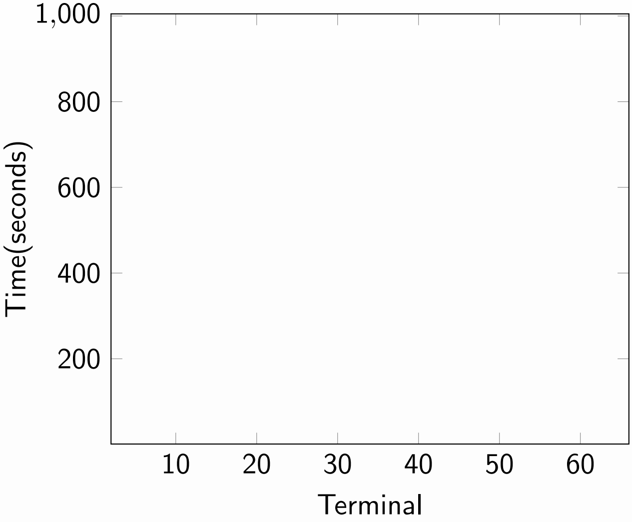
$ApproxMC(F, \varepsilon, \delta)$ makes $\mathcal{O}(\frac{\log n \log(\frac{1}{\delta})}{\varepsilon^2})$ calls to SAT oracle.

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

Plantersville, SC

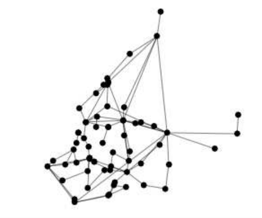
- $G = (V, E)$; source node: s
- Compute Pr[t is disconnected]?

Timeout = 1000 seconds



(DMPV, AAAI17)

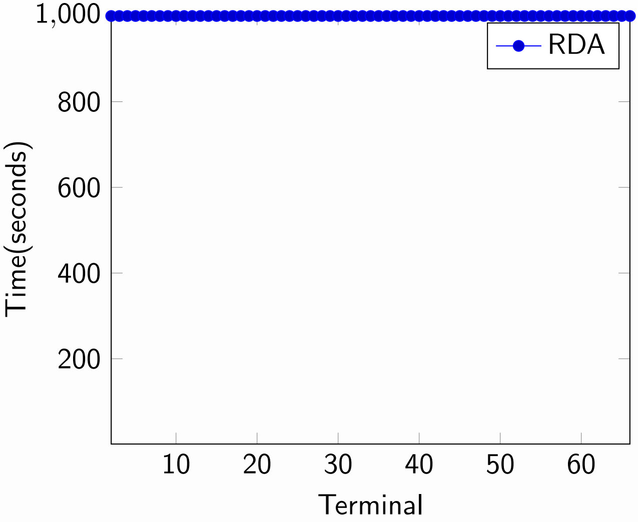
Reliability of Critical Infrastructure Networks



Plantersville, SC

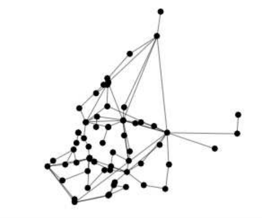
- $G = (V, E)$;
  source node: s
- Compute Pr[t is disconnected]?

Timeout = 1000 seconds

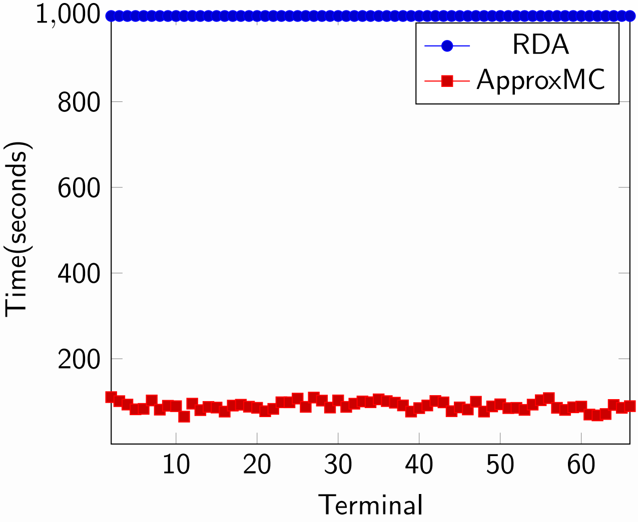


(DMPV, AAAI17)

