Word-Level Hashing Approach to Approximate Probabilistic Inference

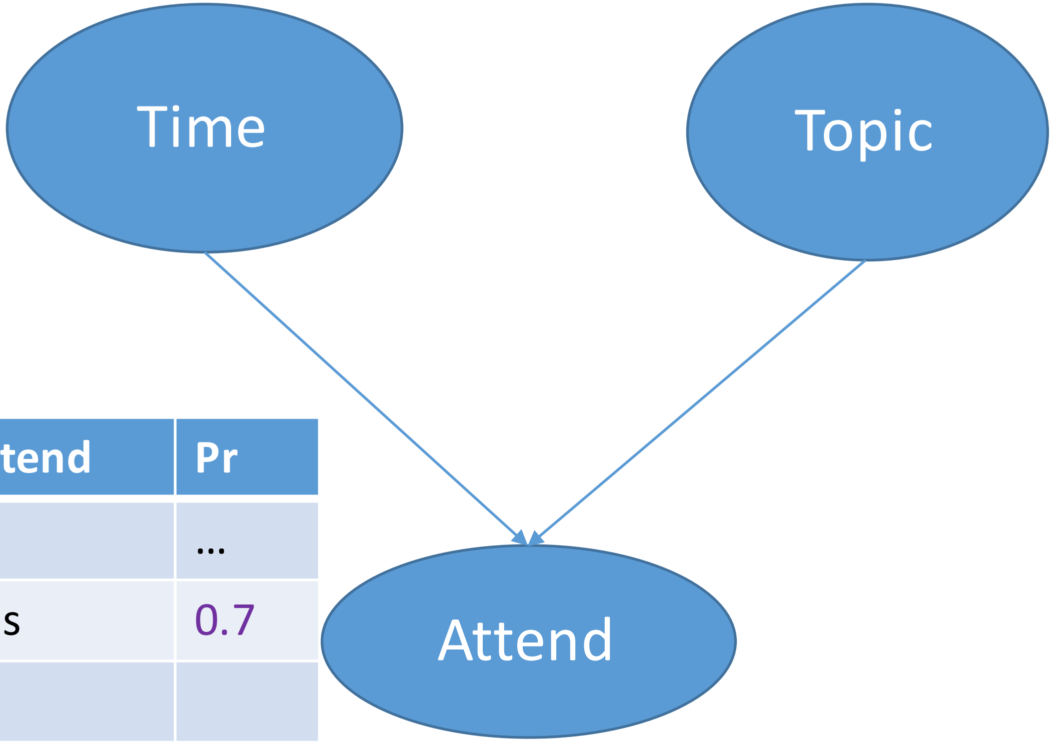
Kuldeep S. Meel

Rice University

Joint work with Supratik Chakraborty (IITB), Rakesh Mistry (IITB), and Moshe Y. Vardi (Rice)

Graphical Models

Morning	0.35
Afternoon	0.2
Evening	0.45



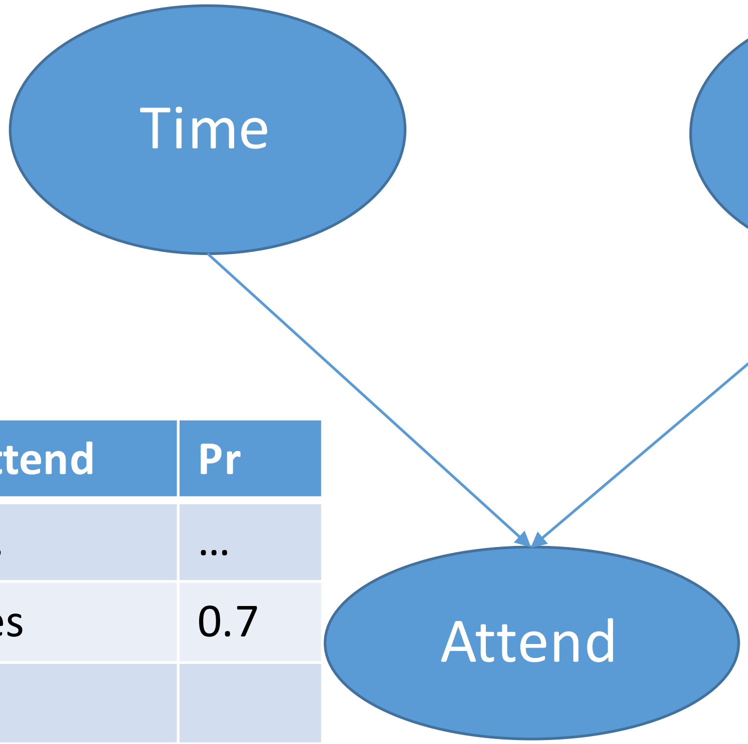
NLP	0.25
GM	0.65
Other	0.1

Time	Topic	Attend	Pr
….	….	….	…
Afternoon	GM	Yes	0.7

Pr[Attend = Yes ∩ Topic=GM ∩ Time=Morning] = 0.7*0.65*0.35

Probabilistic Inference

Morning	0.35
Afternoon	0.2
Evening	0.45



NLP	0.25
GM	0.65
Other	0.1

Time	Topic	Attend	Pr
….	….	….	…
Afternoon	GM	Yes	0.7

Pr [Attend = Yes | Topic=GM]

