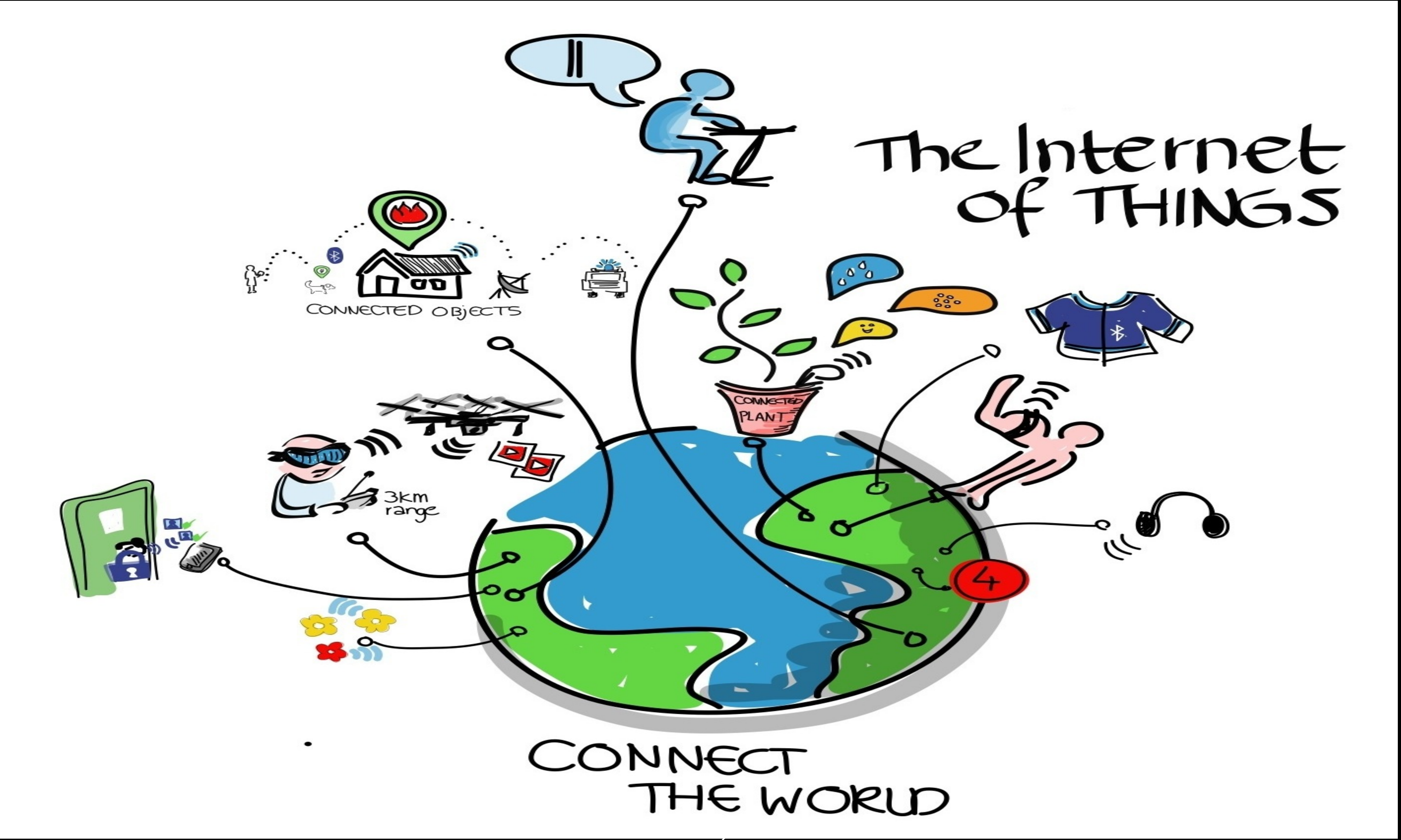
Approximating Probabilistic Inference without Losing Guarantees: Combining Hashing and Feasibility

Kuldeep S. Meel

PhD Student
CAVR Group

Joint work with Supratik Chakraborty, Daniel J. Fremont, Sanjit A. Seshia, Moshe Y. Vardi

IoT: Internet of Things



The Era of Data

- How to make sense of data?
- Modeling the events
- Infer likelihood from data

Probabilistic Inference

Given that a Rice CS grad student queried “music” on google, what is the probability they will click on “The best of Justin Beiber” ?

$$\Pr\,[\text{event}|\text{evid}]$$

Probabilistic Inference

Given that a Rice CS grad student queried “music” on google, what is the probability they will click on “The best of Justin Beiber” ?

$$\Pr\,[\text{event} \mid \text{evid}]$$

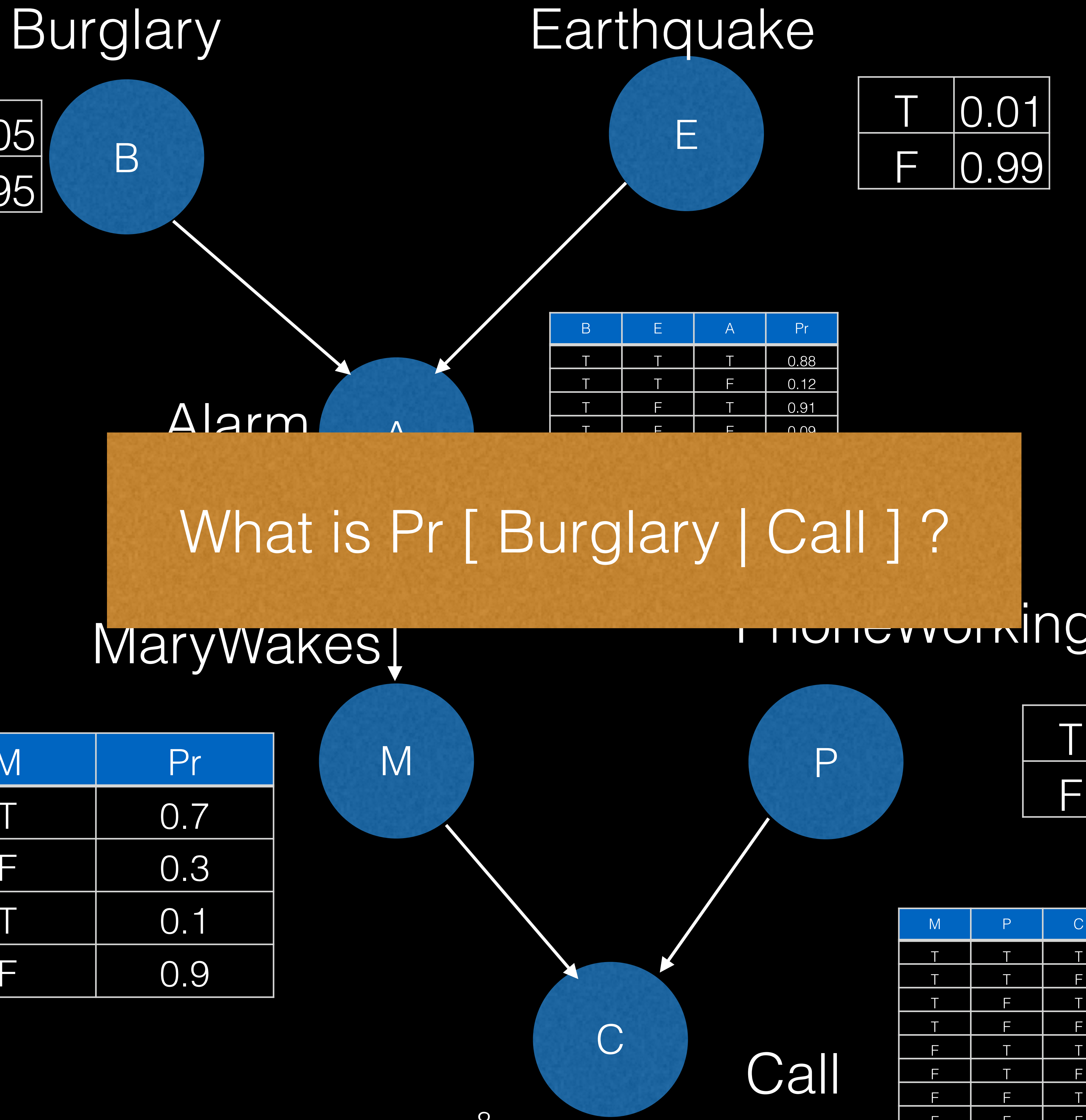
Probabilistic Inference

Given that a Rice CS grad student queried “music” on google, what is the probability they will click on “The best of Justin Beiber” ?

$$\Pr\,[\text{event}|\text{evid}]$$

Graphical Models

Bayesian Networks



T	0.05
F	0.95

T	0.01
F	0.99

B	E	A	Pr
T	T	T	0.88
T	T	F	0.12
T	F	T	0.91
T	F	F	0.09

What is Pr [Burglary | Call] ?

A	M	Pr
T	T	0.7
T	F	0.3
F	T	0.1
F	F	0.9

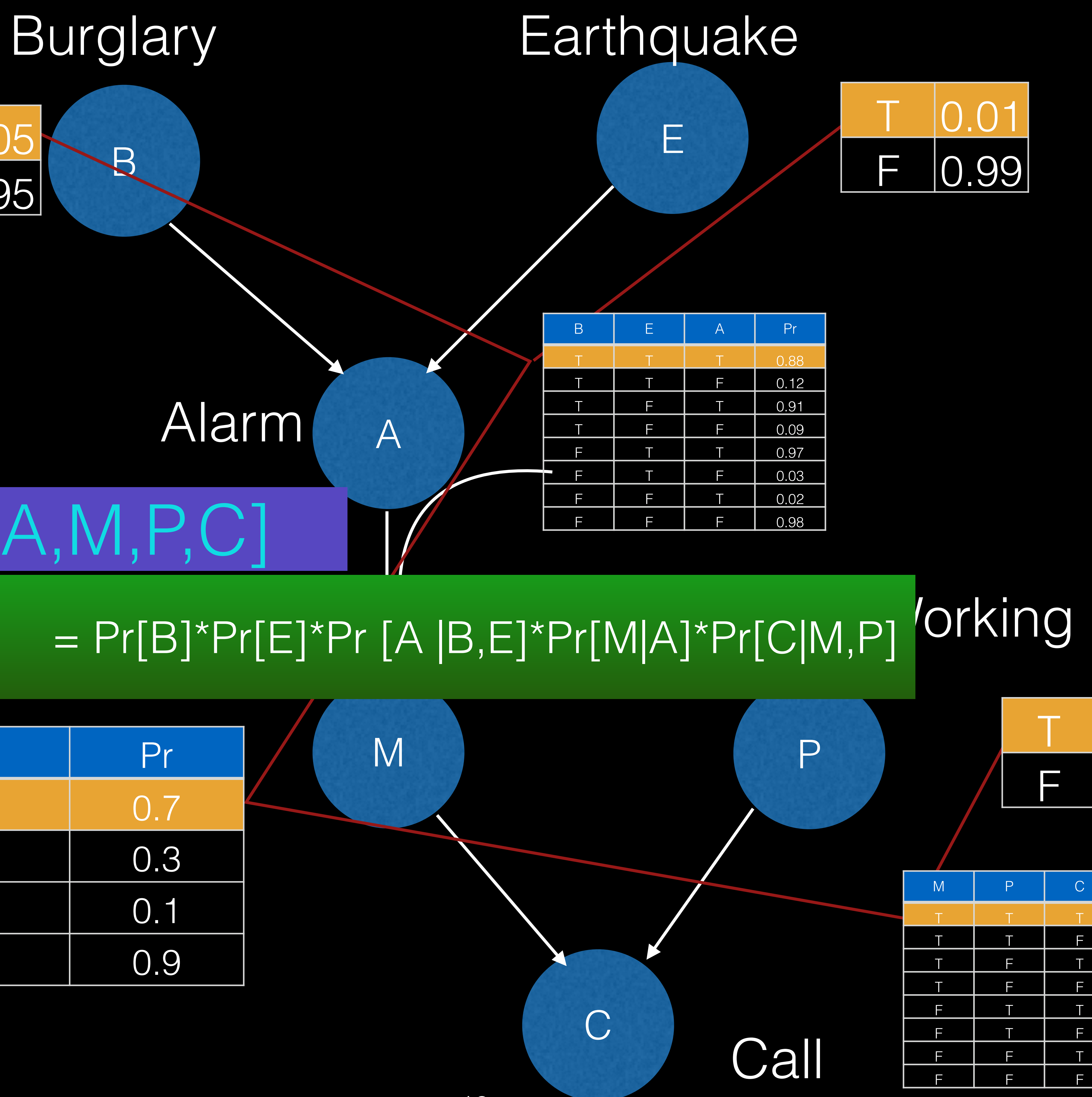
T	0.8
F	0.2

M	P	C	Pr
T	T	T	0.99
T	T	F	0.01
T	F	T	0
T	F	F	1
F	T	T	0
F	T	F	1
F	F	T	0.1
F	F	F	0.9

Bayes' Rule to the Rescue

$$Pr[Burglary|Call] = \frac{Pr[Burglary \cap Call]}{Pr[Call]}$$

$$Pr[Burglary \cap Call] = Pr[B, E, A, M, P, C] + Pr[B, \bar{E}, A, M, P, C] + \cdots$$



T	0.05
F	0.95

T	0.01
F	0.99

B	E	A	Pr
T	T	T	0.88
T	T	F	0.12
T	F	T	0.91
T	F	F	0.09
F	T	T	0.97
F	T	F	0.03
F	F	T	0.02
F	F	F	0.98

Pr [B,E,A,M,P,C]

