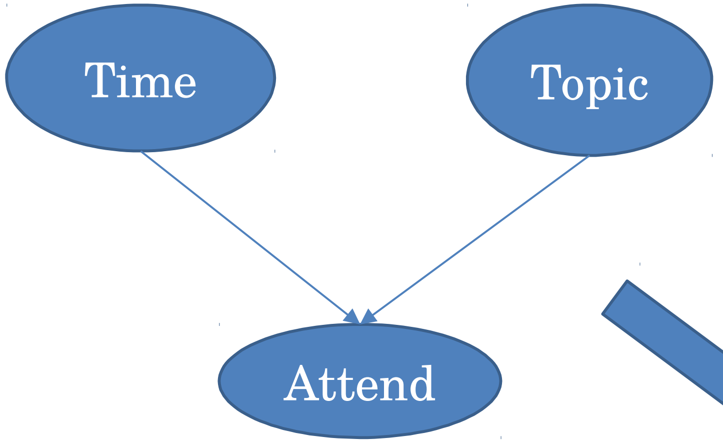
Algorithmic Improvements in Approximate Counting for Probabilistic Inference: From Linear to Logarithmic SAT Calls

Supratik Chakraborty[1], **Kuldeep S. Meel[2]**, Moshe Y. Vardi[2]

[1] Indian Institute of Technology (IIT) – Bombay

[2] Department of Computer Science, Rice University

From Probabilistic Inference to Model Counting



Event ↓ Evidence ↓

Pr [Attend = Yes | Topic=GM]