Event — Attend = Yes

Evidence — Topic=GM

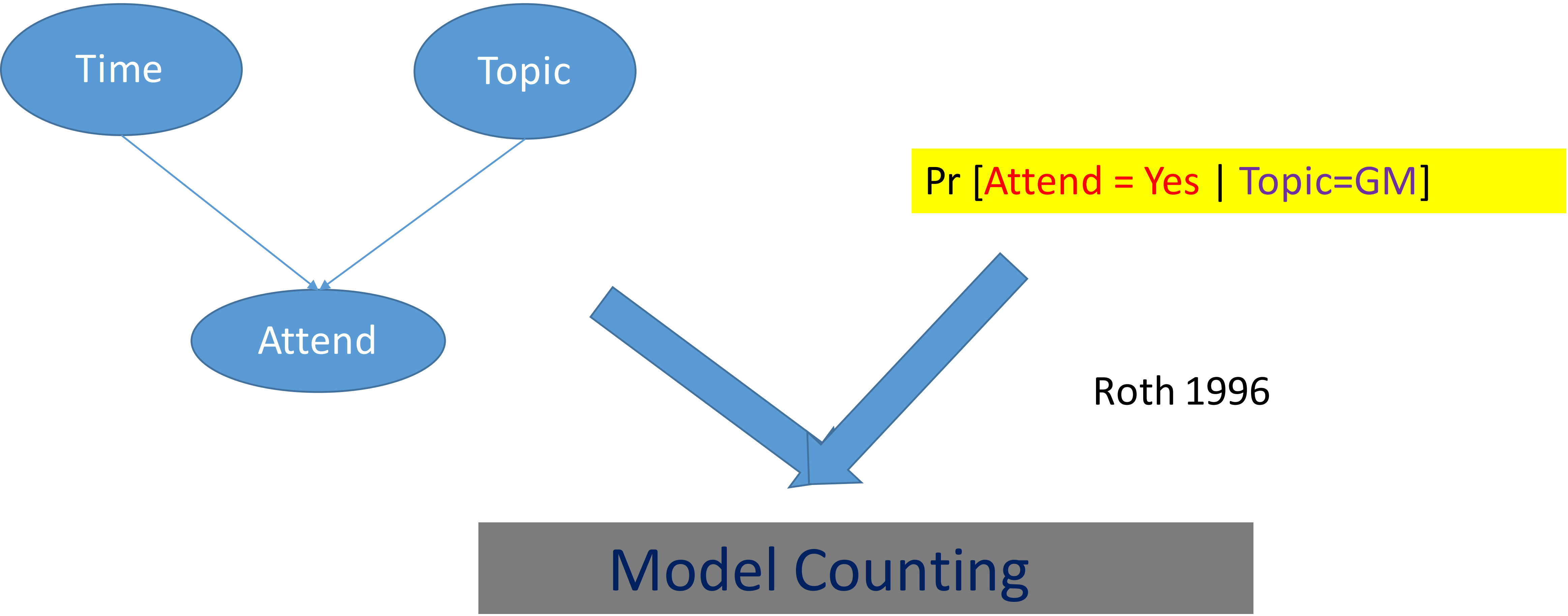
Probabilistic Inference

- Exact computation is intractable (#P-complete)
- Approximate techniques:
  - Markov Chain Monte Carlo Methods
  - Variational Approximation
  - Interval Propagation
  - Randomization in combinatorial reasoning tools

Drawback:

Either Performance or Theoretical Guarantees but Not Both

Reduction to Model Counting



Model Counting

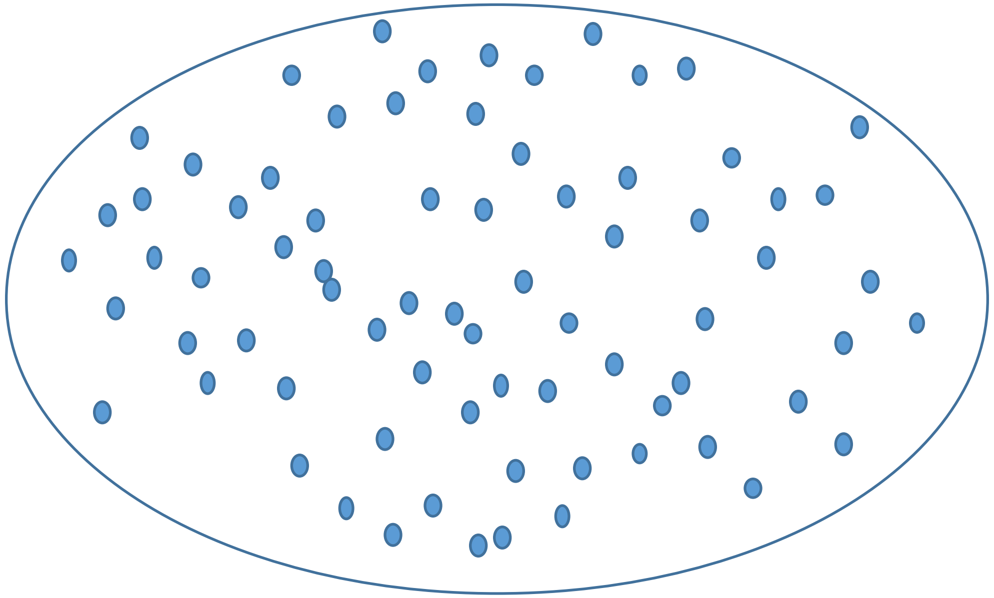
- Given a SAT formula F
- R_F: Set of all solutions of F
- Problem (#SAT): Estimate the number of solutions of F (#F) i.e., what is the cardinality of R_F?
- E.g., F = (a v b)
- $R_F = \{(0,1), (1,0), (1,1)\}$
- The number of solutions (#F) = 3

#P: The class of counting problems for decision problems in NP!