Reliability of Critical Infrastructure Networks



Plantersville, SC

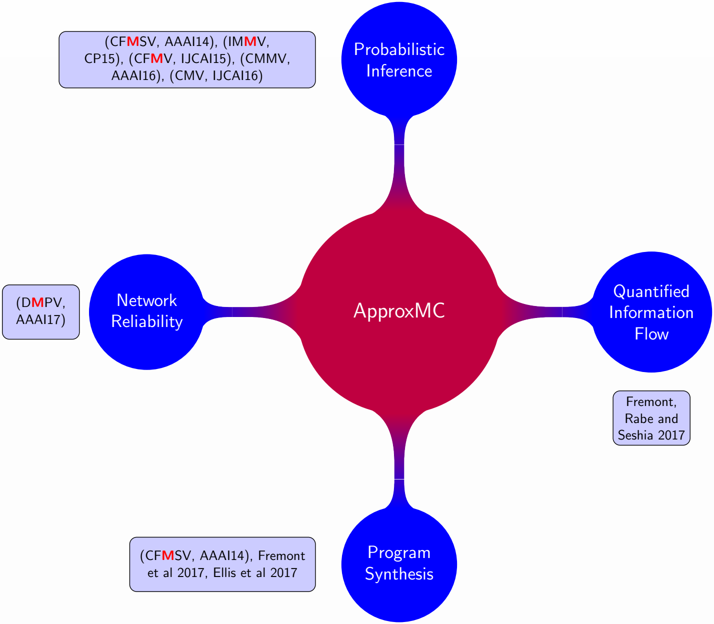


- $G = (V, E)$; source node: s
- Compute Pr[t is disconnected]?

Timeout = 1000 seconds

(DMPV, AAAI17)

Beyond Network Reliability



Network Reliability

Probabilistic Inference

Constrained Counting

Network Reliability

Probabilistic Inference

Constrained Counting

Hashing Framework

Network Reliability

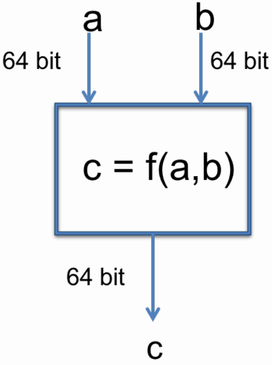
Probabilistic Inference

Hardware Validation

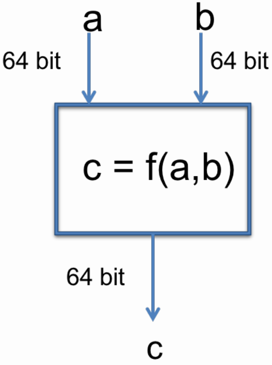
Constrained Counting

Hashing Framework

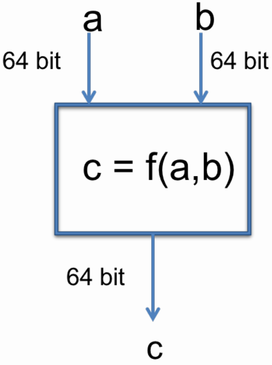
Hardware Validation



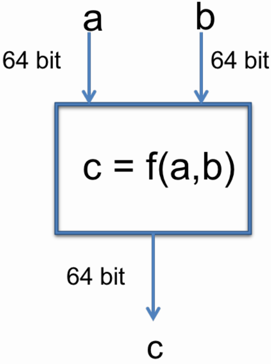
- Design is simulated with test vectors (values of *a* and *b*)
- Results from simulation compared to intended results