Roth, 1996

Model Counting

Model counting as the “assembly language” for probabilistic inference

Approximate Model Counting

- Approximate Model Counting

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

- Hashing-based Approaches

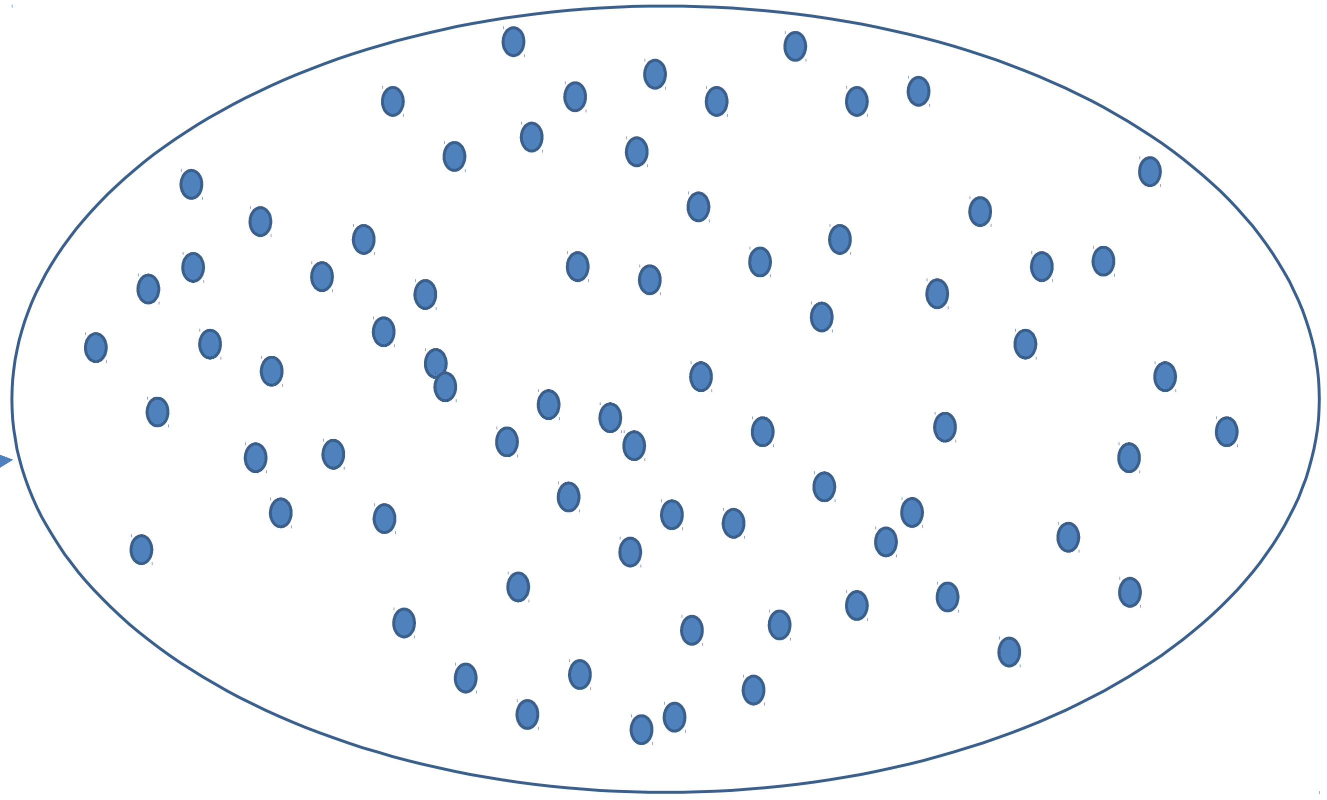
- Hashing-based Approaches