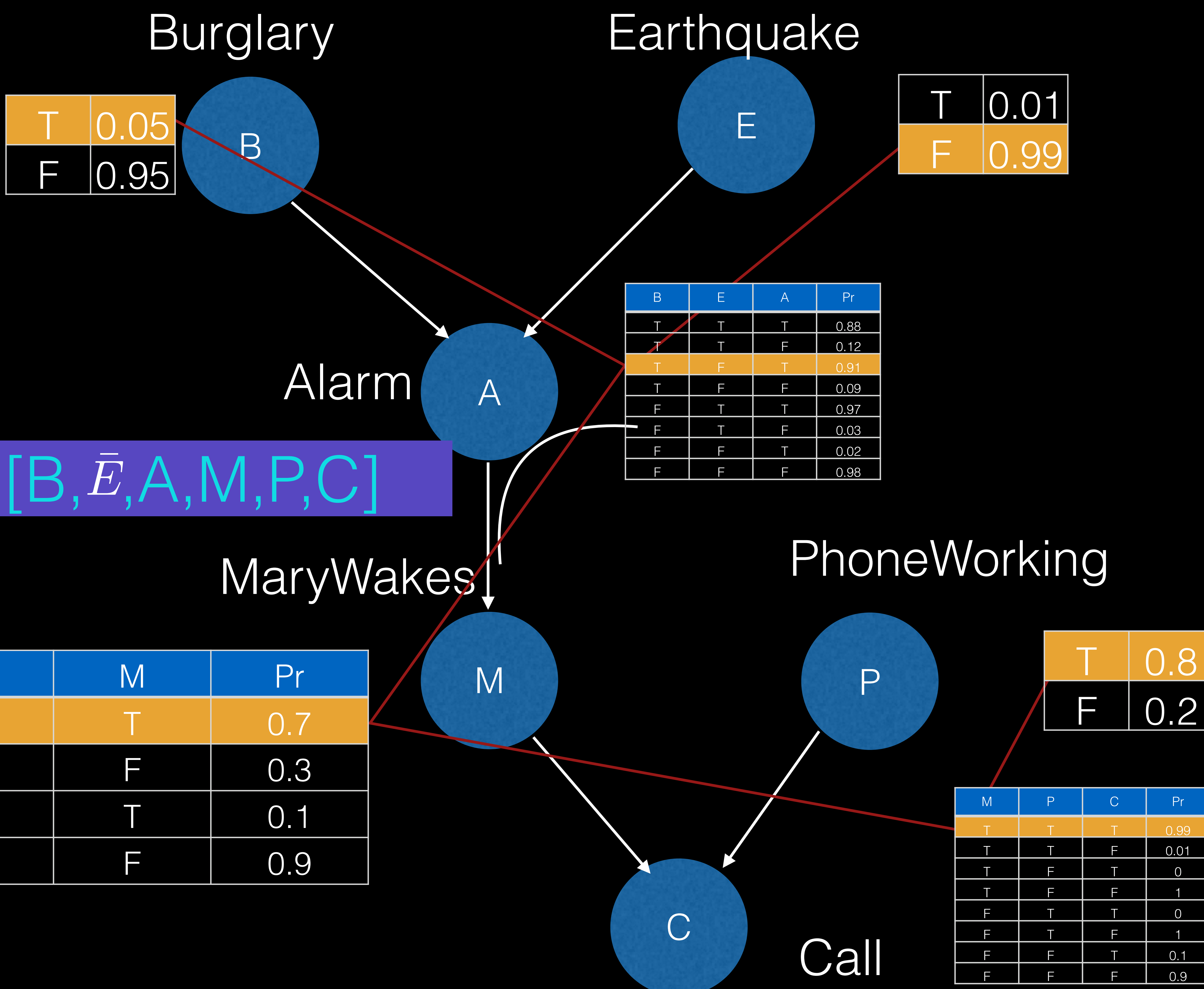
= Pr[B]*Pr[E]*Pr [A |B,E]*Pr[M|A]*Pr[C|M,P]

A	M	Pr
T	T	0.7
T	F	0.3
F	T	0.1
F	F	0.9

T	0.8
F	0.2

M	P	C	Pr
T	T	T	0.99
T	T	F	0.01
T	F	T	0
T	F	F	1
F	T	T	0
F	T	F	1
F	F	T	0.1
F	F	F	0.9

$$\Pr\,[B,\bar{E},A,M,P,C]$$



T	0.05
F	0.95

T	0.01
F	0.99

B	E	A	Pr
T	T	T	0.88
T	T	F	0.12
T	F	T	0.91
T	F	F	0.09
F	T	T	0.97
F	T	F	0.03
F	F	T	0.02
F	F	F	0.98

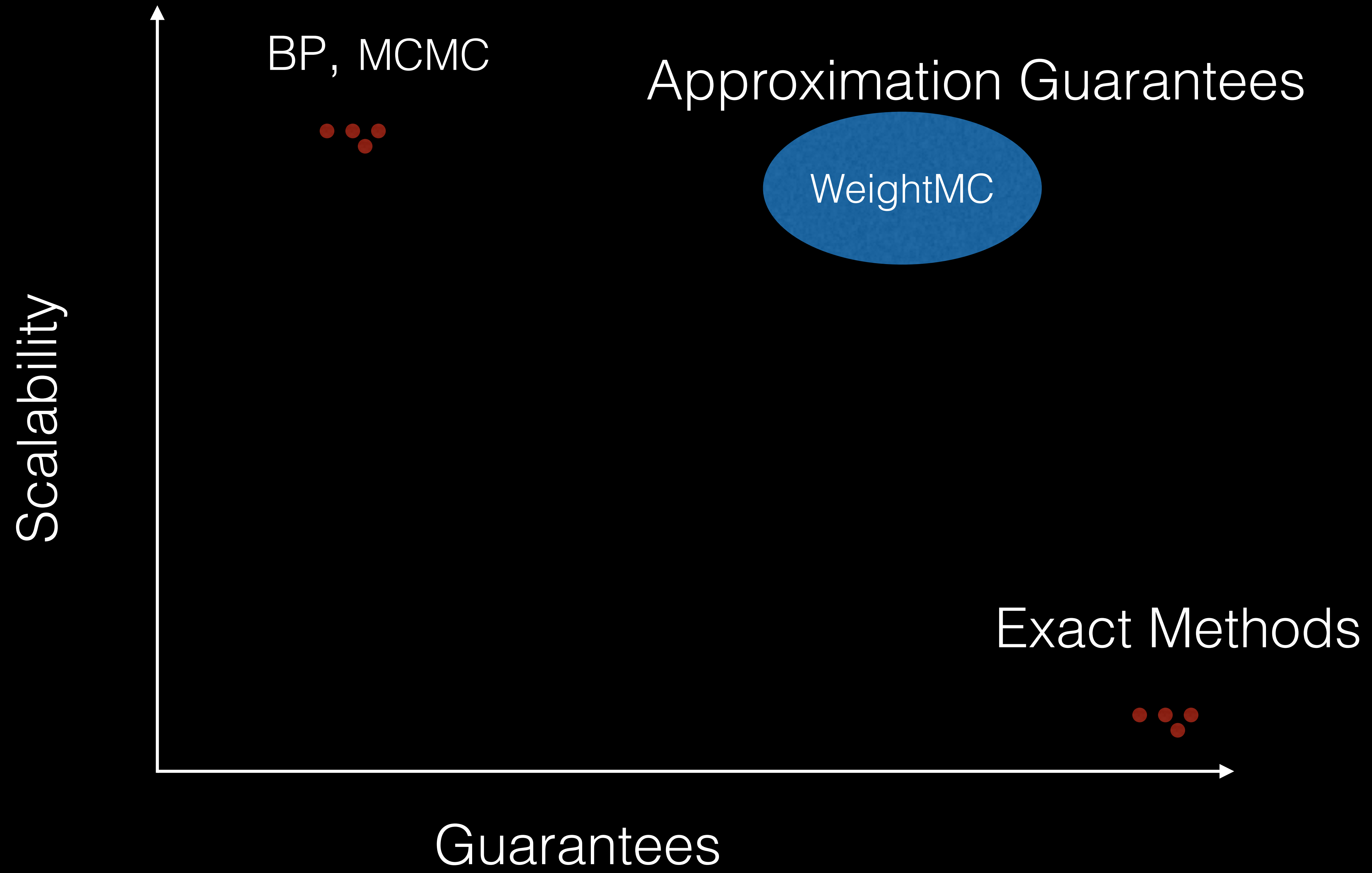
A	M	Pr
T	T	0.7
T	F	0.3
F	T	0.1
F	F	0.9

T	0.8
F	0.2

M	P	C	Pr
T	T	T	0.99
T	T	F	0.01
T	F	T	0
T	F	F	1
F	T	T	0
F	T	F	1
F	F	T	0.1
F	F	F	0.9

Prior Work

Our Contribution



A wild Idea for a new paradigm?

- Partition the space of paths into “small” “equal weighted” cells
    - “Small”: # of paths in a cell is not large (bounded by a constant)
    - “equal weighted”: All the cells have equal weight

Outline

- <u>Reduction to SAT</u>
- Partition-based techniques via (unweighted) model counting
- Extension to Weighted Model Counting
- Looking forward

Boolean Satisfiability

- SAT: Given a Boolean formula F over variables V, determine if F is true for some assignment to V
- $F = (a \vee b)$
- $R_F = \{(0,1),(1,0),(1,1)\}$
- SAT is NP-Complete (Cook 1971)
  - One of the million dollar problems

Model Counting

Given:

- CNF Formula F, Solution Space: R_F

Problem (MC):

What is the total number of satisfying assignments (models) i.e. $|R_F|$?

Example

$F = (a \vee b)$; $R_F = \{[0,1], [1,0], [1,1]\}$

$$|R_F| = 3$$

Weighted Model Counting

Given:

- CNF Formula F, Solution Space: R_F
- Weight Function W(.) over assignments
  - $W(\sigma)$

Problem (WMC):

What is the sum of weights of satisfying assignments i.e. $W(R_F)$?

Example

$F = (a \vee b)$; $R_F = \{[0,1], [1,0], [1,1]\}$

$W([0,1]) = W([1,0]) = 1/3$ $W([1,1]) = W([0,0]) = 1/6$

$$\mathbf{W(R_F) = 1/3+1/3+1/6 = 5/6}$$

Weighted SAT

- Boolean formula F
- Weight function over variables (literals)
- Weight of assignment = product of wt of literals
- $F = (a \vee b)$; $W(a=0) = 0.4$; $W(a = 1) = 1\text{-}0.4 = 0.6$
  $W(b=0) = 0.3$; $W(b = 1) = 0.7$
- $W[(0,1)] = W(a = 0) \times W(b = 1) = 0.4 \times 0.7 = 0.28$

Reduction to W-SAT

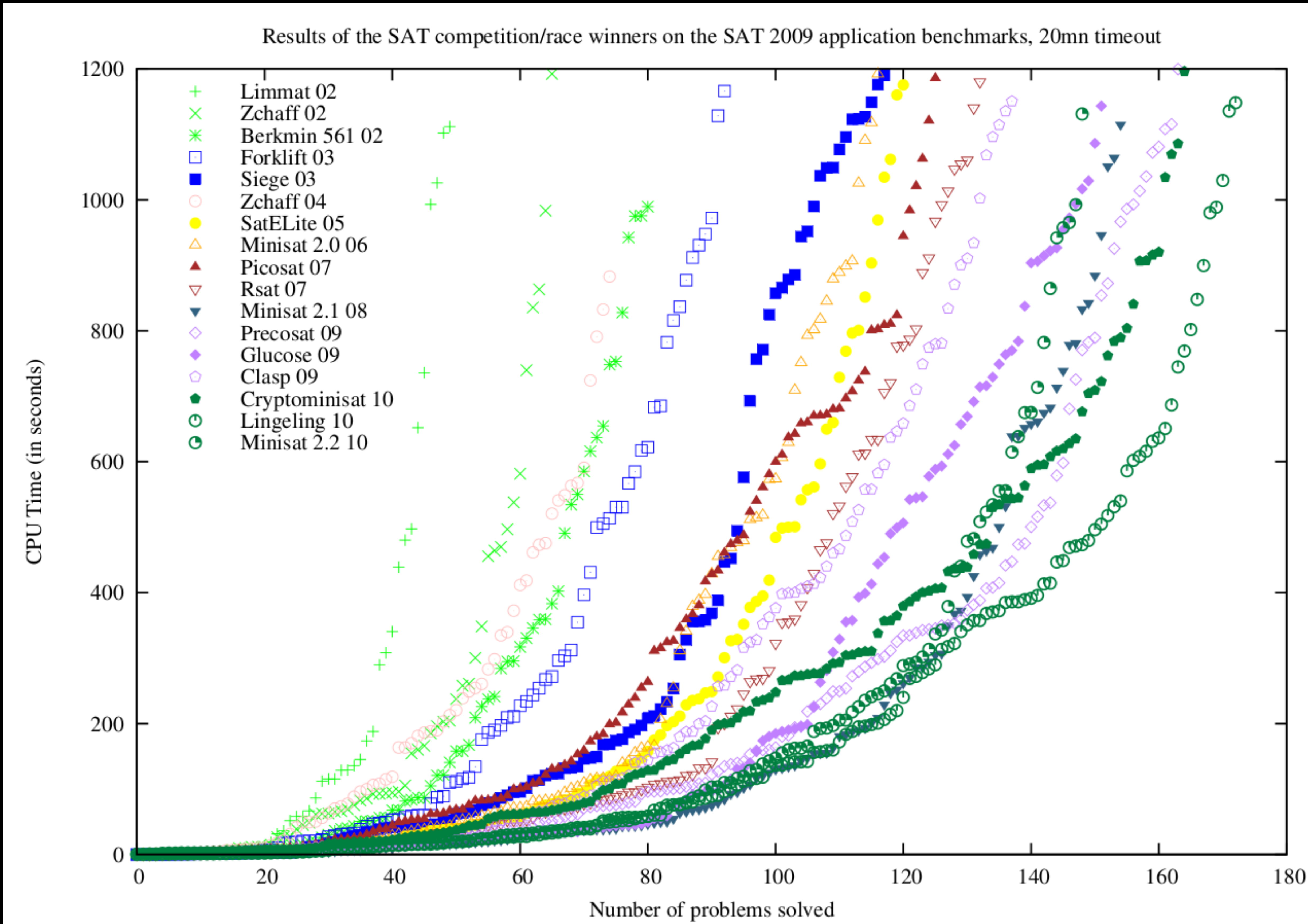
Bayesian Network	SAT Formula
Nodes	Variables
Rows of CPT	Variables
Probabilities in CPT	Weights
Event and Evidence	Constraints

Reduction to W-SAT

- Every satisfying assignment = A valid path in the network
  - Satisfies the constraint (evidence)
- Probability of path = Weight of satisfying assignment = Product of weight of literals = Product of conditional probabilities
- Sum of probabilities = Weighted Sum

Why SAT?