- Design is simulated with test vectors (values of *a* and *b*)
- Results from simulation compared to intended results
- Challenge: How do we generate test vectors?
  - 2^{128} combinations for a toy circuit



- Design is simulated with test vectors (values of *a* and *b*)
- Results from simulation compared to intended results
- Challenge: How do we generate test vectors?
  - 2^{128} combinations for a toy circuit
- Use constraints to represent *interesting* verification scenarios



Constraints

- Designers:
  - $a +_{64} 11 * 32b = 12$
  - $a <_{64} (b >> 4)$
- Past Experience:
  - $40 <_{64} 34 + a <_{64} 5050$
  - $120 <_{64} b <_{64} 230$
- Users:
  - $232 * 32a +_{64} b! = 1100$
  - $1020 <_{64} (b /_{64} 2) +_{64} a <_{64} 2200$

Test vectors: random solutions of constraints

Constrained Sampling

- Given:
  - Set of Constraints F over variables $X_1, X_2, \cdots X_n$
- Uniform Sampler

$$\forall y \in \mathsf{Sol}(F), \Pr[\text{y is output}] = \frac{1}{|\mathsf{Sol}(F)|}$$

- Almost-Uniform Sampler

$$\forall y \in \mathsf{Sol}(F), \frac{1}{(1+\varepsilon)|\mathsf{Sol}(F)|} \leq \Pr[\text{y is output}] \leq \frac{(1+\varepsilon)}{|\mathsf{Sol}(F)|}$$

Close Cousins: Counting and Sampling

- Approximate counting and almost-uniform sampling are inter-reducible (Jerrum, Valiant and Vazirani, 1986)
- Is the reduction efficient?

- Approximate counting and almost-uniform sampling are inter-reducible (Jerrum, Valiant and Vazirani, 1986)
- Is the reduction efficient?
  - Almost-uniform sampler (JVV) require linear number of approximate counting calls

Prior Work

Strong guarantees but poor scalability

- Polynomial calls to NP oracle (Bellare, Goldreich and Petrank,2000)
- BDD-based techniques (Yuan et al 1999, Yuan et al 2004, Kukula and Shiple 2000)
- Reduction to approximate counting (Jerrum, Valiant and Vazirani 1986)

Weak guarantees but impressive scalability

- Randomization in SAT solvers (Moskewicz 2001, Nadel 2011)
- MCMC-based approaches (Sinclair 1993, Jerrum and Sinclair 1996, Kitchen and Kuehlmann 2007,...)
- Belief Networks (Dechter 2002, Gogate and Dechter 2006)

Prior Work

Strong guarantees but poor scalability

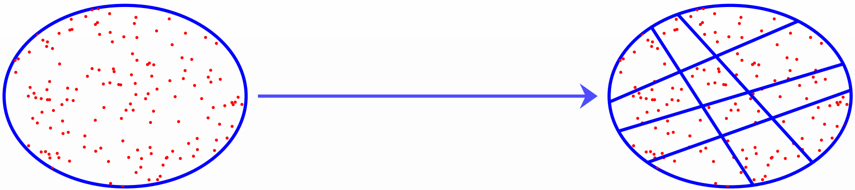
- Polynomial calls to NP oracle (Bellare, Goldreich and Petrank,2000)
- BDD-based techniques (Yuan et al 1999, Yuan et al 2004, Kukula and Shiple 2000)
- Reduction to approximate counting (Jerrum, Valiant and Vazirani 1986)

Weak guarantees but impressive scalability

- Randomization in SAT solvers (Moskewicz 2001, Nadel 2011)
- MCMC-based approaches (Sinclair 1993, Jerrum and Sinclair 1996, Kitchen and Kuehlmann 2007,...)
- Belief Networks (Dechter 2002, Gogate and Dechter 2006)

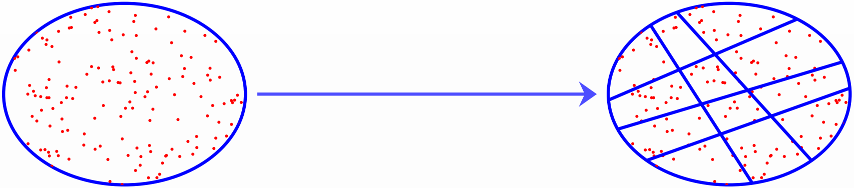
How to bridge this gap?