- 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
- ICML 2016

Counting Dots

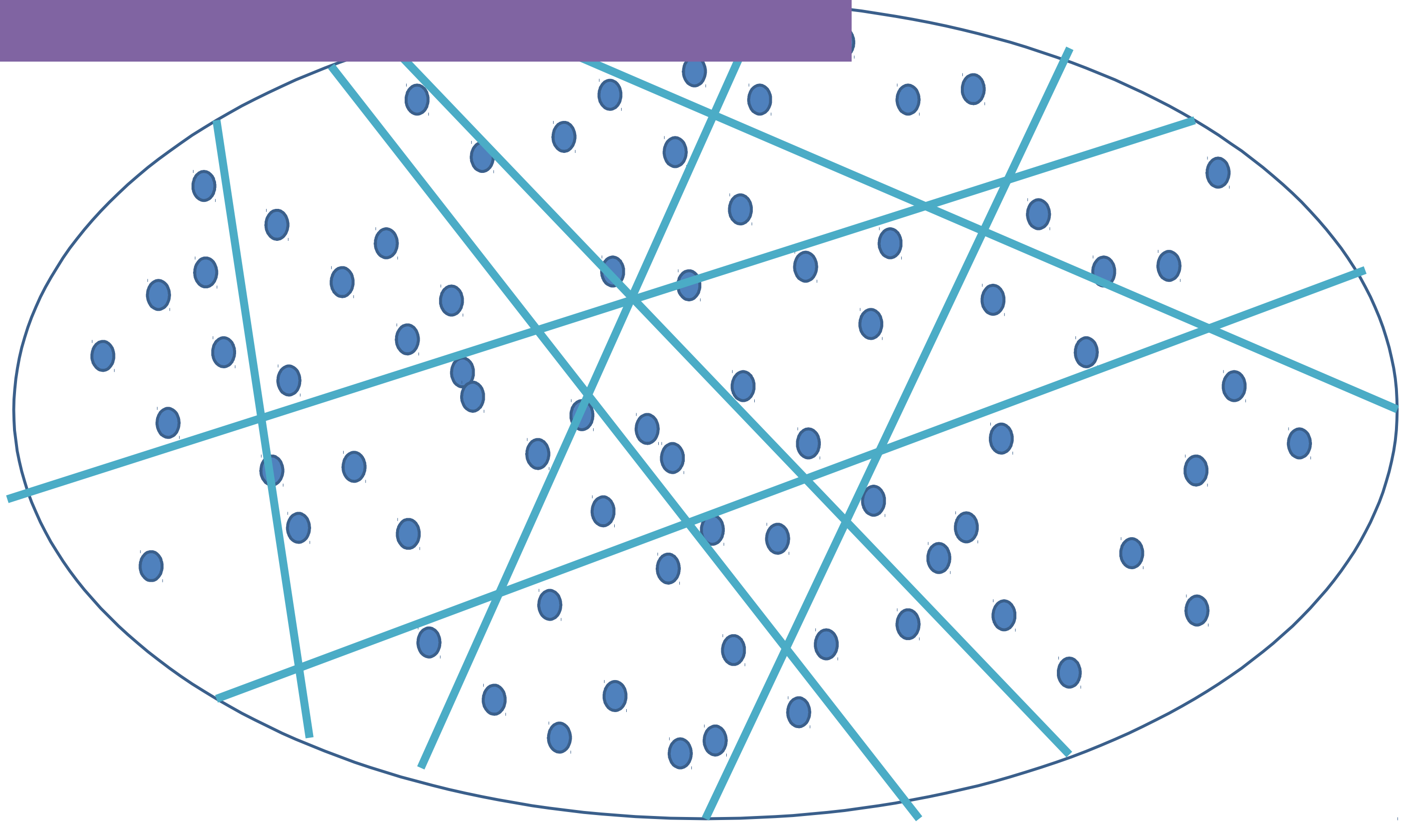
- Solution to constraints

$\{0,1\}^n$



Partitioning into equal "small" cells

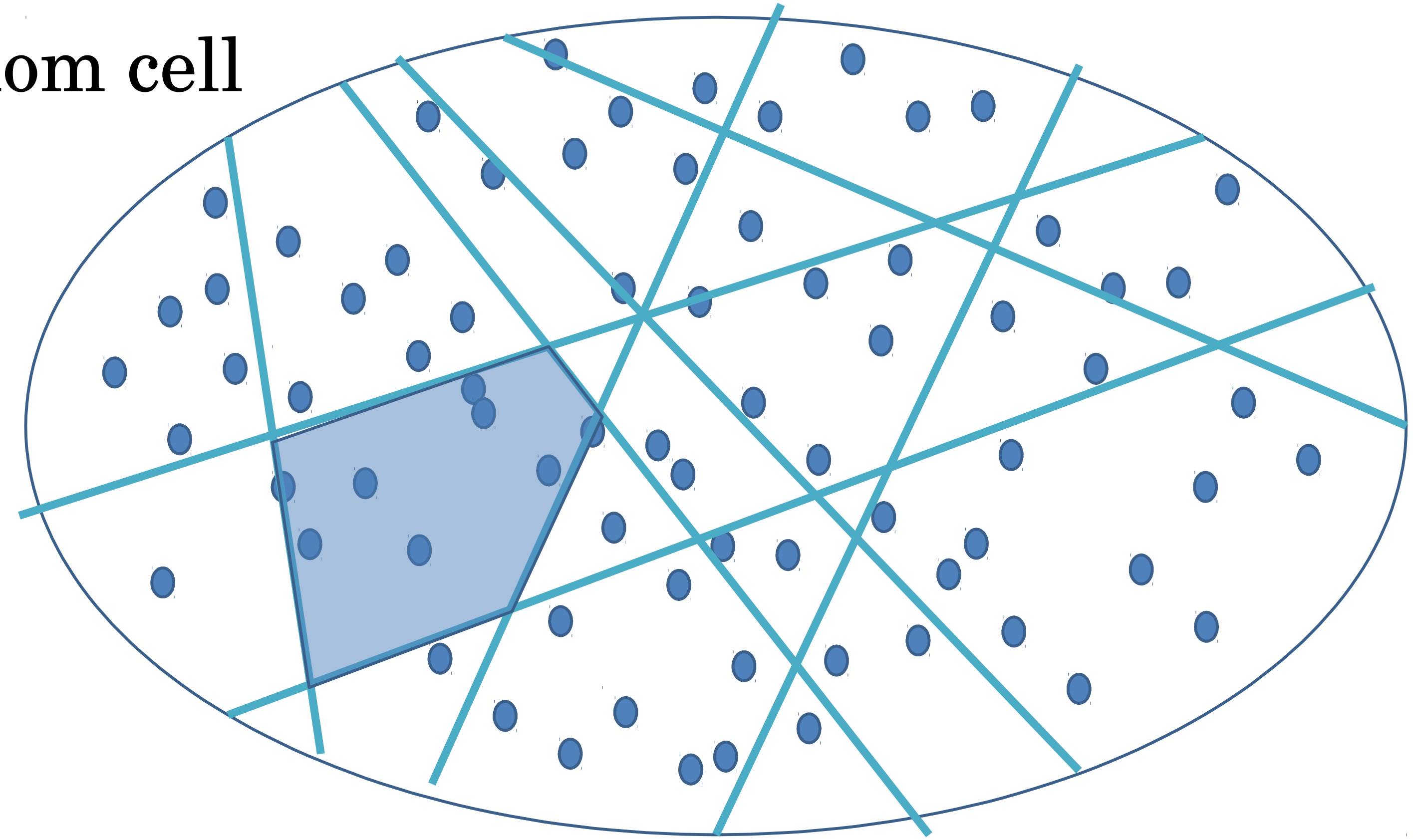
"small" cell: # of solutions < pivot
pivot = $5(1+1/\varepsilon)^2$
pivot = 5(1+