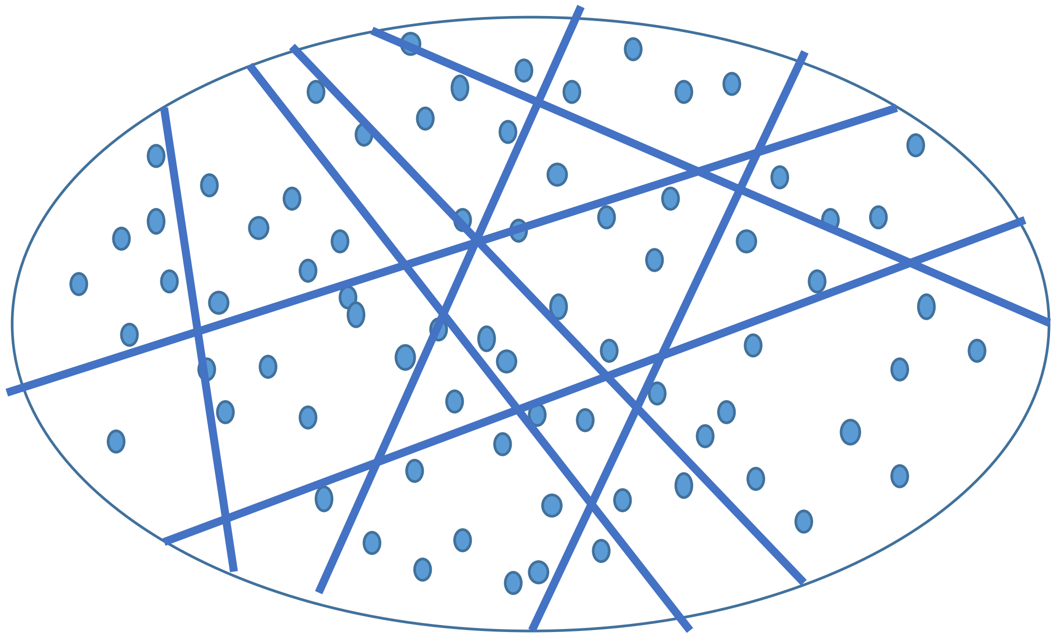
Long History of Work

- Proved #P complete (Valiant 1977)
- Approximate variant: introduced by Stockmeyer (1983)
- Uniform sampling is inter-reducible to approximate counting (Jerrum, Valiant and Vazirani 1986)
- FPRAS for approximate #DNF (Karp, Luby 1985)
- No practical techniques for CNF

Partitioning into equal “small” cells

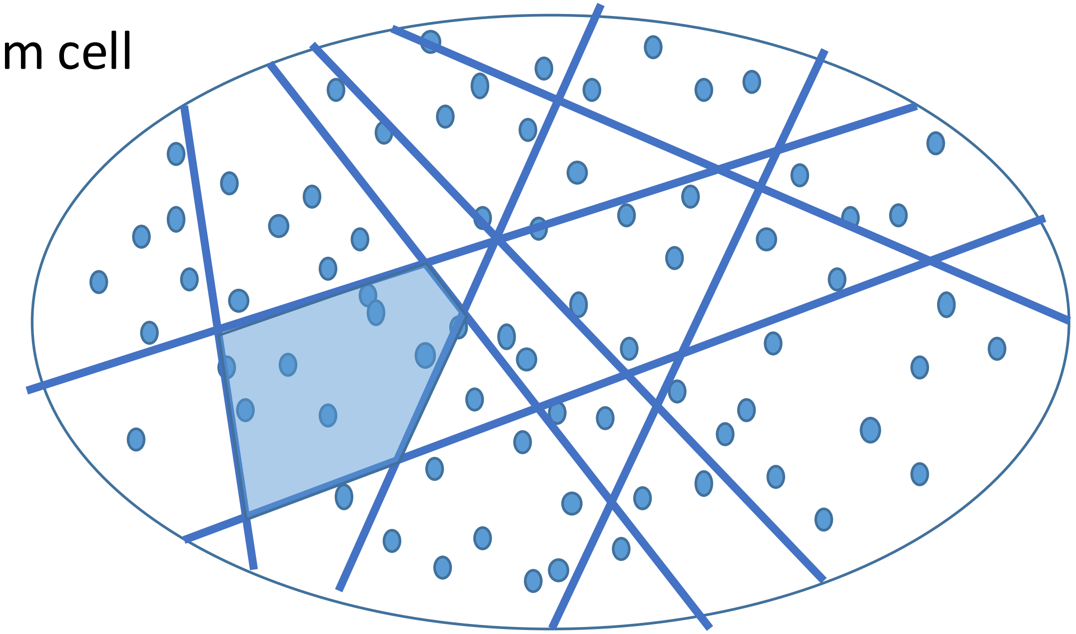


Partitioning into equal “small” cells



Partitioning into equal “small” cells

Pick a random cell



Estimate = # of models in cell * # of cells

How to Partition?

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

Universal Hashing
[Carter-Wegman 1979]