- SAT stopped being NP-complete in practice!
- zchaff (Malik, 2001) started the SAT revolution
- SAT solvers follow Moore’s law



Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout
CPU Time (in seconds)
Number of problems solved
0
200
400
600
800
1000
1200
0
20
40
60
80
100
120
140
160
180
Limmat 02
Zchaff 02
Berkmin 561 02
Forklift 03
Siege 03
Zchaff 04
SatELite 05
Minisat 2.0 06
Picosat 07
Rsat 07
Minisat 2.1 08
Precosat 09
Glucose 09
Clasp 09
Cryptominisat 10
Lingeling 10
Minisat 2.2 10

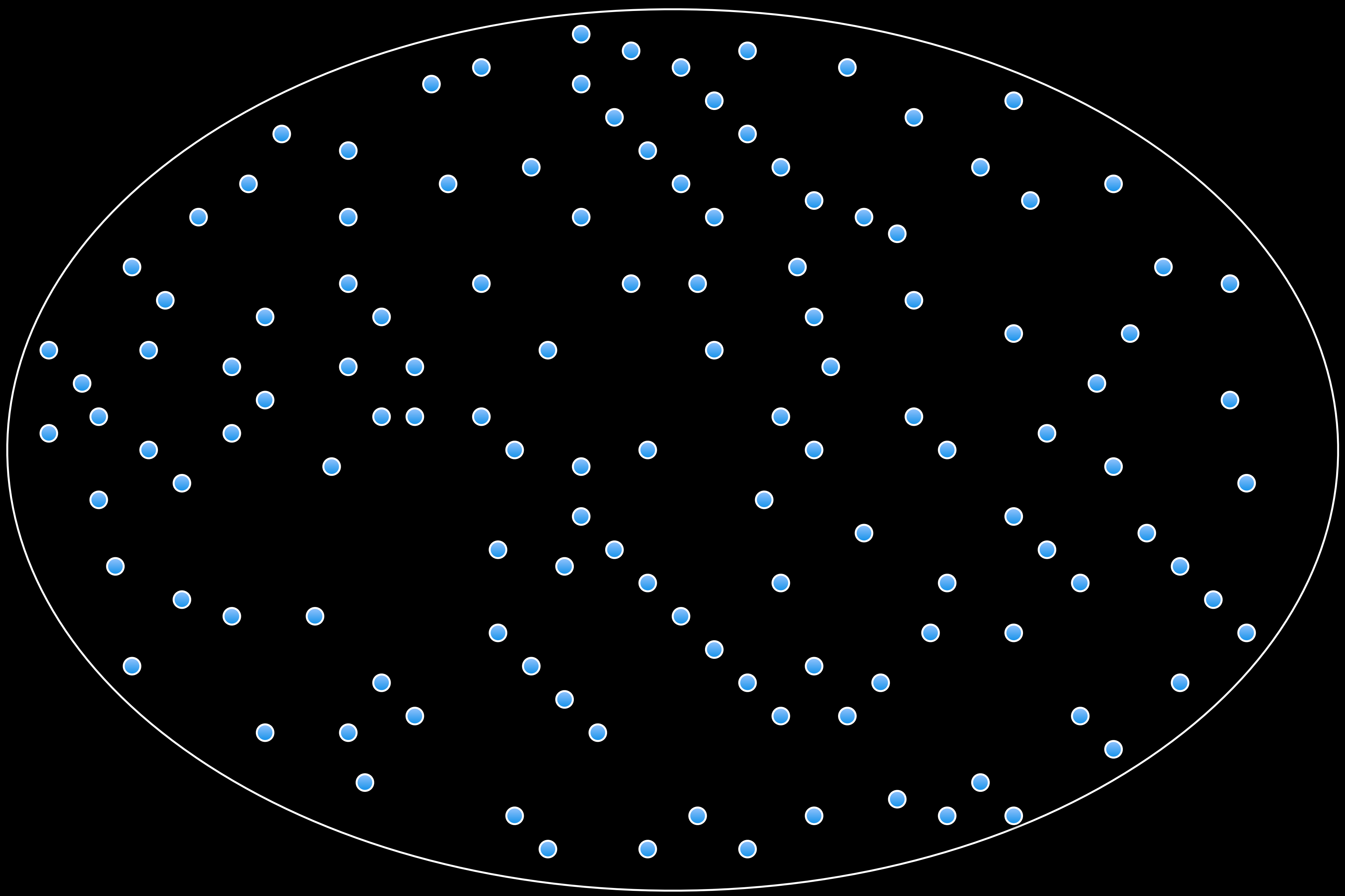
Why SAT?

- SAT stopped being NP-complete in practice!
- zchaff (Malik, 2001) started the SAT revolution
- SAT solvers follow Moore’s law
- “Symbolic Model Checking without BDDs”: most influential paper in the first 20 years of TACAS
- A simple input/output interface

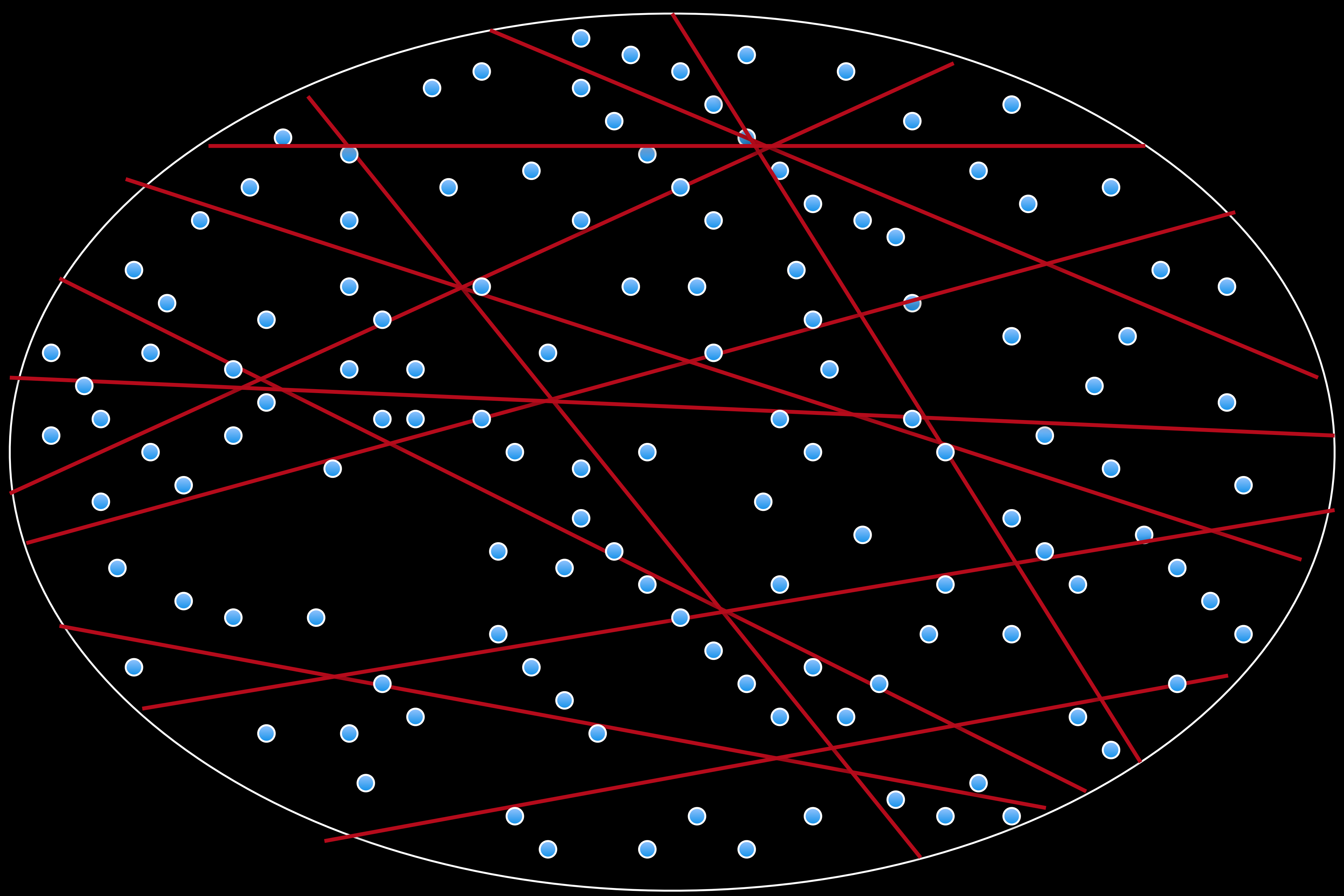
Outline

- Reduction to SAT
- Partition-based techniques via (unweighted) model counting
- Extension to Weighted Model Counting
- Looking forward

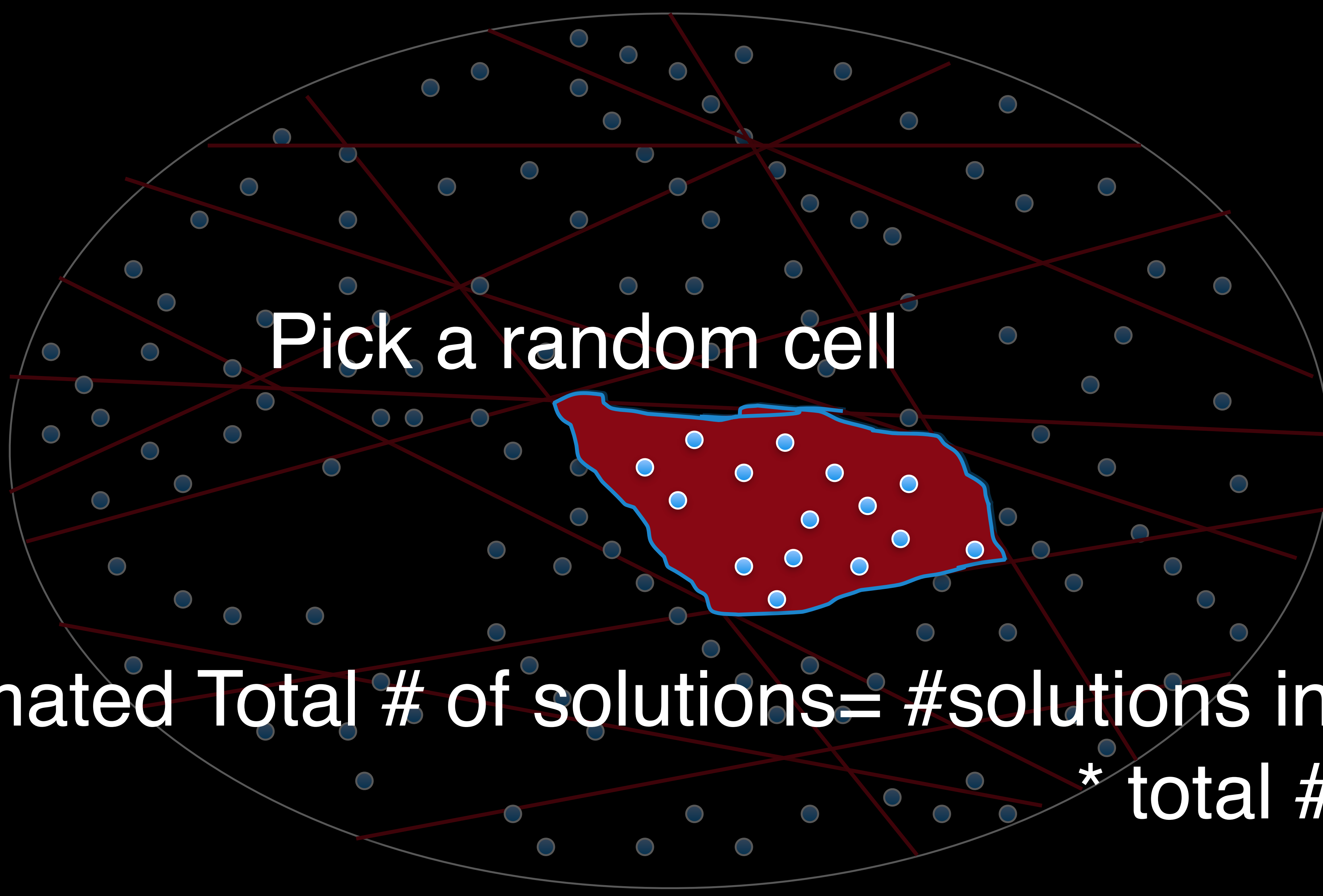
Counting through Partitioning



Counting through Partitioning



Counting through Partitioning



Estimated Total # of solutions= #solutions in the cell
* total # of cells

How to Partition?

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

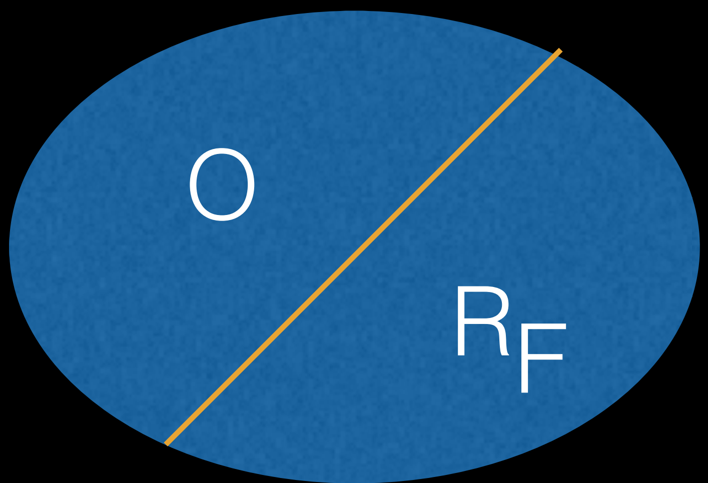
Universal Hashing
[Carter-Wegman 1979, Sipser 1983]

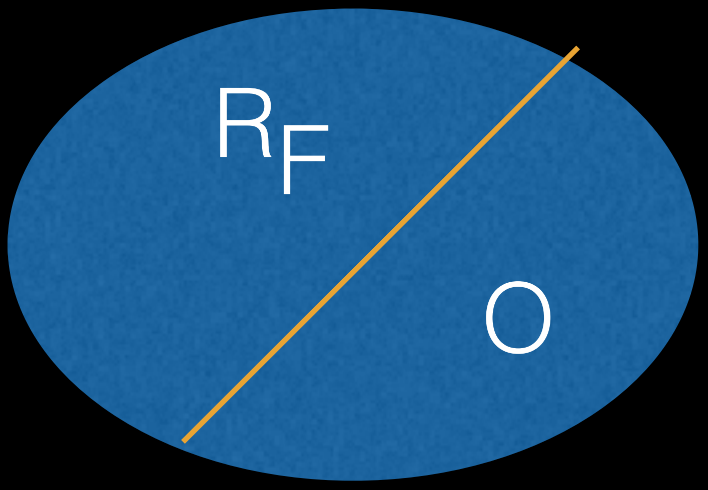
Universal Hashing

- Hash functions from mapping $\{0,1\}^n$ to $\{0,1\}^m$
  - (2^n elements to 2^m cells)

- Random inputs => All cells are *roughly* equal in expectation

- Universal hash functions:
  - For any distribution) inputs => All cells are *roughly* equal in expectation





Universal Hashing

- Hash functions from mapping $\{0,1\}^n$ to $\{0,1\}^m$
  - (2^n elements to 2^m cells)
- Random inputs => All cells are *roughly* equal in <u>expectation</u>
- Universal hash functions:
  - For any distribution) inputs => All cells are *roughly* equal in <u>expectation</u>
- Need stronger bounds on range of the size of cells