2-universal hashing (Carter-Wegman 1977)

Partitioning into equal "small" cells

Pick a random cell



Estimate = # of solutions (dots) in cell * # of cells

The secret sauce

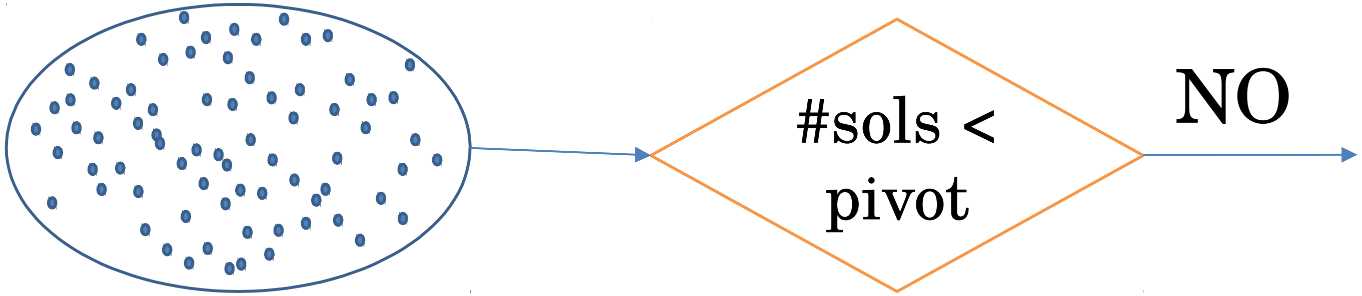
- How many cells should we partition into?

Ideally, $\#\text{of cells} = \frac{|R_F|}{\text{pivot}}$

But we do not know $|R_F|$!

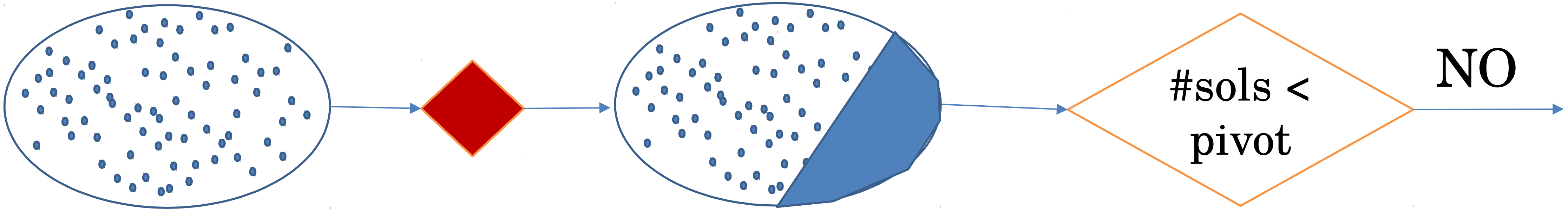
ApproxMC(F, ε, δ) (CMV CP 2013)

ApproxMC(F, (CMV CP 2013)