Key Ideas



- For right choice of number of cells, large number of cells are *small*
  - *almost all* the cells are *roughly* equal
- Check if a randomly picked cell is *small*
- If yes, pick a solution randomly from randomly picked cell

Key Ideas



- For right choice of number of cells, large number of cells are *small*
  - *almost all* the cells are *roughly* equal
- Check if a randomly picked cell is *small*
- If yes, pick a solution randomly from randomly picked cell

Challenge: How many cells?

How many cells?

- Desired Number of cells: $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\text{thresh}}$ ($m^* = \log \frac{|\mathsf{Sol}(F)|}{\text{thresh}}$)
  - ApproxMC(F, ε, δ) returns C such that
    $$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq C \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1 - \delta$$
  - $\tilde{m} = \log \frac{C}{\text{thresh}}$

How many cells?

- Desired Number of cells: $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$ ($m^* = \log \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$)
  - ApproxMC(F, ε, δ) returns C such that
$$\Pr\left[\frac{|\mathsf{Sol}(F)|}{1+\varepsilon} \leq C \leq |\mathsf{Sol}(F)|(1+\varepsilon)\right] \geq 1-\delta$$
  - $\tilde{m} = \log \frac{C}{\mathrm{thresh}}$
  - Check for $m = \tilde{m}-1, \tilde{m}, \tilde{m}+1$ if a randomly chosen cell is *small*

How many cells?

- Desired Number of cells: $2^{m^*} = \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$ ($m^* = \log \frac{|\mathsf{Sol}(F)|}{\mathrm{thresh}}$)
  - ApproxMC(F, ε, δ) returns C such that

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

  - $\tilde{m} = \log \frac{C}{\mathrm{thresh}}$
  - Check for $m = \tilde{m}-1, \tilde{m}, \tilde{m}+1$ if a randomly chosen cell is *small*
  - Not just a practical hack required non-trivial proof

(C**M**V, CAV13)
(C**M**V, DAC14)
(CF**M**SV, TACAS15)

Theoretical Guarantees

Theorem (Almost-Uniformity)

$$\forall y \in \mathsf{Sol}(F),\ \frac{1}{(1+\varepsilon)|\mathsf{Sol}(F)|} \leq \Pr[y \textit{ is output}] \leq \frac{1+\varepsilon}{|\mathsf{Sol}(F)|}$$

Theoretical Guarantees

Theorem (Almost-Uniformity)

$$\forall y \in \mathsf{Sol}(F),\ \frac{1}{(1+\varepsilon)|\mathsf{Sol}(F)|} \leq \Pr[y \textit{ is output}] \leq \frac{1+\varepsilon}{|\mathsf{Sol}(F)|}$$

Theorem (Query)

For a formula F over n variables UniGen makes **one call** *to approximate counter*

Theoretical Guarantees

Theorem (Almost-Uniformity)

$$\forall y \in \mathsf{Sol}(F),\ \frac{1}{(1+\varepsilon)|\mathsf{Sol}(F)|} \leq \Pr[y \textit{ is output}] \leq \frac{1+\varepsilon}{|\mathsf{Sol}(F)|}$$

Theorem (Query)

For a formula F over n variables UniGen makes **one call** *to approximate counter*

- *JVV (Jerrum, Valiant and Vazirani 1986) makes n calls*

Three Orders of Improvement

	Relative Runtime
SAT Solver	1
Desired Uniform Generator	10

(C**M**V, CAV13)
(C**M**V, DAC14)
(CF**M**SV, TACAS15)