Lower Universality $\Rightarrow$ Lower Complexity

- H(n,m,r): Family of r-universal hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$ (2^n elements to 2^m cells)

- Higher the r => Stronger guarantees on variance of size of cells

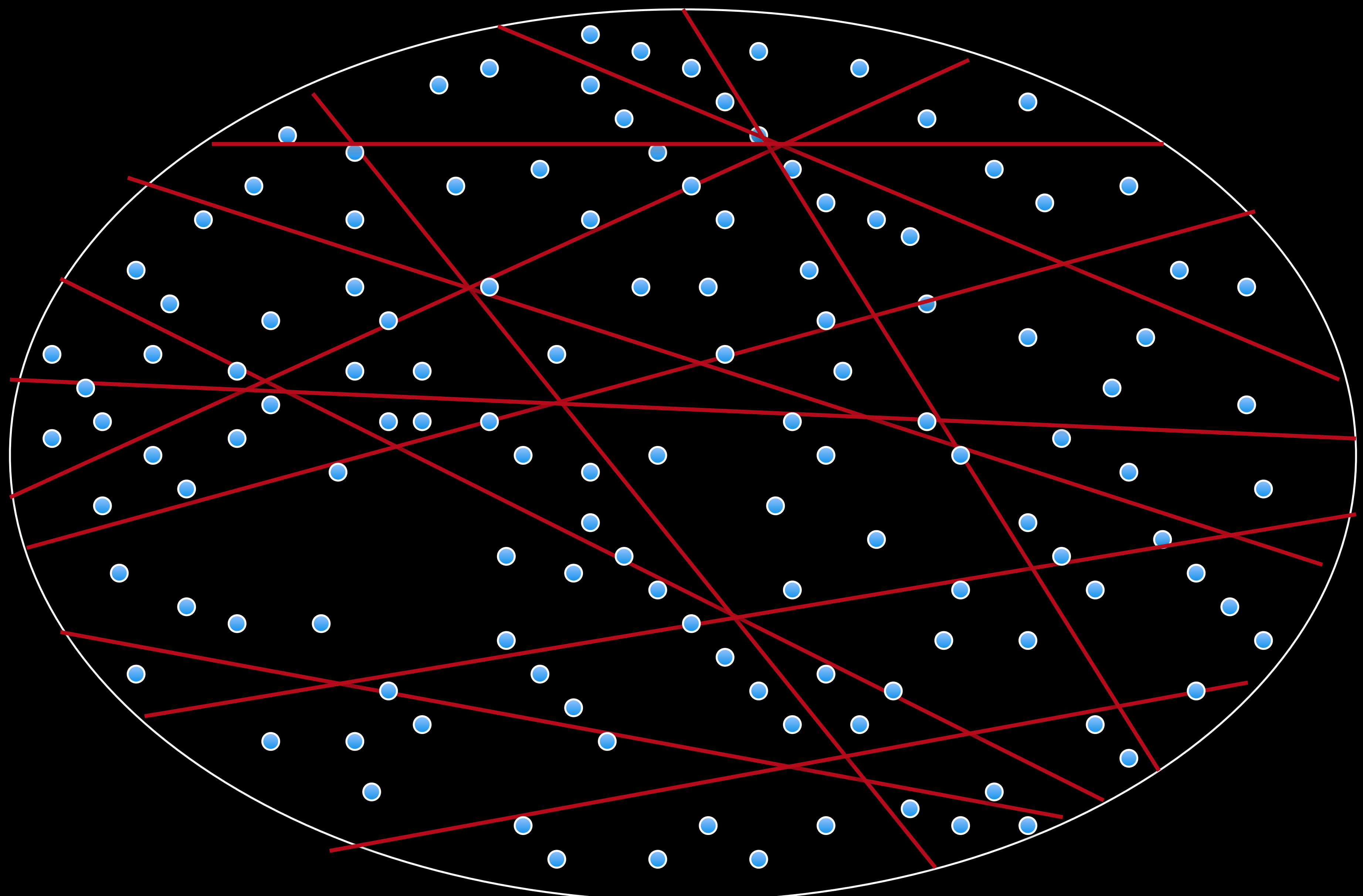
- r-wise universality => Polynomials of degree r-1

- Lower universality => lower complexity

XOR-Based Hashing

- 3-universal 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 equate to 0/1 with prob. ½
- $X_1+X_3+X_6+\ldots X_{n-1} = 0$
- m XOR equations -> 2^m cells

Counting through Partitioning



Partitioning

- How large should the cells be?

- How many cells?

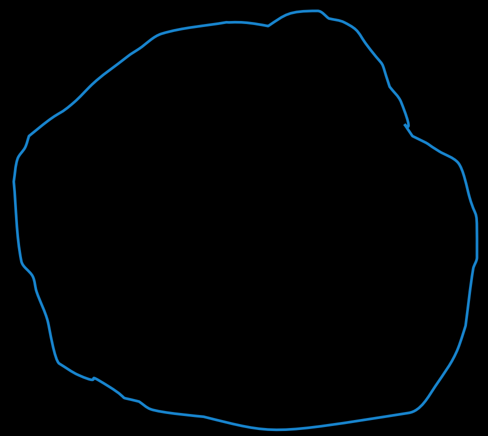
Size of cell

- Too large => Hard to enumerate
- Too small => Variance can be very high
- More tight bounds => larger cell

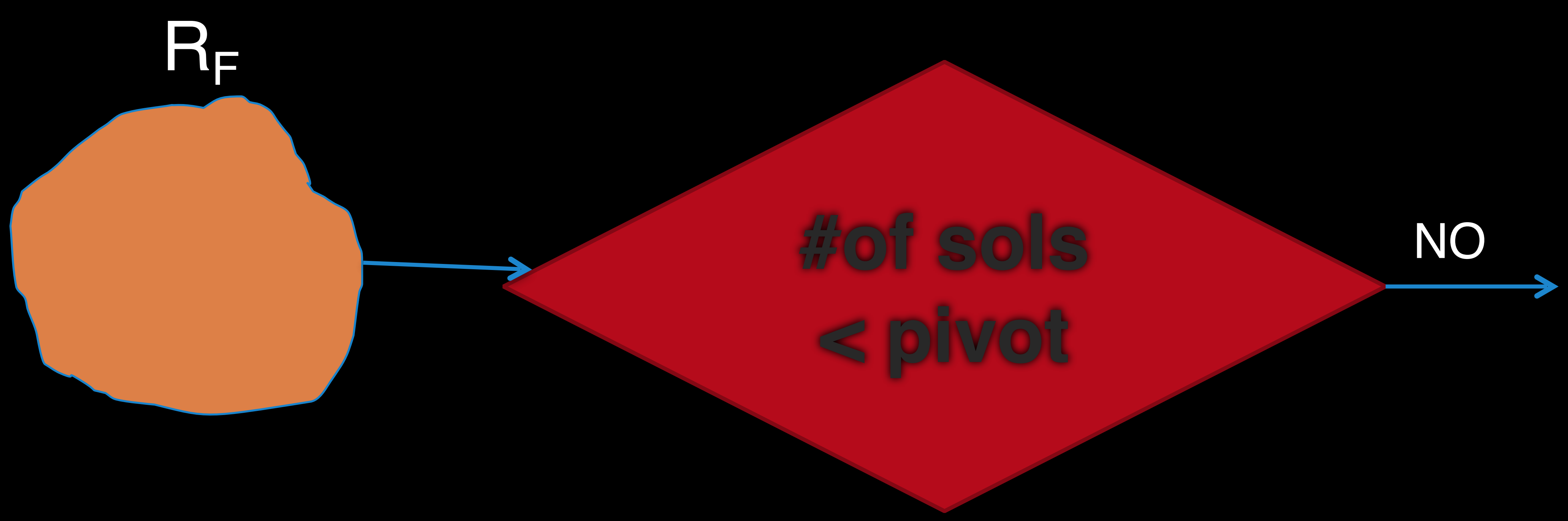
$$\text{pivot} = 5(1 + 1/\varepsilon)^2$$