XOR-Based Hashing

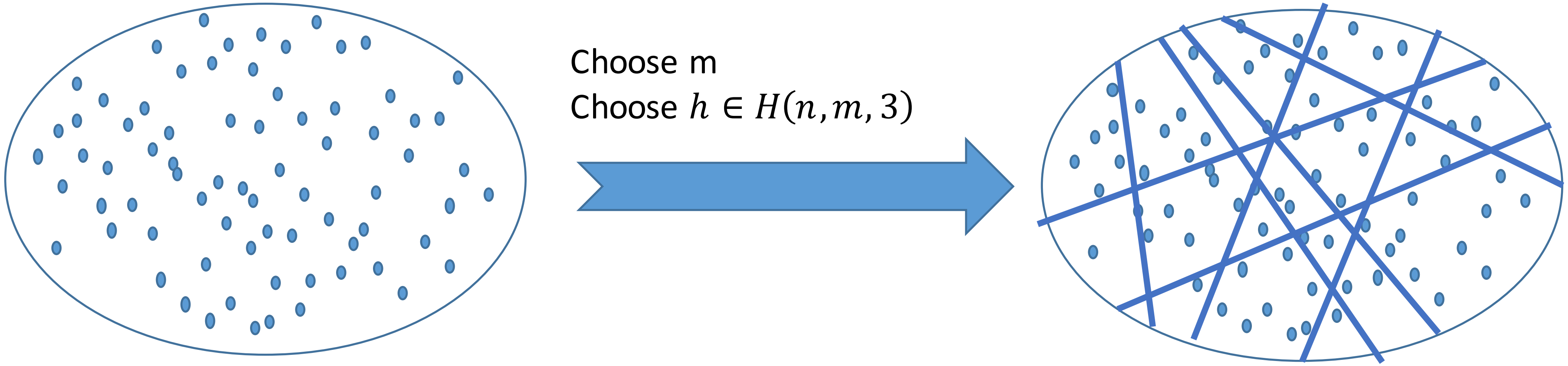
- Partition 2^n space into 2^m cells
- Variables: $X_1, X_2, X_3, \ldots, X_n$
- Pick every variable with prob. ½ ,XOR them and add 0/1 with prob. ½
- $X_1+X_3+X_6+\ldots X_{n-1} + 0$
- To construct $h: \{0,1\}^n \rightarrow \{0,1\}^m$, choose m random XORs
- $\alpha \in \{0,1\}^m \rightarrow$ Set every XOR equation to 0 or 1 randomly
- The cell: $F \wedge$ XOR (CNF+XOR)

Size of cell

- Too large => Hard to enumerate
- Too small => Variance can be very high

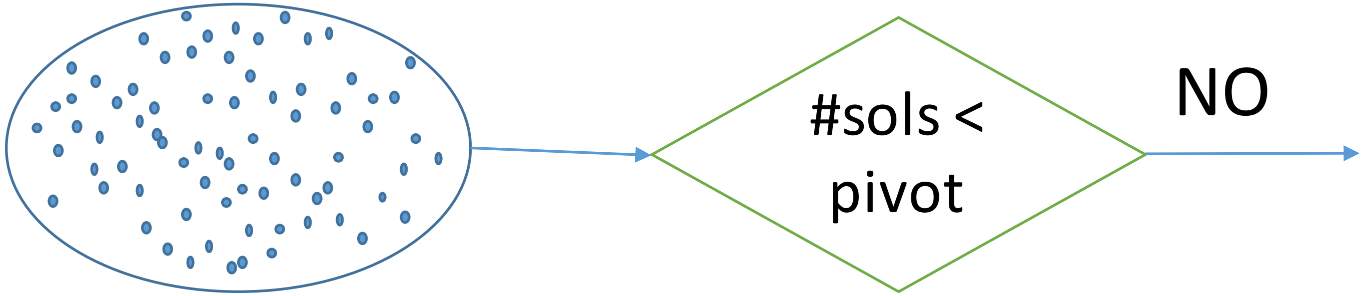
$$\text{pivot} = 5(1 + 1/\varepsilon)^2$$

PAC Counter: ApproxMC(F,ε, δ)

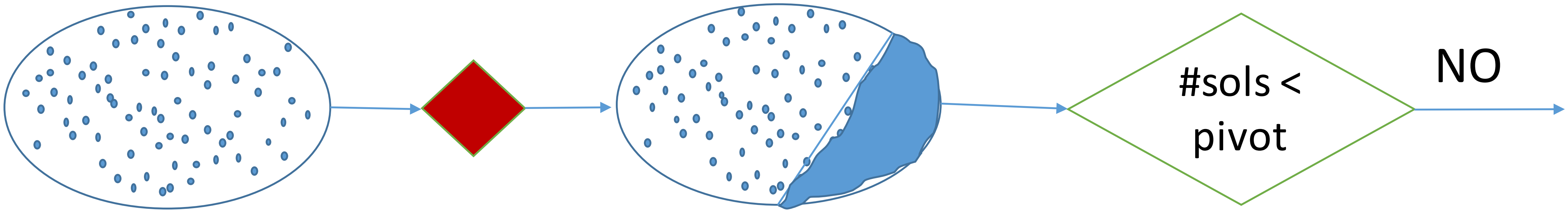


- For right choice of m, large number of cells are “small”
    - “almost all” the cells are “roughly” equal
- Check if a randomly picked cell is “small”
- If yes, then estimate = # of solutions in cell * 2^m