Experiments over 200+ benchmarks

Three Orders of Improvement

	Relative Runtime
SAT Solver	1
Desired Uniform Generator	10
XORSample (2012 state of the art)	50000

(C**M**V, CAV13)
(C**M**V, DAC14)
(CF**M**SV, TACAS15)

Experiments over 200+ benchmarks

Three Orders of Improvement

	Relative Runtime
SAT Solver	1
Desired Uniform Generator	10
XORSample (2012 state of the art)	50000
UniGen (2015)	21

(CMV, CAV13)
(CMV, DAC14)
(CFMSV, TACAS15)

Experiments over 200+ benchmarks

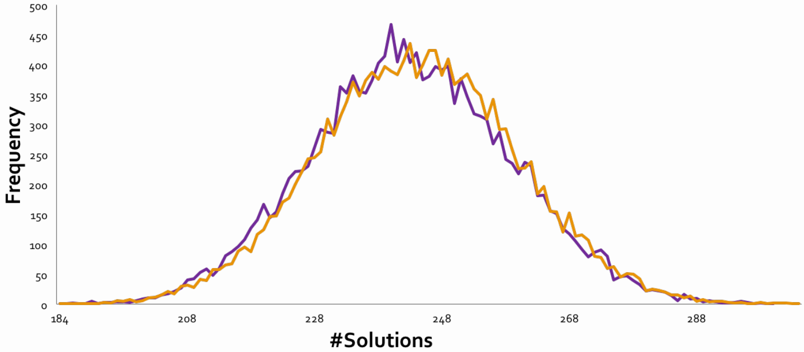
Three Orders of Improvement

	Relative Runtime
SAT Solver	1
Desired Uniform Generator	10
XORSample (2012 state of the art)	50000
UniGen (2015)	21

(CMV, CAV13)
(CMV, DAC14)
(CFMSV, TACAS15)

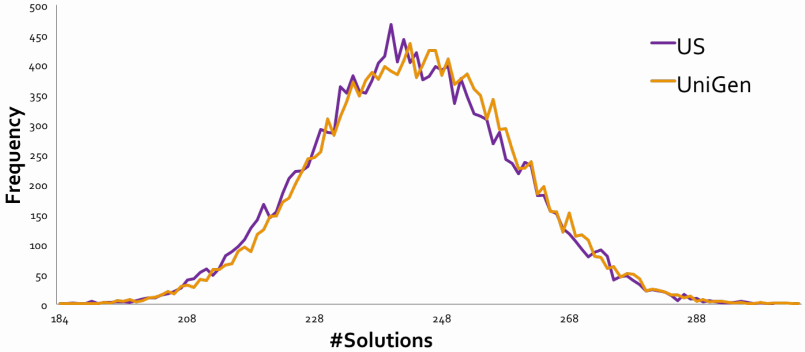
Experiments over 200+ benchmarks
Closer to technical transfer

Uniformity



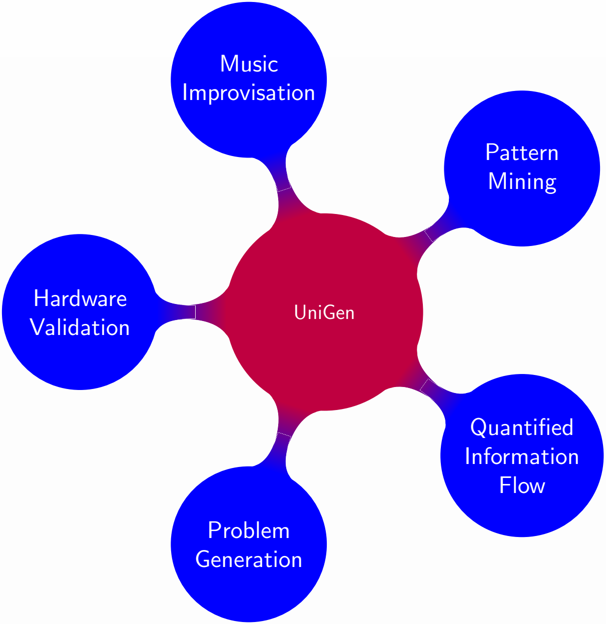
- Benchmark: case110.cnf; #var: 287; #clauses: 1263
- Total Runs: 4×10^6; Total Solutions : 16384