ApproxMC

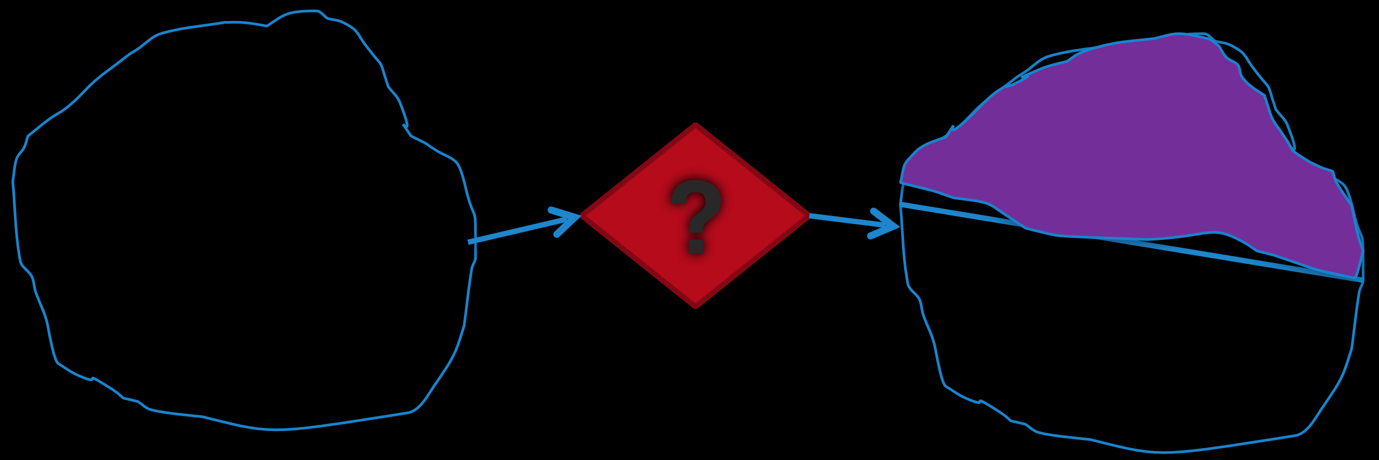
R_F



ApproxMC

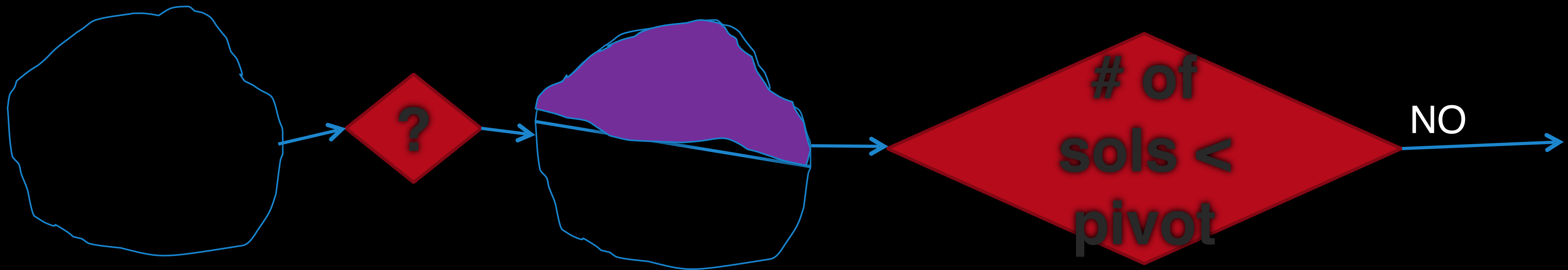


ApproxMC

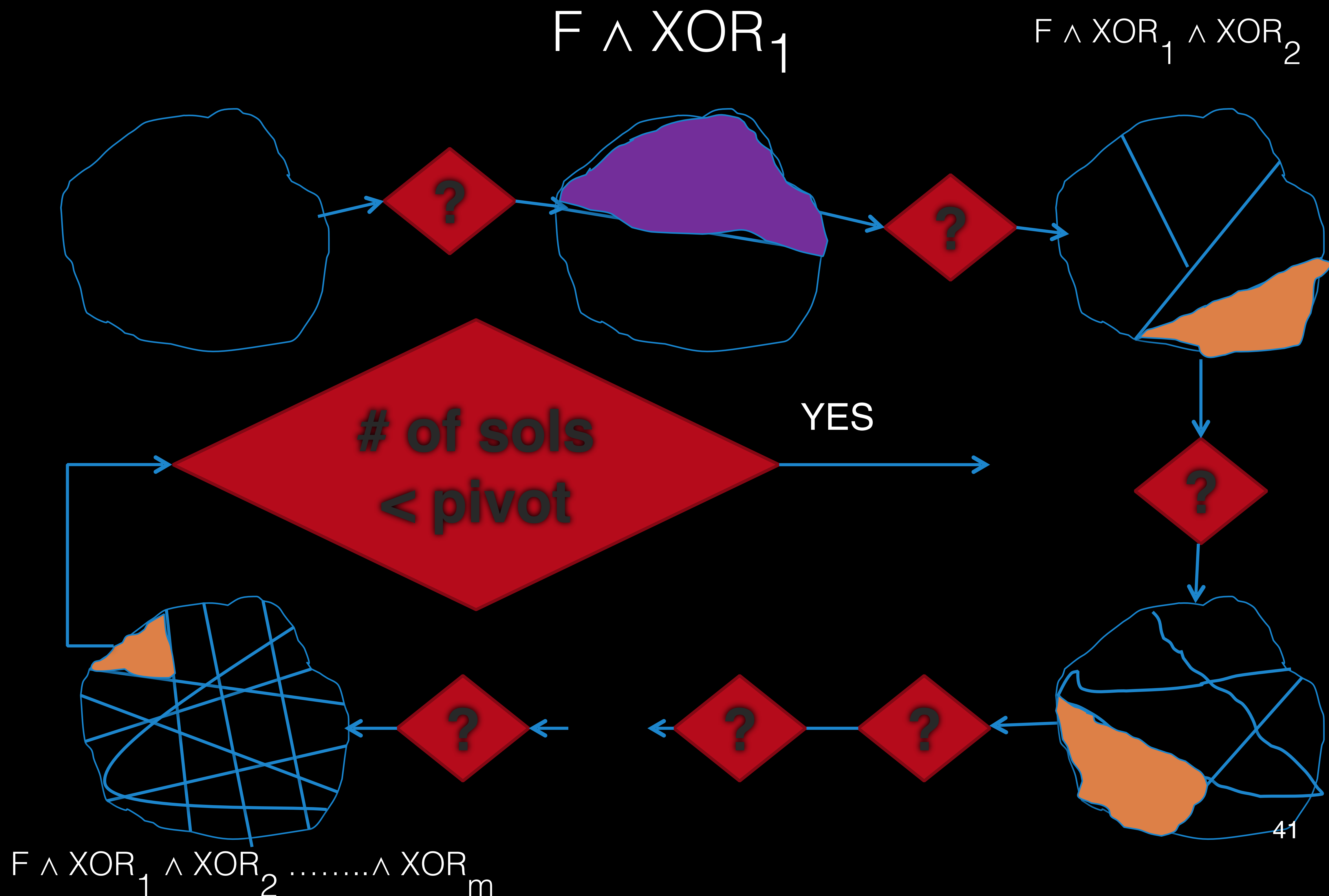


$F \wedge XOR_1$

ApproxMC

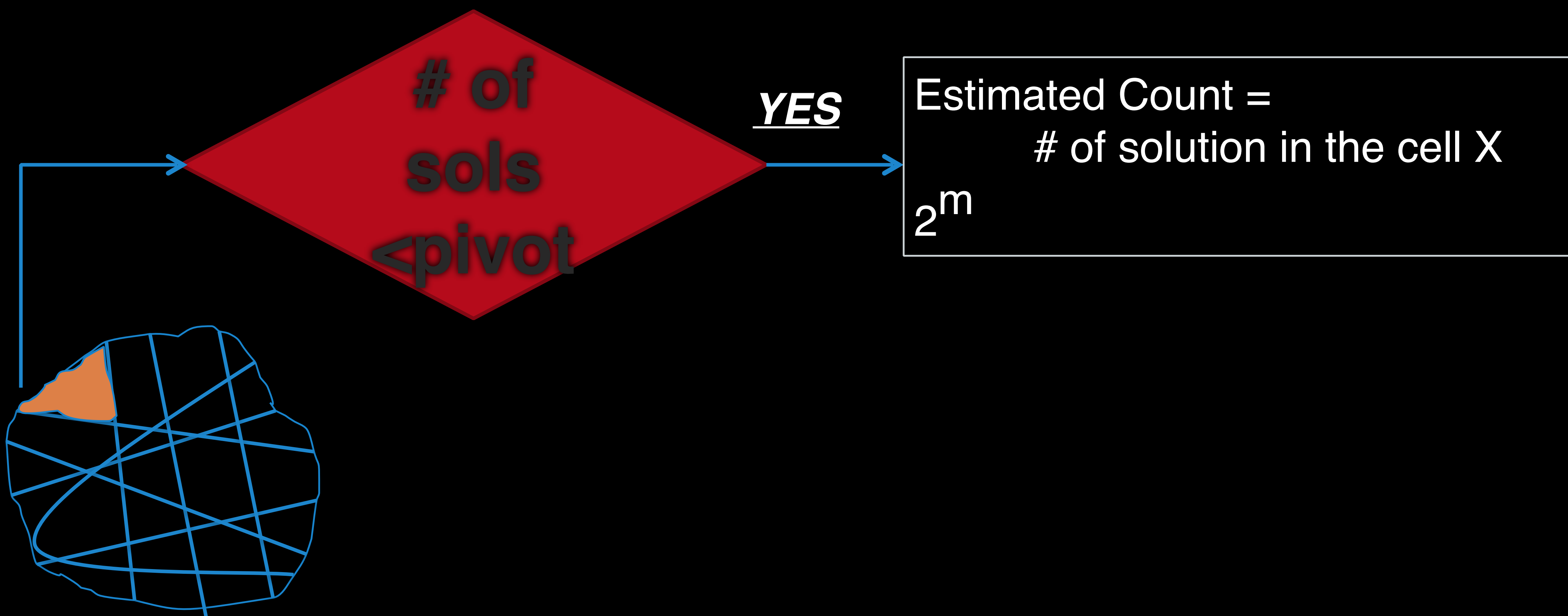


ApproxMC



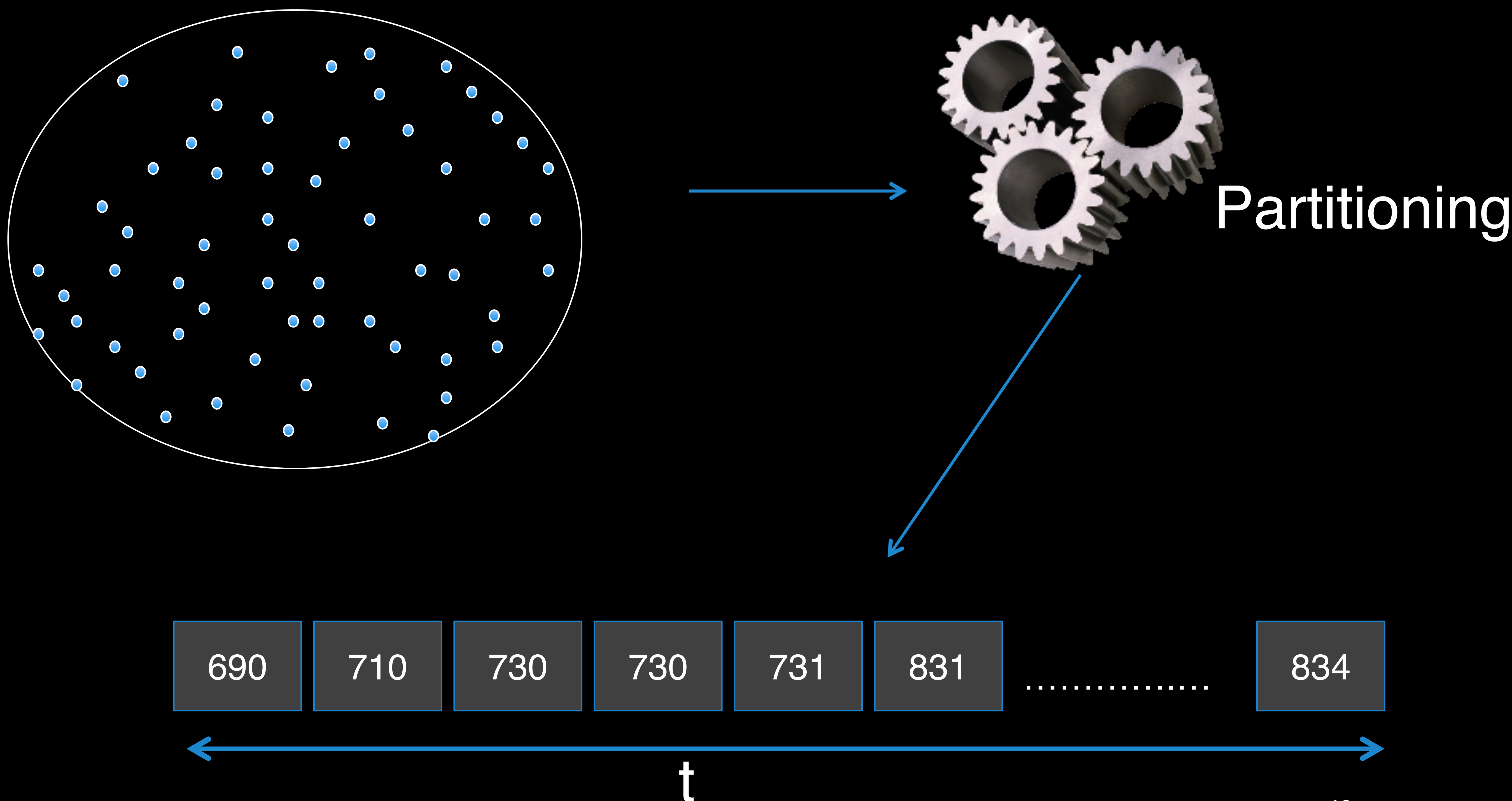
$F \wedge XOR_1$
$F \wedge XOR_1 \wedge XOR_2$
?
# of sols < pivot
YES
$F \wedge XOR_1 \wedge XOR_2 \ldots\ldots\wedge XOR_m$

ApproxMC

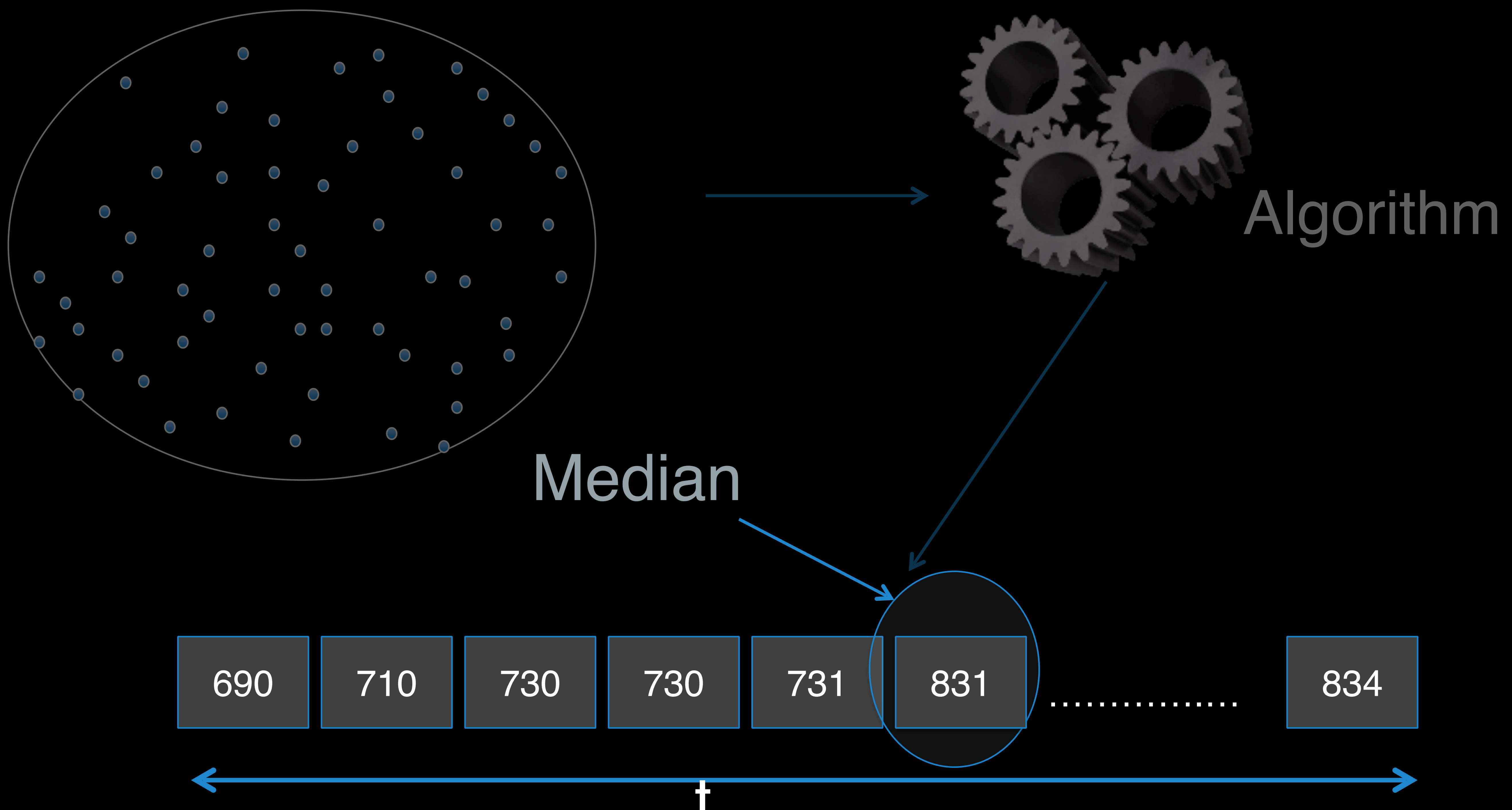


$F \wedge XOR_1 \wedge XOR_2 \ldots\ldots\wedge XOR_m$

ApproxMC



ApproxMC in Action



Strong Theoretical Results

ApproxMC (CNF: F, tolerance: ε, confidence: δ)

Suppose ApproxMC(F, ε, δ) returns C. Then,

$$\mathbf{Pr}\ [\ \#\mathbf{F}/(1+\varepsilon) <= \mathbf{C} <= (1+\varepsilon)\ \#\mathbf{F}\ ] \geq \delta$$

ApproxMC runs in time polynomial in log $(1-\delta)^{-1}$, |F|, ε^{-1} relative to SAT oracle

Key Idea behind the Proof

$$\text{Let } I_1, I_2, I_3, \cdots I_n \text{ be } 3 - \text{wise independent variables in } [0,1],$$