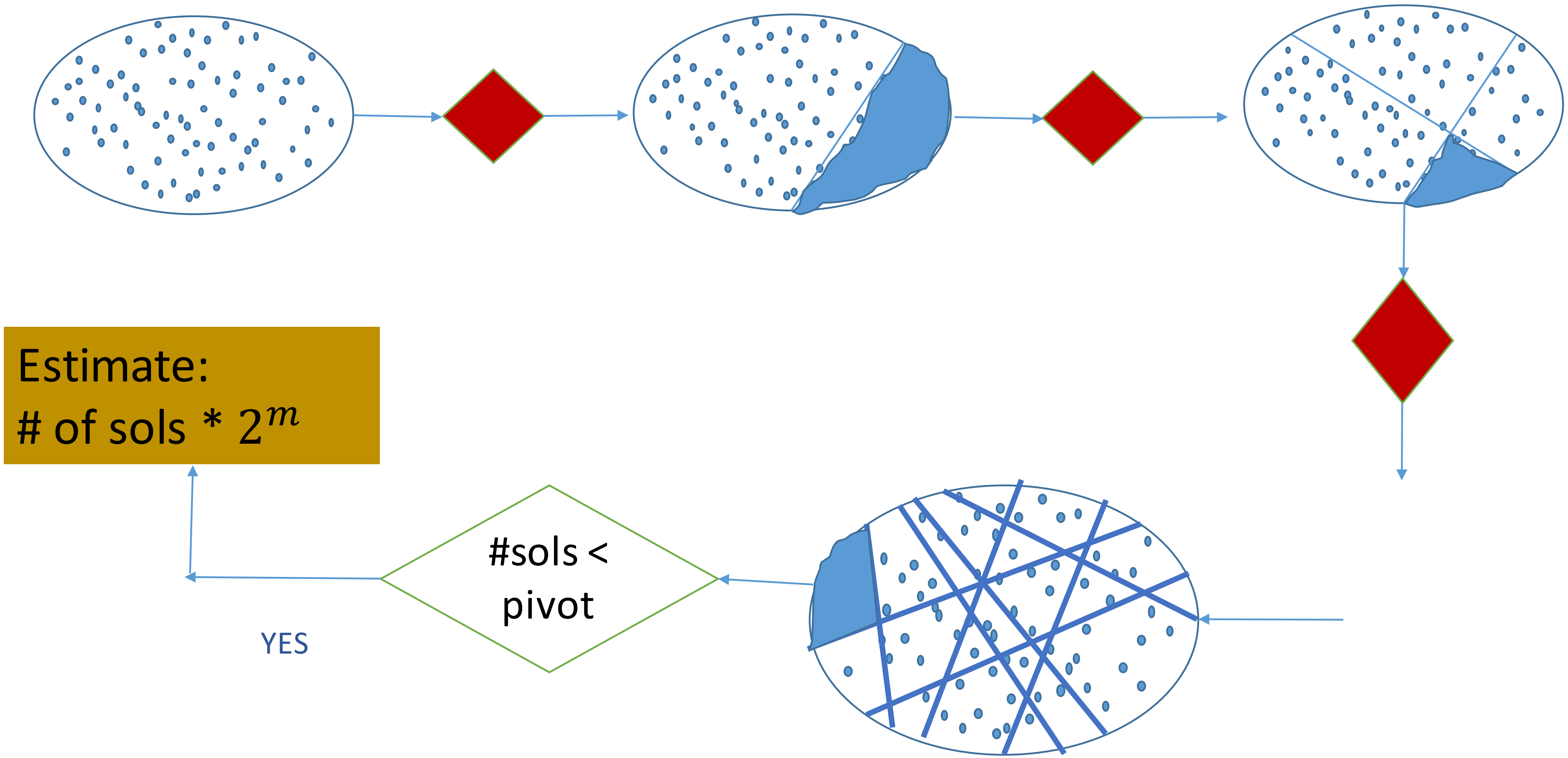
ApproxMC(F,ε, δ)



ApproxMC(F,ε, δ)



ApproxMC(F,ε, δ)



ApproxMC(F,ε, δ)

Key Lemmas

Let $m^* = \log|R_F| - \log pivot$

Lemma 1: The algorithm terminates with $m \in [m^* - 1, m^*]$ with high probability

Lemma 2: The estimate from a randomly picked cell for $m \in [m^* - 1, m^*]$ is correct with high probability

Approximate Model Counting

- Approximate Model Counting

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

- Hashing-based Approaches
  - CAV 2013
  - CP 2013
  - UAI 2013
  - NIPS 2013
  - DAC 2014
  - ICML 2014
  - AAAI 2014
  - TACAS 2015
  - IJCAI 2015
  - ICML 2015
  - UAI 2015
  - AAAI 2016
  - AISTATS 2016

Bit-level reasoning

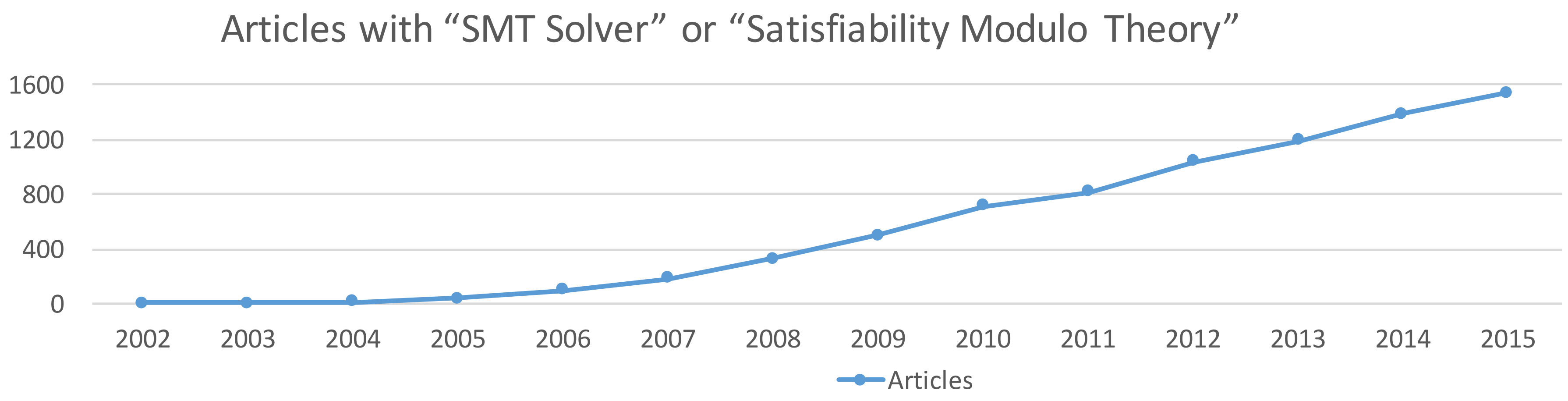
- XOR-based (mod 2) hash functions in **all** prior works
- Variables in Graphical Models are not binary
- Approach: Perform "bit-blasting"
  - $Dom(X) = \{0, 1, 2, 3\}$
  - X can be represented using two bits (y_1, y_2) such that $X = y_1 y_2$
  - XOR constraints over y_i variables

- Require solvers to perform bit-level reasoning

Word-level Revolution

- Development of SMT Solvers to reason directly at the level of “words”, i.e. variables
  - No need for “bit-blasting”
- The biggest advance in formal methods in last 25 years