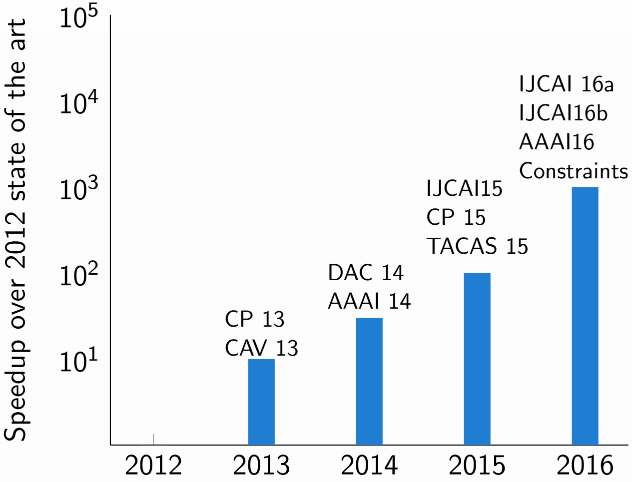
Statistically Indistinguishable

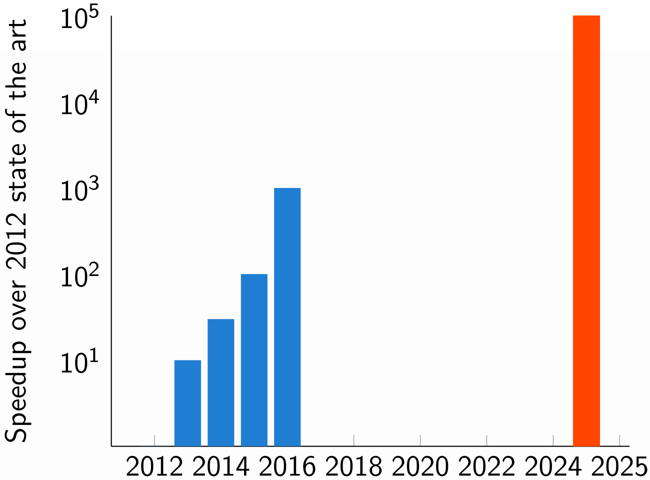


- Benchmark: case110.cnf; #var: 287; #clauses: 1263
- Total Runs: 4×10^6; Total Solutions : 16384

Usages of Open Source Tool: UniGen




Speedup over 2012 state of the art
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
Constraints
2012
2013
2014
2015
2016



Requires combinations of ideas from theory, statistics and systems

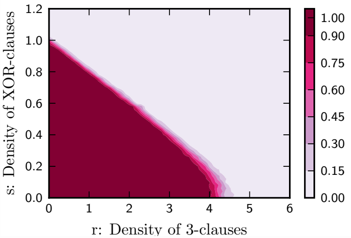
Mission 2025: Constrained Counting and Sampling Revolution

- Tighter integration between solvers and algorithms

Mission 2025: Constrained Counting and Sampling Revolution

- Tighter integration between solvers and algorithms
- Exploring solution space structure of CNF+XOR formulas

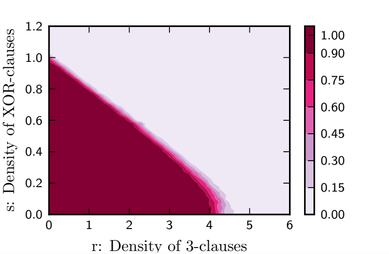
(D**M**V, IJCAI16),



Mission 2025: Constrained Counting and Sampling Revolution

- Tighter integration between solvers and algorithms
- Exploring solution space structure of CNF+XOR formulas