$$\text{then for } I = \sum I_k, \mu = E[I]$$

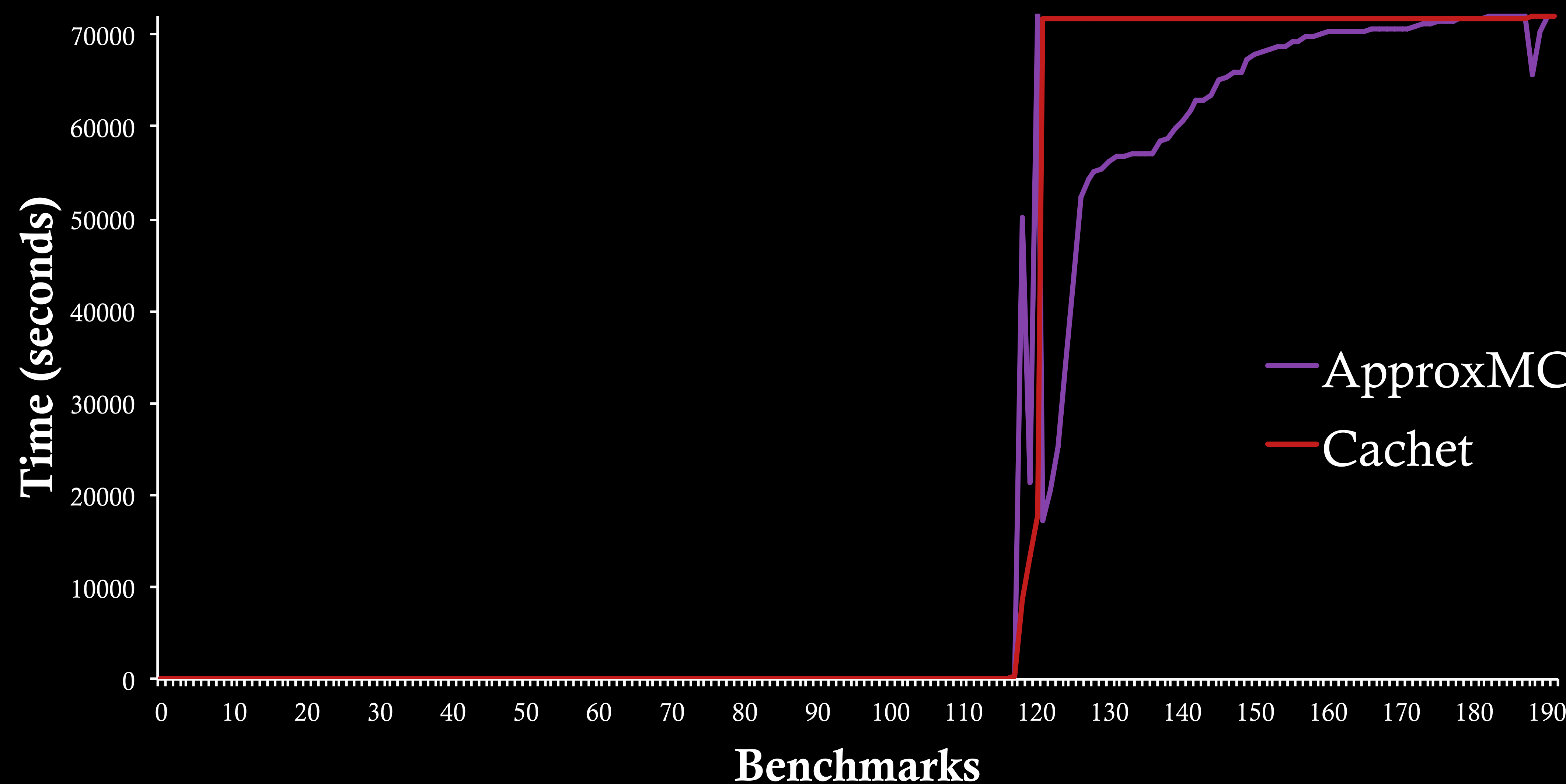
$$Pr[|I - \mu| < \beta\mu] \geq 0.7$$

$I_k = 1$ if y_k is in the cell

Pr[$I_k = 1$] = 1/2^m $\mu = \frac{R_F}{2^m}$

Pr [#**F**/(1+ε) <= **C** <= (1+ ε) #**F** **]** **≥** 0.7

Can Solve a Large Class of Problems

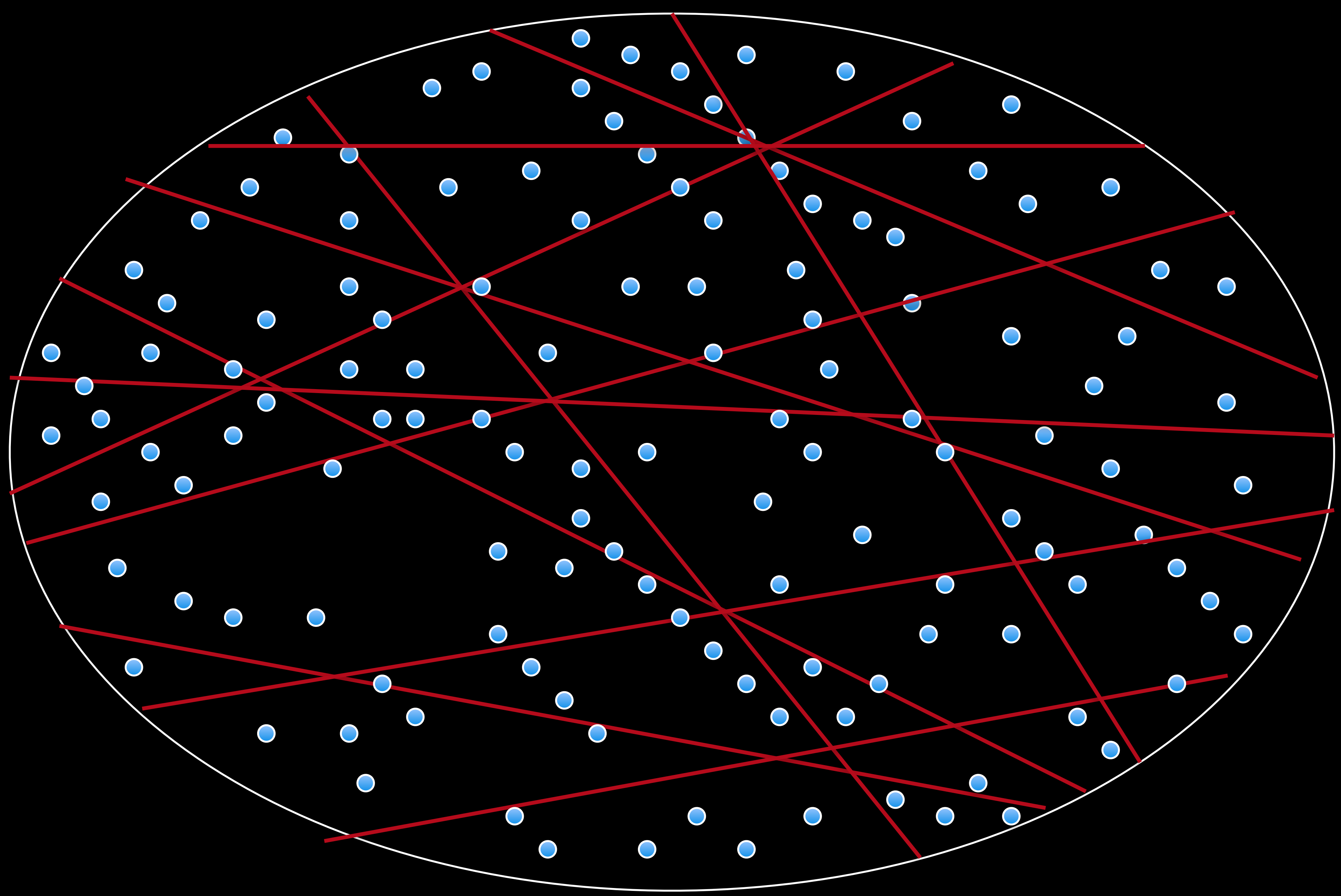


Large class of problems that lie beyond the exact counters but can be computed by ApproxMC

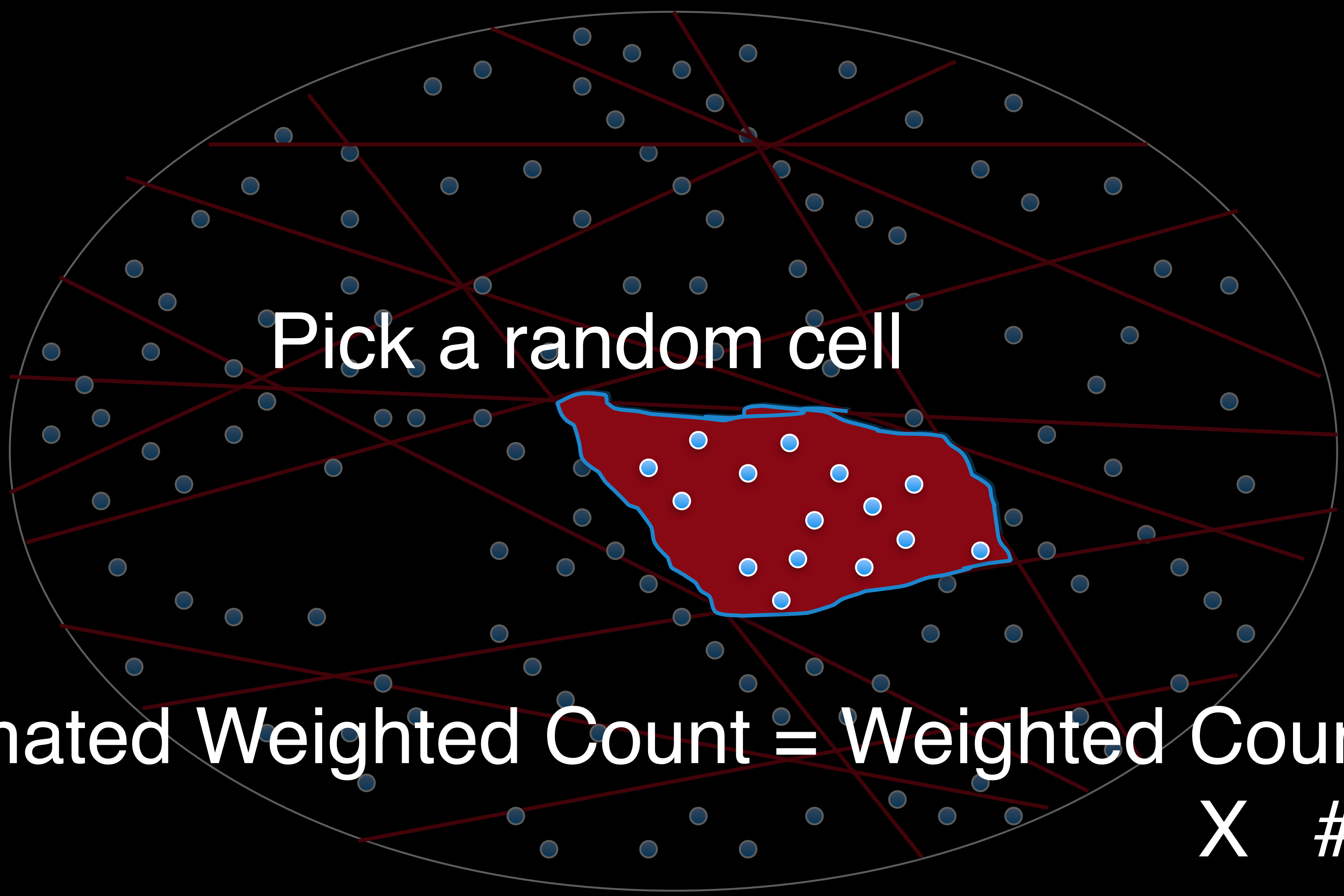
Outline

- Reduction to SAT
- Partition-based techniques via (unweighted) model counting
- Extension to Weighted Model Counting
- Looking forward

Partitioning into equal (weighted) “small” cells



Partitioning into equal (weighted) “small” cells



Estimated Weighted Count = Weighted Count of cell X # of cells

Key Modifications

$$\text{Let } I_1, I_2, I_3, \cdots I_n \text{ be } 3-\text{wise independent variables in } [0,1],$$