  [John Rushby, 2011]



Our Contributions

- H_{SMT}: Efficient word-level Hash Function

- SMTApproxMC: Efficient word-level counter

Theory: QF-BV

Towards Efficient word-level Hashing

- Lifting hashing from (mod 2) to (mod 2^k) constraints
  - k: largest "bit-width"

- Linear inequality constraints
  - $h_1 \coloneqq a_1x_1 + a_2x_2 + \cdots + a_nx_n + b$
  - $a_1, a_2, \ldots . a_n, b$, are randomly chosen from 0 to 2^k-1
  - $\alpha_1 :=$ "$< 2^{k-1}$" or "$\geq 2^{k-1}$"

Theoretical Guarantees: 2-universal

- $h_1 := (a_1x_1 + a_2x_2 + \cdots + a_nx_n + b)$
- $\alpha_1 := \ < 2^{k-1}$
- $\sigma_1 = \{x_1 = v_1, x_2 = v_2 \ldots . x_n = v_n\}$
- $\Pr[\sigma_1 \vDash (h_1 = \alpha_1)]$
  - Transform σ_1 to (0,0….0)
  - $\Pr[(0,0,\ldots.0) \vDash (h_1 = \alpha_1)] = \Pr[\mathrm{b} < 2^{k-1}] = \frac{1}{2}$
- $\Pr[\sigma_2 \vDash (h_1 = \alpha_1) \mid \sigma_1 \vDash (h_1 = \alpha_1)]$
  - Transform σ_1 to (0,0….0)
  - Transform σ_2 to (1,0…..0)
  - $\Pr[\sigma_2 \vDash (h_1 = \alpha_1) \mid \sigma_1 \vDash (h_1 = \alpha_1)] = \Pr[a_1 + b < 2^{k-1} \mid b < 2^{k-1}] = \frac{1}{2}$

Word-Level Counter

1. $F' = F$
2. for i= 1 to k:
3. If (| $R_{F'}$ | > pivot):
4. $F' = F \wedge \{(a_1x_1 + a_2x_2 + \cdots . a_nx_n + b$ `` $\geq$ '' or " $<$ " 2^{k-1})}
5. Else:
6. If (| $R_{F'}$ |==0):
7. Return ⊥
8. Return | $R_{F'}$| $*$ 2^{i}

Diagnosis

- Look for hash functions that are polynomial to solve by themselves

Towards Efficient word-level Hashing

- Lifting hashing from (mod 2) to (mod p) constraints
  - **p**: smallest prime greater than domain of variables (2^k)

- Linear equality (mod p) constraints to partition into p cells
  - $|Dom(x_i)| \leq 2^k$
  - $h_1 \coloneqq (a_1x_1 + a_2x_2 + \cdots + a_nx_n + b)$ (mod p)
  - $a_1, a_2, \ldots. a_n, b,$ are randomly chosen from 0 to p-1

Theoretical Guarantees: 2-universal

- $h_1 \coloneqq (a_1x_1 + a_2x_2 + \cdots + a_nx_n + b)$ (mod p)
- $\sigma_1 = \{x_1 = v_1, x_2 = v_2 \ldots. x_n = v_n\}$
- $\Pr[\sigma_1 \vDash (h_1 = \alpha_1)]$
  - Transform σ_1 to (0,0….0)
  - $\Pr[(0,0,\ldots.0) \vDash (h_1 = \alpha_1)] = \Pr[b == 0] = \frac{1}{p}$
- $\Pr[\sigma_2 \vDash (h_1 = \alpha_1) \mid \sigma_1 \vDash (h_1 = \alpha_1)]$
  - Transform σ_1 to (0,0….0)
  - Transform σ_2 to (1,0…..0)
  - $\Pr[\sigma_2 \vDash (h_1 = \alpha_1) \mid \sigma_1 \vDash (h_1 = \alpha_1)] = \Pr[a_1 = 1] = \frac{1}{p}$

Word-Level Counter

1. $F' = F$
2. for i= 1 to k:
3. If ($|R_{F'}| >$ pivot):
4. $F' = F \wedge \{(a_1x_1 + a_2x_2 + \cdots . a_nx_n + b = \alpha)$ mod p$\}$
5. Else:
6. If ($|R_{F'}|$==0):
7. Return $\perp$
8. Return $|R_{F'}| * \mathrm{p}^{\mathrm{i}}$

Diagnosis

- Number of cells (N) = p^c
  - C: Number of Linear Constraints
- N is too small → Number of solutions is too large
- N is too large → Number of solutions is very small (Avg $<$ 0)

- Need finer control over number of cells

SMTApproxMC(F, ε, δ)

p_i = smallest prime greater than 2^{k+1-2^i}

1. $F' = F$; i = 0
2. For j = 1 to k:
3. If ($|R_{F'}|$ > pivot):
4. $F' = F \wedge \{(a_1 x_1 + a_2 x_2 + \cdots . a_n x_n + b = \alpha) \bmod p_i\}$
5. Else:
6. If ($|R_{F'}|$==0 & p_i >2):
7. F' = Pop out last constraint; i++
8. $F' = F \wedge \{(a_1 x_1 + a_2 x_2 + \cdots . a_n x_n + b = \alpha) \bmod p_i\}$
9. Return $|R_{F'}| * N$