(D**M**V, IJCAI16),



- Beyond Boolean variables – without *bit blasting*

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Civil Engineering Reliability for Los Angeles Transmission Grid

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Civil Engineering Reliability for Los Angeles Transmission Grid

Privacy Leakage Measurement for C++ program with 100 lines

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Civil Engineering	Reliability for Los Angeles Transmission Grid
Privacy	Leakage Measurement for C++ program with 100 lines
Artificial Intelligence	Inference for Bayesian network with 1K nodes

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Civil Engineering Reliability for Los Angeles Transmission Grid

Privacy Leakage Measurement for C++ program with 100 lines

Artificial Intelligence Inference for Bayesian network with 1K nodes

The Potential of Hashing-based Framework

Machine Learning Probabilistic programming

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Civil Engineering Reliability for Los Angeles Transmission Grid

Privacy Leakage Measurement for C++ program with 100 lines

Artificial Intelligence Inference for Bayesian network with 1K nodes

The Potential of Hashing-based Framework

Machine Learning Probabilistic programming

Theory Classification of Approximate counting complexity

Mission 2025: Constrained Counting and Sampling Revolution

Challenge Problems

Civil Engineering	Reliability for Los Angeles Transmission Grid
Privacy	Leakage Measurement for C++ program with 100 lines
Artificial Intelligence	Inference for Bayesian network with 1K nodes

The Potential of Hashing-based Framework

Machine Learning	Probabilistic programming
Theory	Classification of Approximate counting complexity
Databases	Streaming algorithms

Collaborators



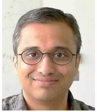


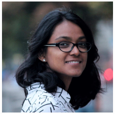








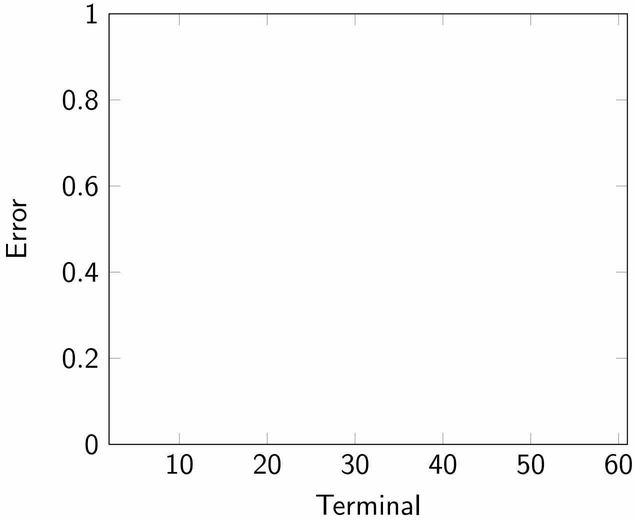




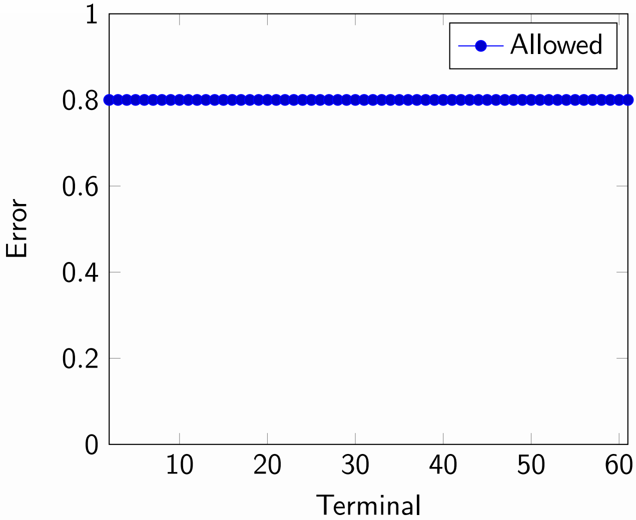
Part I

Backup

Highly Accurate Estimates



Highly Accurate Estimates



Highly Accurate Estimates