$$\text{then for } I = \sum I_k, \mu = E[I]$$

$$Pr[|I-\mu| < \beta\mu] \geq 0.7$$

$$I_k = \frac{w(y_k)}{w_{max}} \text{ if } y_k \text{ is in the cell}$$

Pr[$I_k = 1$] = 1/2^m

$$\mu = \frac{W(R_F)}{2^m}$$

# of solutions in a cell < $\frac{w_{max}}{w_{min}}$pivot

$$\rho = \frac{w_{max}}{w_{min}}$$

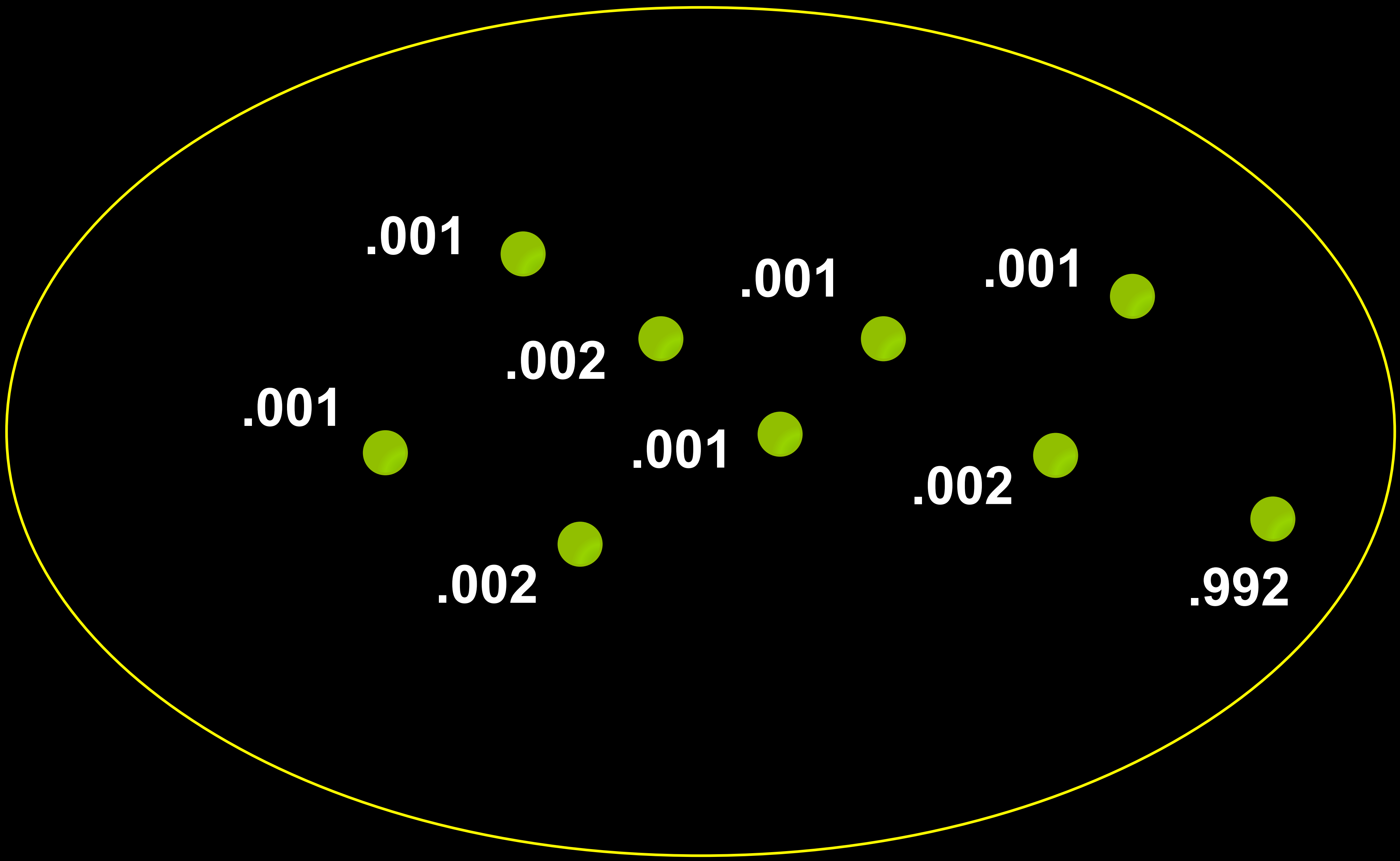
Strong Theoretical Guarantees

- **Approximation: WeightMC(** B, ϵ, δ **), returns C s.t.**

$$Pr[\frac{f}{1+\epsilon} \le C \le f(1+\epsilon)] \ge 1-\delta$$

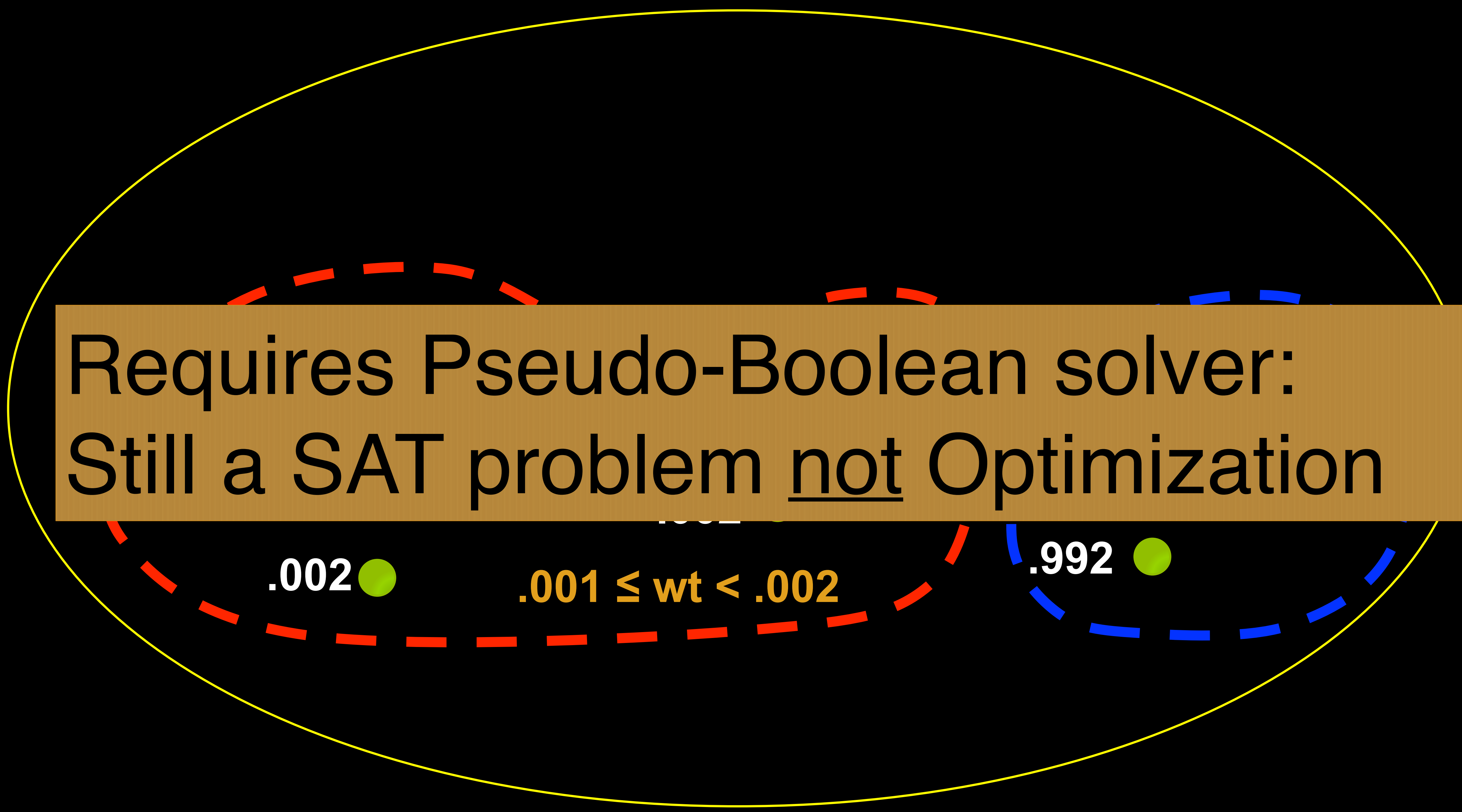
- **Complexity:** # of calls to SAT solver is linear in ρ and polynomial in $\log \delta^{-1}, |F|, 1/\varepsilon$

Handling Large Tilt



Tilt: 992

Handling Large Tilt



Requires Pseudo-Boolean solver:
Still a SAT problem <u>not</u> Optimization

Tilt: 992

Tilt for each region: 2

Main Contributions

- Novel parameter, tilt (ρ), to characterize complexity
  - $\rho = W_{max} / W_{min}$ over satisfying assignments
- Small Tilt (ρ)
  - Efficient hashing-based technique requires only SAT solver
- Large Tilt (ρ)
  - Divide-and-conquer using Pseudo-Boolean solver

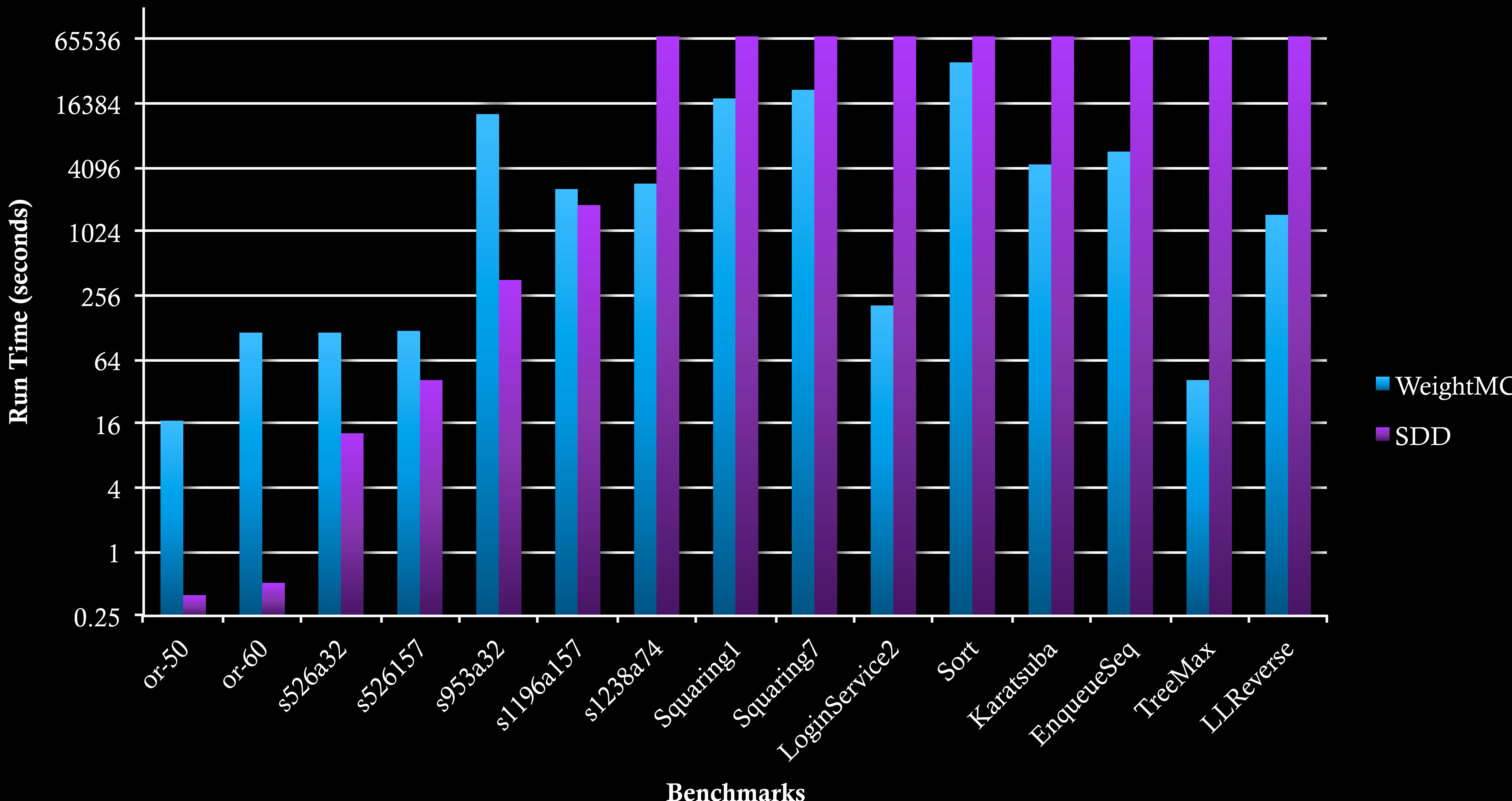
Strong Theoretical Guarantees

- **Approximation:** **WeightMC(** B, ϵ, δ **), returns C s.t.**

$$Pr[\frac{f}{1+\epsilon} \leq C \leq f(1+\epsilon)] \geq 1-\delta$$

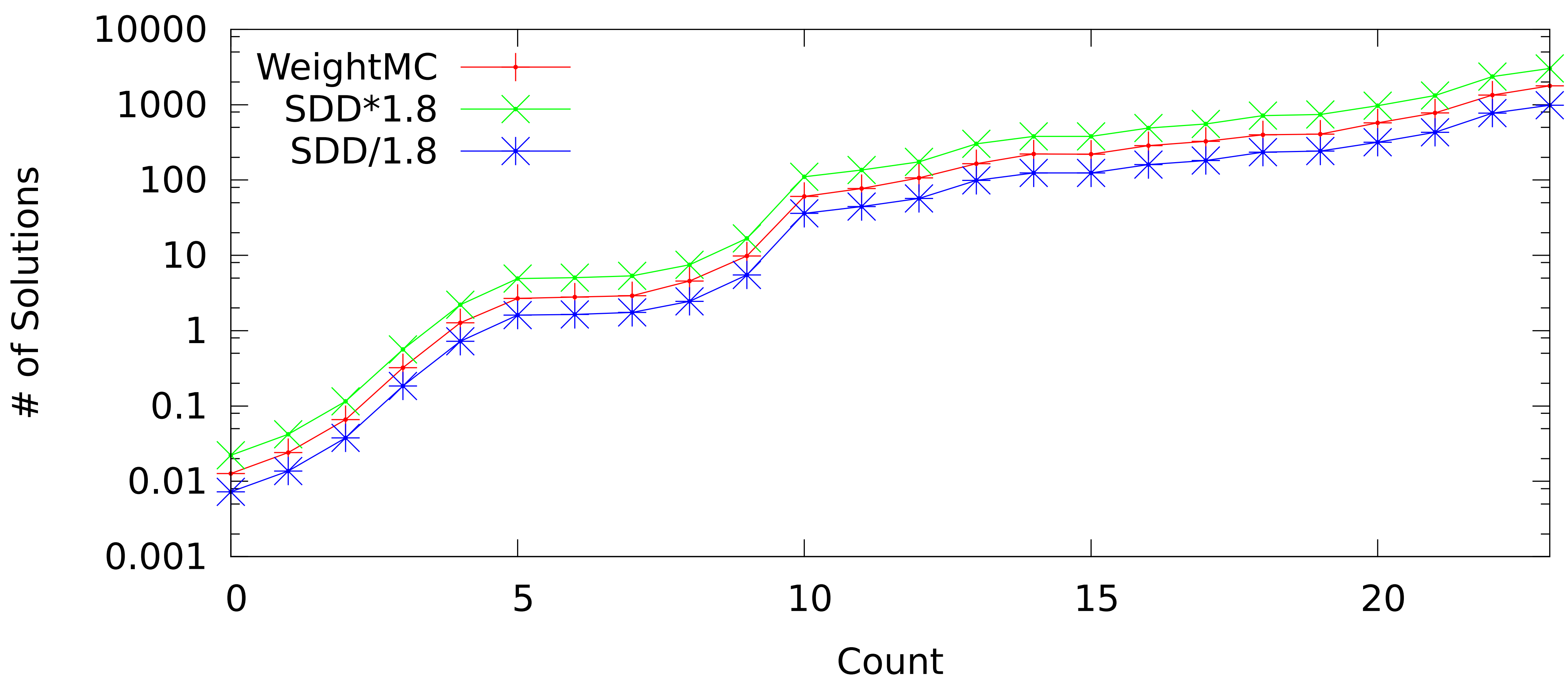
- **Complexity:** # of calls to SAT solver is linear in $\log \delta^{-1}, |F|, 1/\varepsilon$ and polynomial in $\log \rho$

Significantly Faster than SDD



# of variables ——>

Mean Error: 4% (Allowed: 80%)



Outline

- Reduction to SAT
- Partition-based techniques via (unweighted) model counting
- Extension to Weighted Model Counting
- Looking forward

Distribution-Aware Sampling

Given:

- CNF Formula F, Solution Space: R_F
- Weight Function W(.) over assignments
  - $W(\sigma)$

Problem (Sampling):

Pr (Solution y is generated) = $W(y)/W(R_F)$

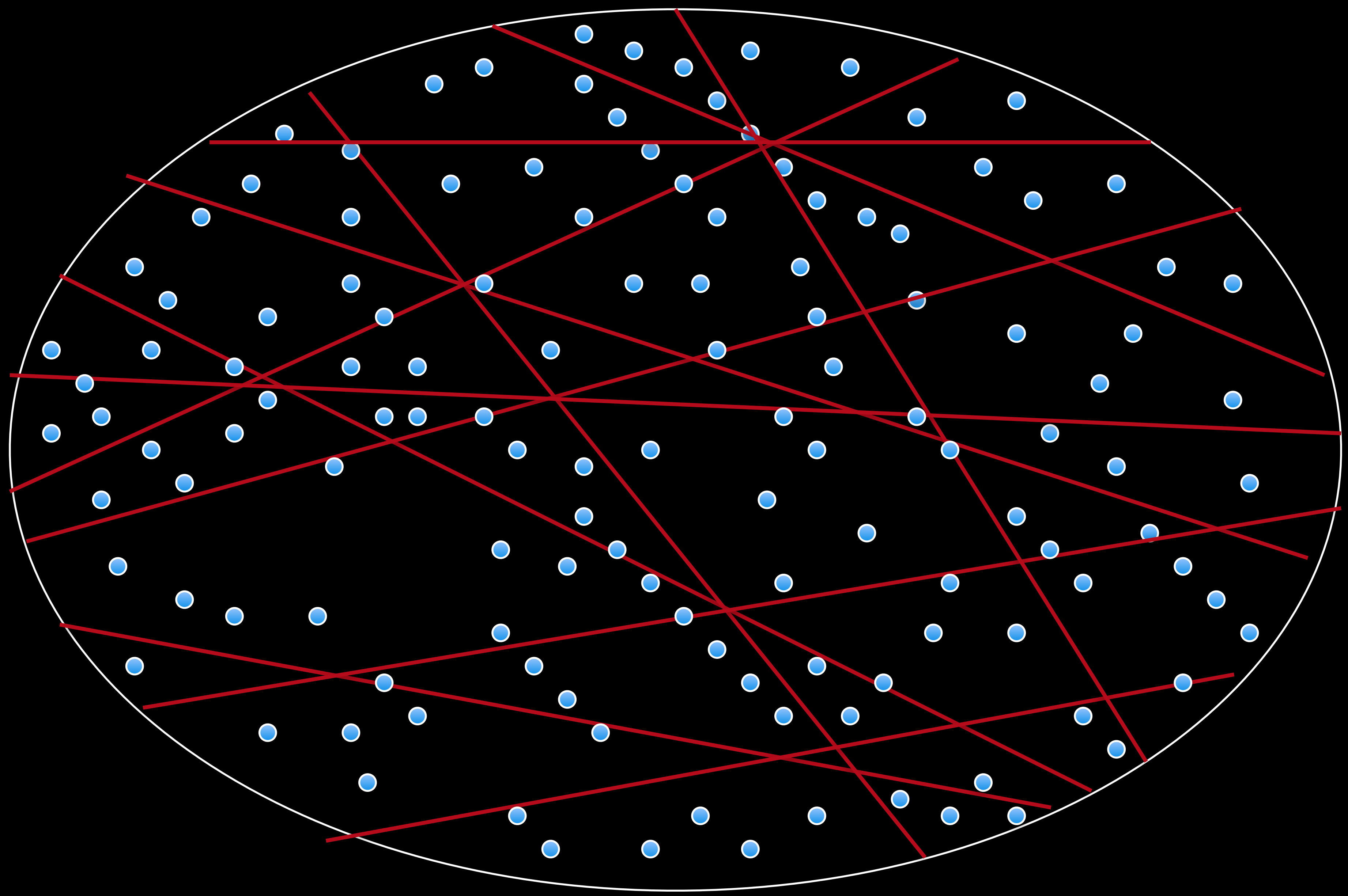
Example:

$F = (a \vee b)$; $R_F = \{[0,1], [1,0], [1,1]\}$

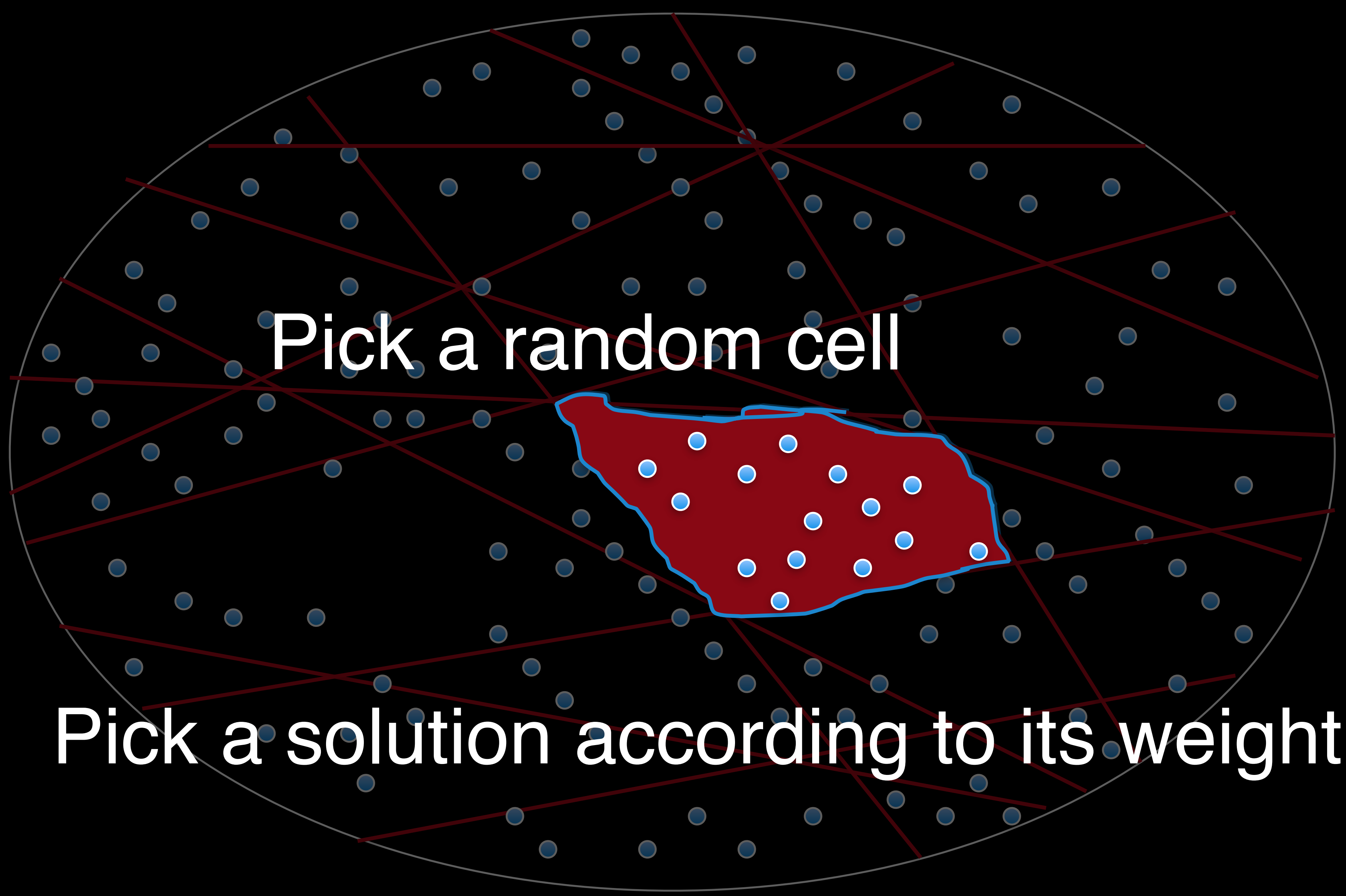
$W([0,1]) = W([1,0]) = 1/3$ $W([1,1]) = W([0,0]) = 1/6$

Pr ([0,1] is generated] = (1/3) / (5/6) = 2/5

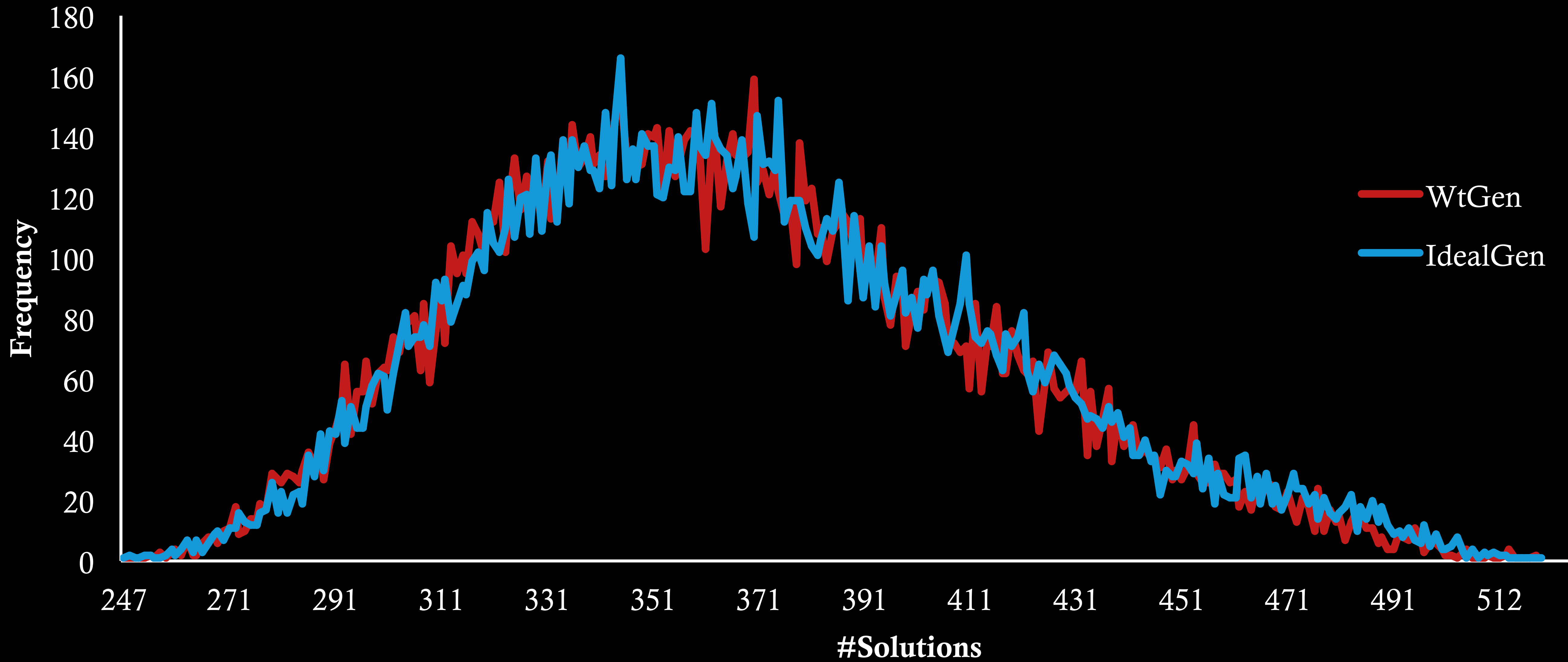
Partitioning into equal (weighted) “small” cells



Partitioning into equal (weighted) “small” cells



Sampling Distribution



- Benchmark: case110.cnf; #var: 287; #clauses: 1263
- Total Runs**: 4x10^{6};** Total Solutions **: 16384**

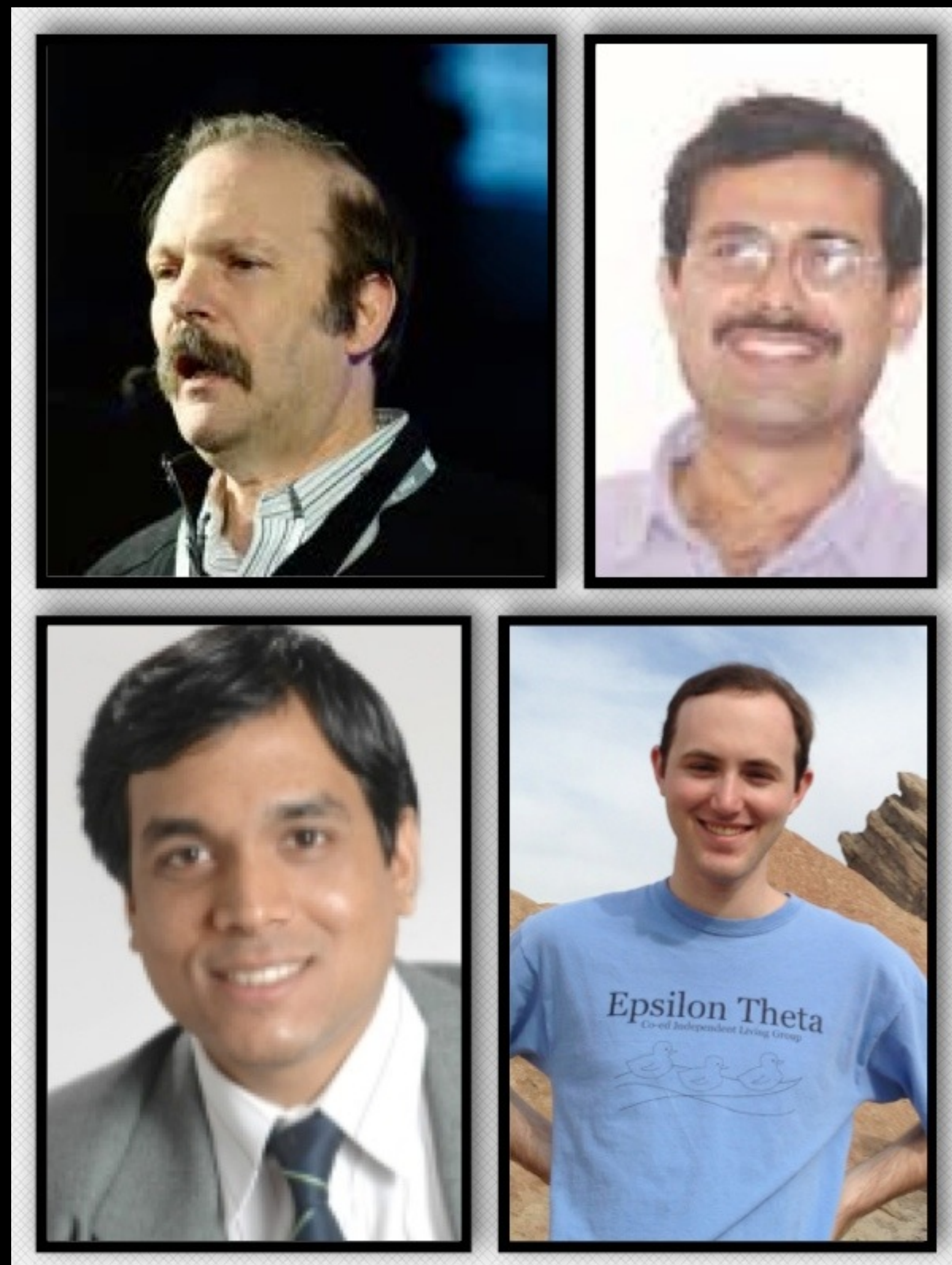
Tackling Tilt

- What kind of problems have low tilt?
- How to handle CNF+PBO+XOR
  - Current PBO solvers can’t handle XOR
  - CMS can’t handle PBO queries

Extension to More Expressive Domains (SMT, CSP)

- Efficient 3-independent hashing schemes
  - Extending bit-wise XOR to SMT provides guarantees but no advantage of SMT progress
- Solvers to handle F + Hash efficiently
  - CryptoMiniSAT has fueled progress for SAT domain
  - Similar solvers for other domains?

Collaborators



EXTRA SLIDES

Complexity

- Tilt captures the ability of hiding a large weight solution.
- Is it possible to remove <u>tilt</u> from complexity?

Exploring CNF+XOR

- Very little understanding as of now
- Can we observe phase transition?
- Eager/Lazy approach for XORs?
- How to reduce size of XORs further?

Outline

- Reduction to SAT
- Partition-based techniques via (unweighted) model counting
- Extension to Weighted Model Counting
- <u>Discussion on hashing</u>
- Looking forward

XOR-Based Hashing

- 3-universal 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 equate to 0/1 with prob. ½
- $X_1+X_3+X_6+\ldots.\ X_{n-1} = 0$ (Cell ID: 0/1)
- m XOR equations -> 2^m cells
- The cell: F && XOR (CNF+XOR)

XOR-Based Hashing

- CryptoMiniSAT: Efficient for CNF+XOR
- Avg Length : n/2
- Smaller the XORs, better the performance

How to shorten XOR clauses?

Independent Variables

- Set of variables such that assignments to these uniquely determine assignments to rest of variables for formula to be true
- (a V b = c) ➔ Independent Support: {a, b}
- # of auxiliary variables introduced: 2-3 orders of magnitude
- Hash only on the independent variables (huge speedup)