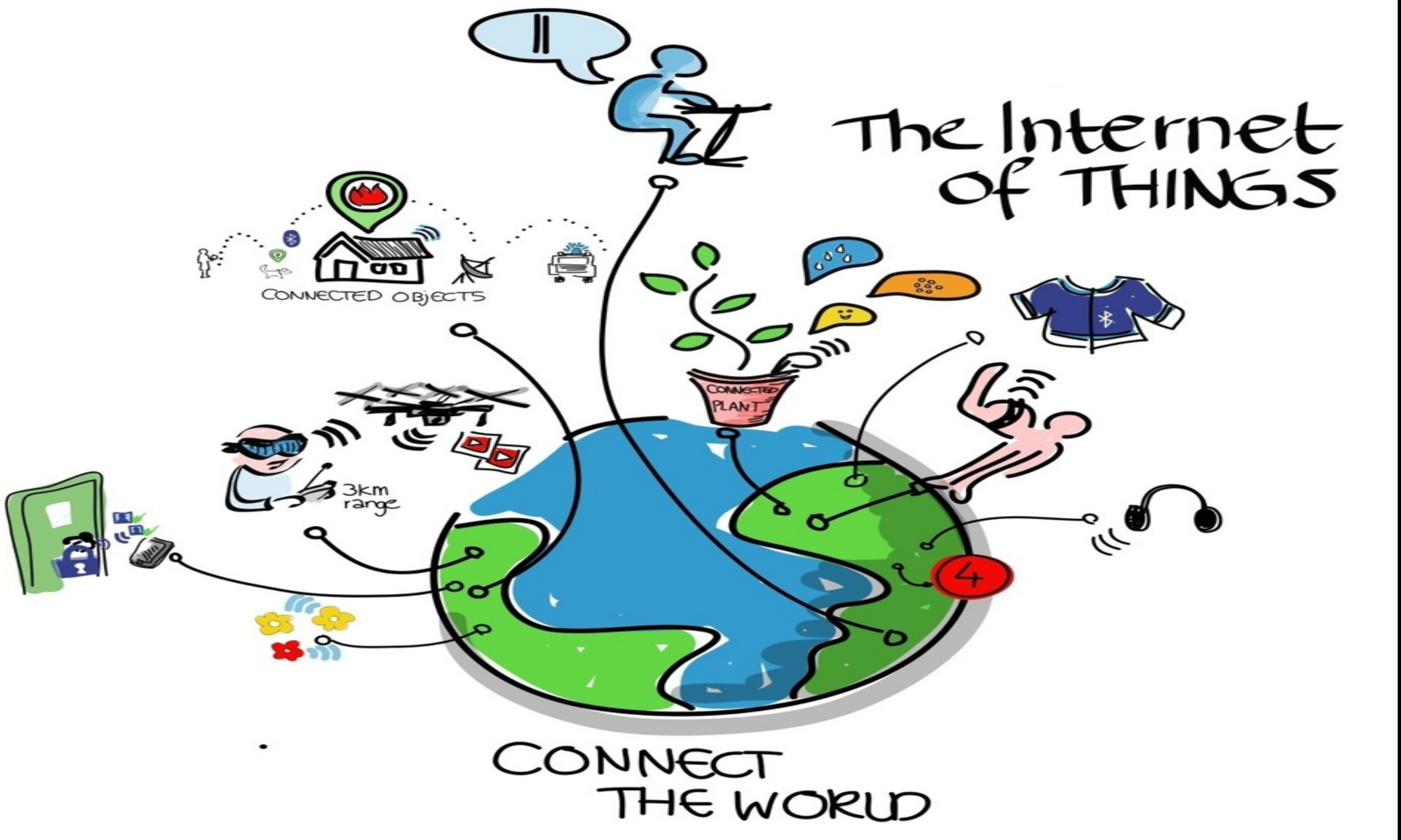
Approximating Probabilistic Inference

Kuldeep S. Meel

PhD Student
CAVR Group

Joint work with Supratik Chakraborty (IITB), Daniel J. Fremont (UCB), Sanjit A. Seshia (UCB), Moshe Y. Vardi (Rice)

IoT: Internet of Things



The Era of Data

How to make inferences from data

Probabilistic Inference

Given that Mary (aged 65) called 911, what is the probability of the burglary in the house?