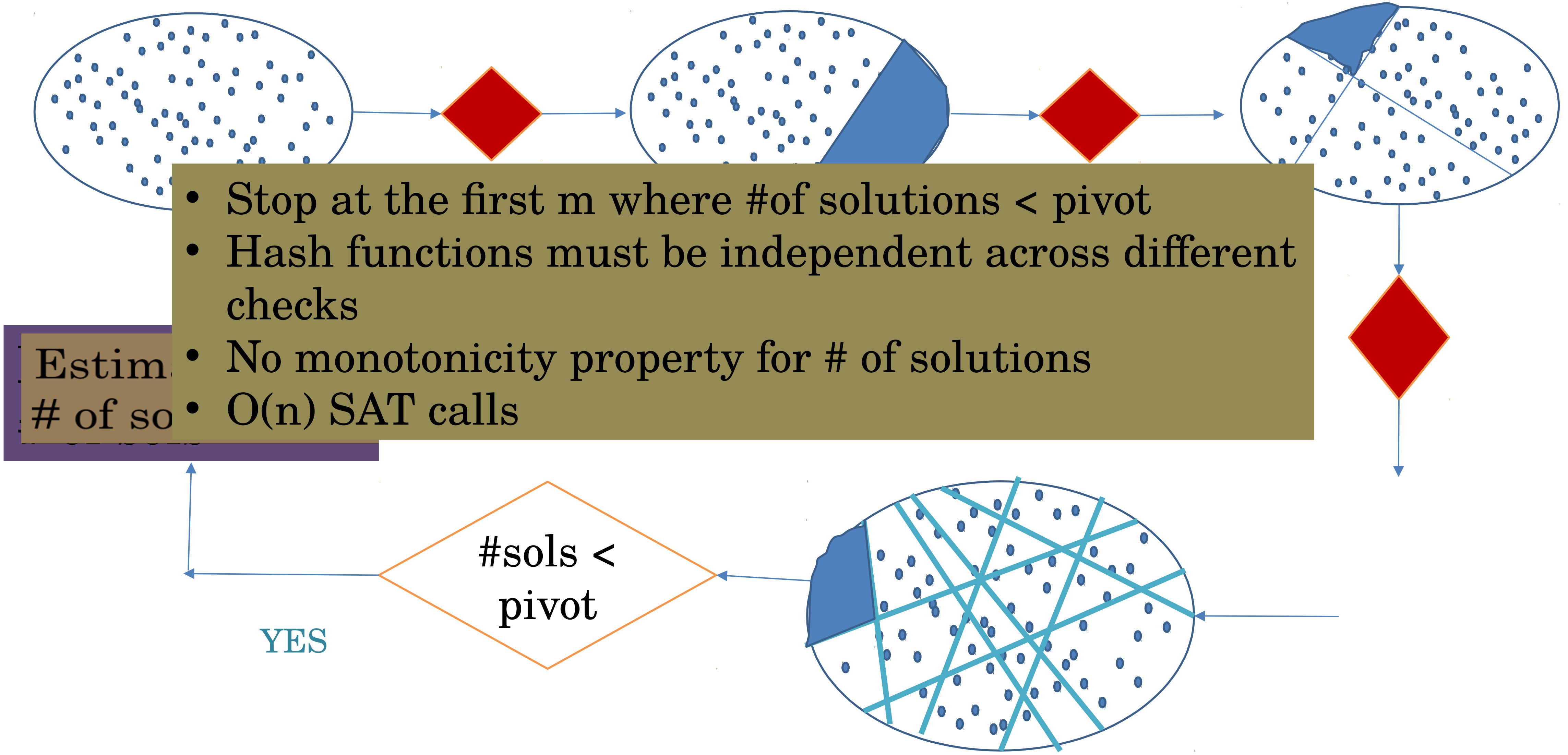
ApproxMC(F, ε, δ)

ApproxMC(F,



ApproxMC(F, ε, δ)