$\mathrm{H_{SMT}}$: Efficient word-level Hash Function

- Use different primes to control the number of cells

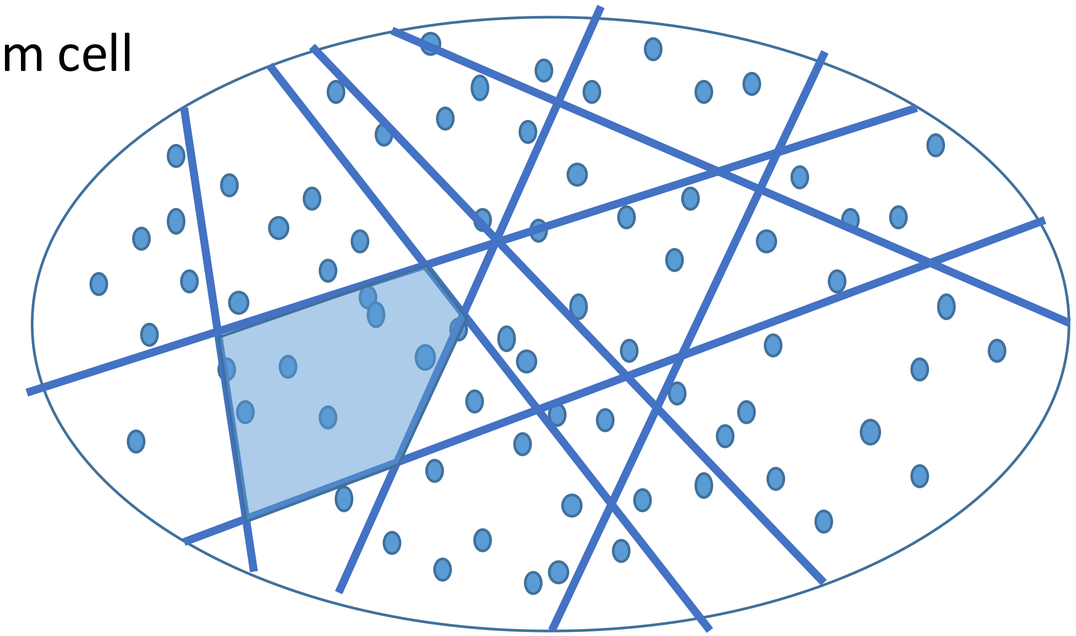
- Choose appropriate N and express as product of *preferred* primes, i.e. $N = p_1^{c_1} p_2^{c_2} p_3^{c_3} \ldots\ldots p_n^{c_n}$

- $\mathrm{H_{SMT}}$:
  - c_1 (mod p_1) constraints
  - c_2 (mod p_2) constraints
  - ....

- $\mathrm{H_{SMT}}$ satisfies guarantees of 2-universality

SMTApproxMC

Pick a random cell



Estimate = # of models in cell * # of cells

Theoretical Guarantees

- F: Formula over bounded domain variables;
- R_F : Solution Space of F
- SMTApproxMC

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

- Polynomial in $F, \frac{1}{\varepsilon}, \log\left(\frac{1}{\delta}\right)$ relative to word-level oracle

Experimental Evaluations

- Over 150 benchmarks from:
  - Ising Models
  - ISCAS89 Circuits
  - Program Synthesis
- Comparison with state of the art tool: CDM
  - Based on Chistikov, Dimitrova, and Majumdar 2015
    - Similar to Ermon et al, Chakraborty et al, Belle et al, etc..
    - Uses XOR-based hash functions (bit level!)
- Objectives:
  - Quality of estimates
  - Runtime performance comparison

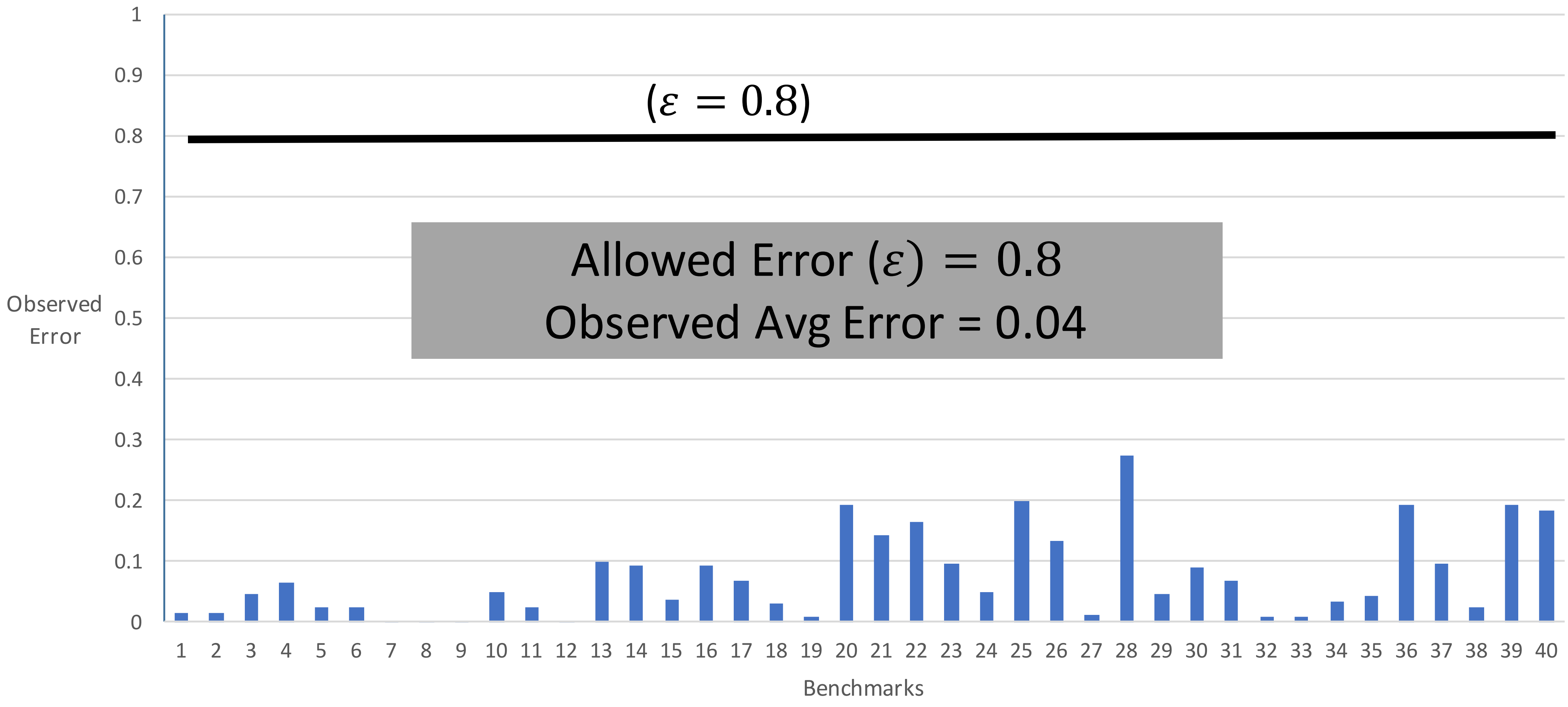
Quality Comparison

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

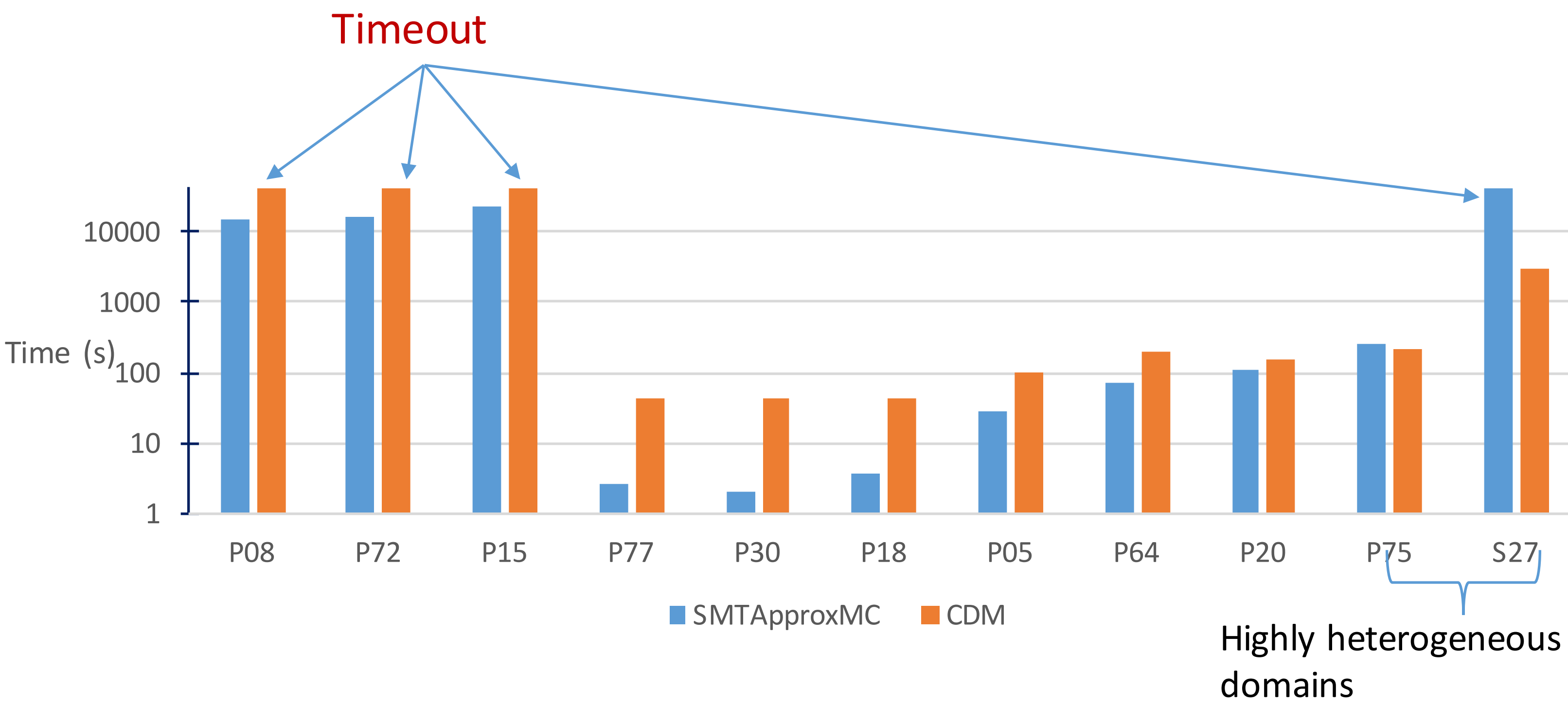
- Experiments with $\varepsilon = 0.8$ $\delta = 0.1$

- Observed $\varepsilon = \max\{\frac{|R_F|}{\text{SMTApproxMC}(F,\varepsilon,\delta)} - 1, \frac{\text{SMTApproxMC}(F,\varepsilon,\delta)}{|R_F|} - 1\}$

Quality Comparison



Runtime Performance Comparison



SMTApproxMC is 2-10 times faster than CDM

Future Work

SMT + Mod p

- For SAT: CNF + XOR
- CryptoMiniSAT has been solver of choice
  - Gaussian elimination for added XOR constraints
- SMT Solver with Gaussian elimination for added Linear equality constraints
- Preferred primes dependent on SMT solver's architecture?

SMT Sampling

- Sampling is inter-reducible to counting (JVV 1986)
  - Algorithm is highly impractical (linear number of calls to approx counter)

- Hashing-based framework for sampling
  - UniGen (Chakraborty,M.,Vardi, 2013)
  - Requires 3-universal guarantees

- H_{SMT} can provide only 2-universal guarantees
  - Design efficient algorithms with only 2-universal requirement?

For tools/papers: www.kuldeepmeel.com