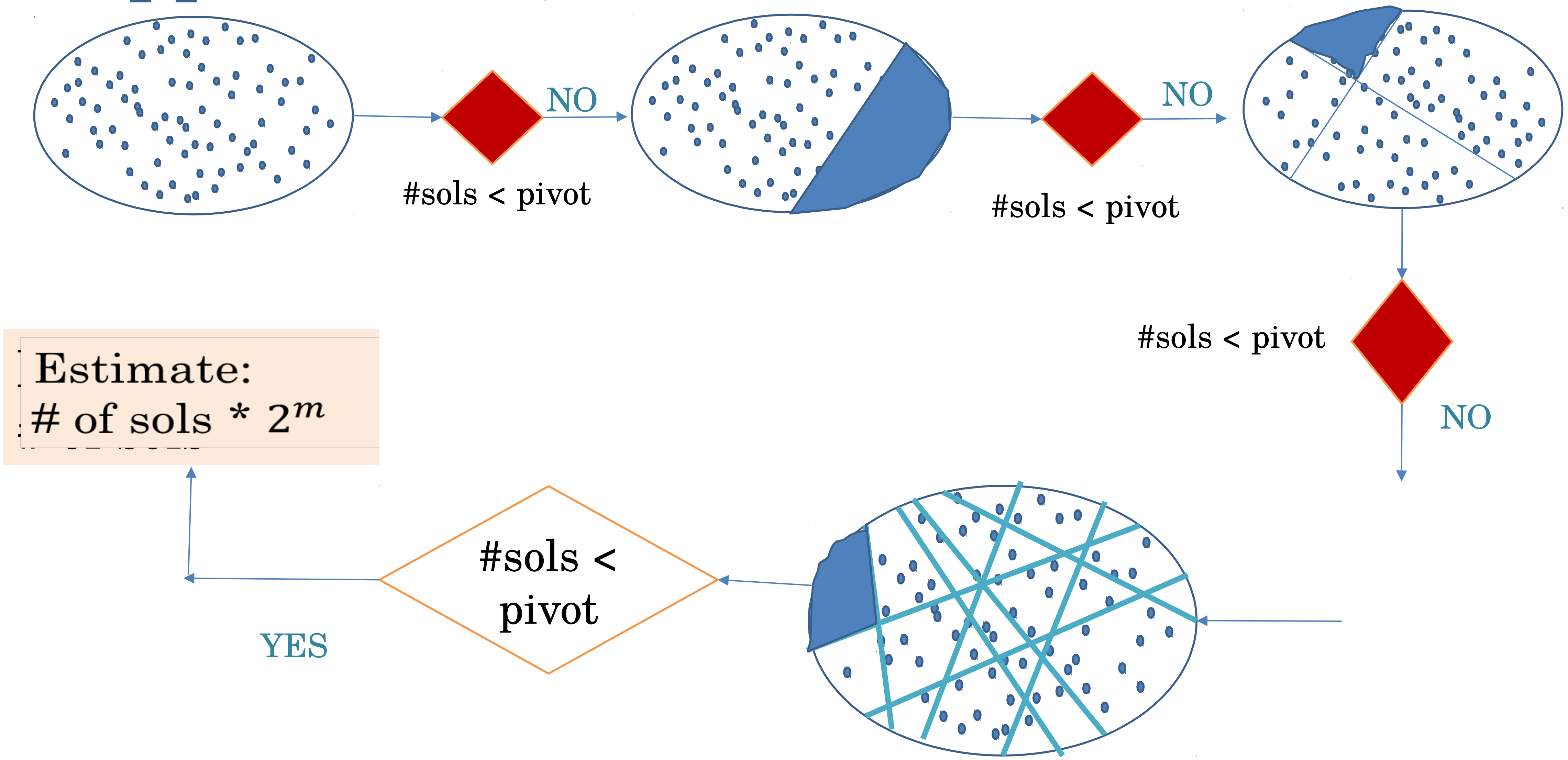
ApproxMC(F,



- Stop at the first m where #of solutions < pivot
- Hash functions must be independent across different checks
- No monotonicity property for # of solutions
- O(n) SAT calls

ApproxMC(F, ε, δ)

ApproxMC(F,



Challenge

- Can we reduce the number of SAT calls from O(n)?

Experimental Observations

- ApproxMC “seems to work” even if we do not have independence across different hash functions
  - Can we really give up independence?

Beyond ApproxMC

- We want to partition into 2^m cells
  - Check for every m = 0,1….n if the number of solutions < pivot
  - Stop at the first m where number of solutions < pivot
  - Hash functions must be independent across different checks (Stockmeyer 1983, Jerrum, Valiant and Vazirani 1986…..)
- **Suppose:** Hash functions *can be dependent* across different checks
- # of solutions is monotonically non-increasing with m
  - Can find the right value of m by search in any order.
  - Binary search

ApproxMC2: From Linear to Logarithmic SAT calls

- The Proof: Hash functions *can be dependent* across different checks

- Key Idea: Probability of making a bad choice early on is very small.
  - Inversely (exponentially!) proportional to distance from m*)

ApproxMC2(F, ε, δ)

ApproxMC2(F,

Theorem 1:

Theorem 1:

$$\Pr\left[\frac{|R_F|}{(1+\varepsilon)} \leq \text{ApproxMC2}(F, \varepsilon, \delta) \leq |R_F|(1+\varepsilon)\right] \geq 1-\delta$$

Theorem 2:

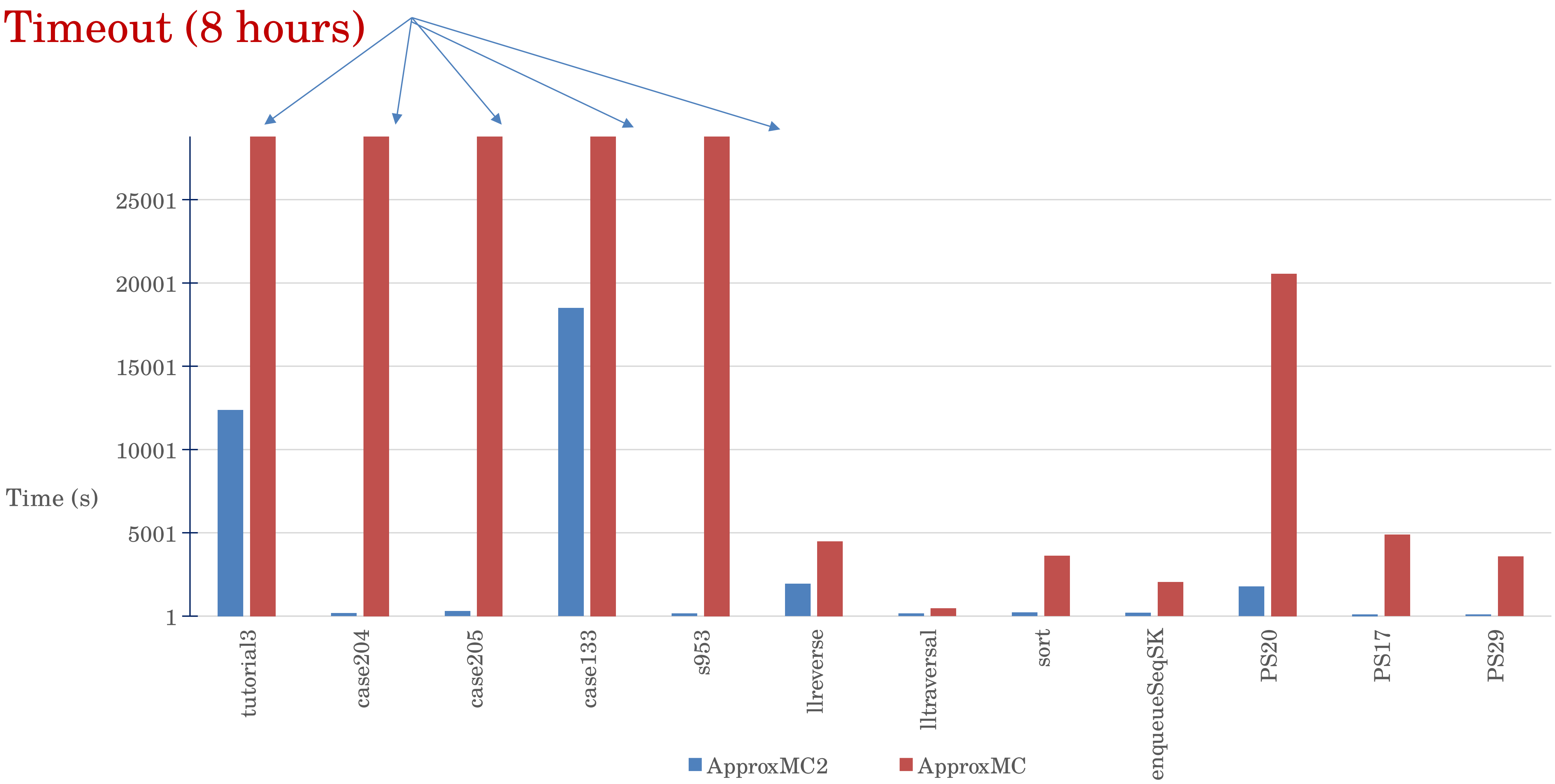
Theorem 2:

ApproxMC2(F, calls to NP oracle

ApproxMC2(F, ε, δ) makes $O\left(\frac{(\log n) \log\frac{1}{\delta}}{\varepsilon^2}\right)$ calls to NP oracle

Theorem 1 requires a completely new proof framework.

Runtime Performance Comparison



Beyond ApproxMC

- The proposed proof framework can be applied to other algorithms
  - PAWS (Ermon et al 2014)
  - WeightMC (Chakraborty et al 2014, Belle et al 2015)
- Reduces number of SAT calls from O(n) to O(log n)

Summary

- Model Counting as "assembly language" for inference
- Hashing-based techniques make O(n) or O(n log n) NP-oracle (SAT) queries
- The new proof framework reduces number of queries to O(log n)
- Massive speedups for ApproxMC, the state of the art approximate model counter