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

Probabilistic Inference

Given that Mary (aged 65) called 911, what is the probability of the burglary in the house?

$$\Pr[\text{event} | \text{evidence}]$$

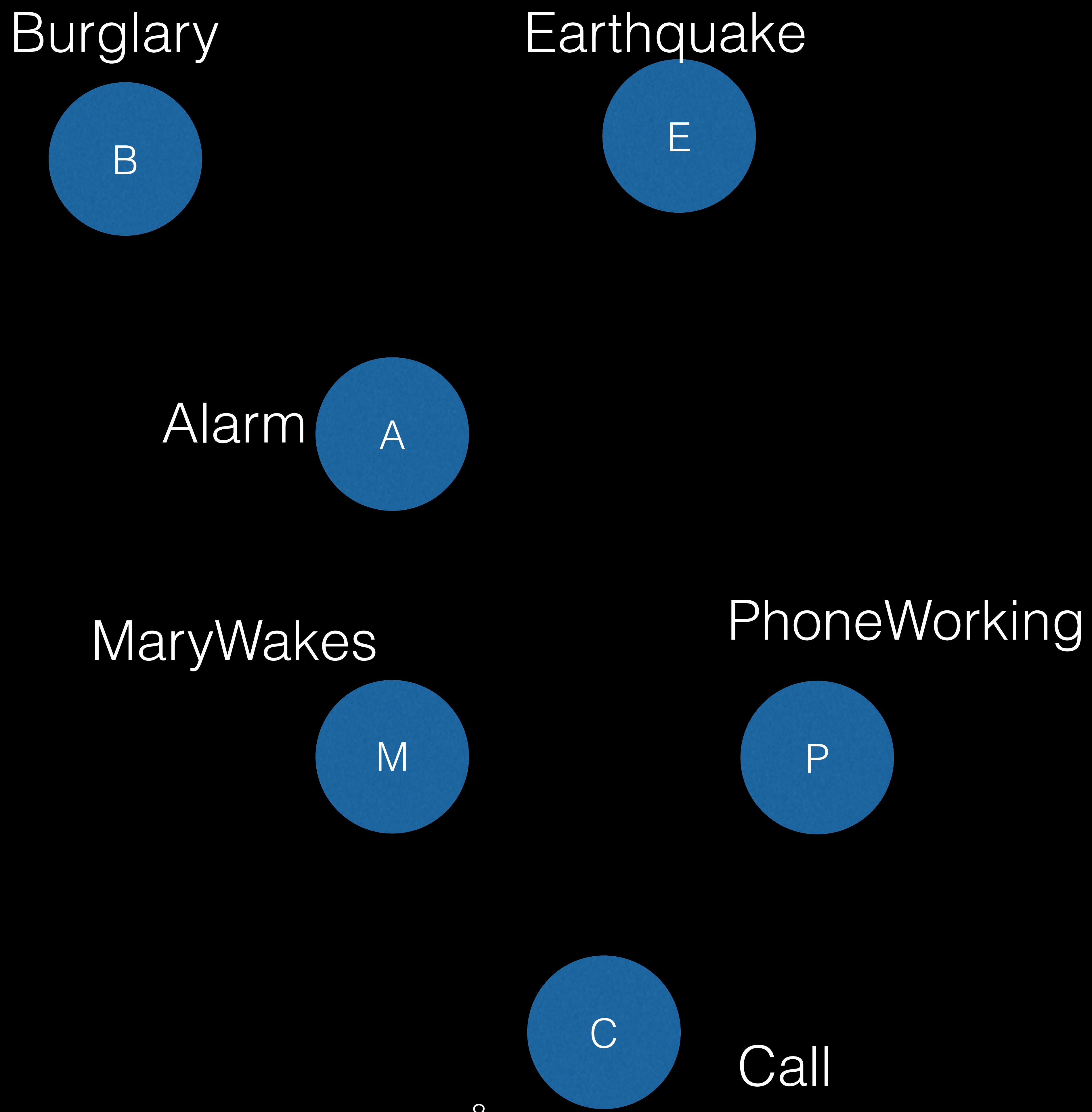
Probabilistic Inference

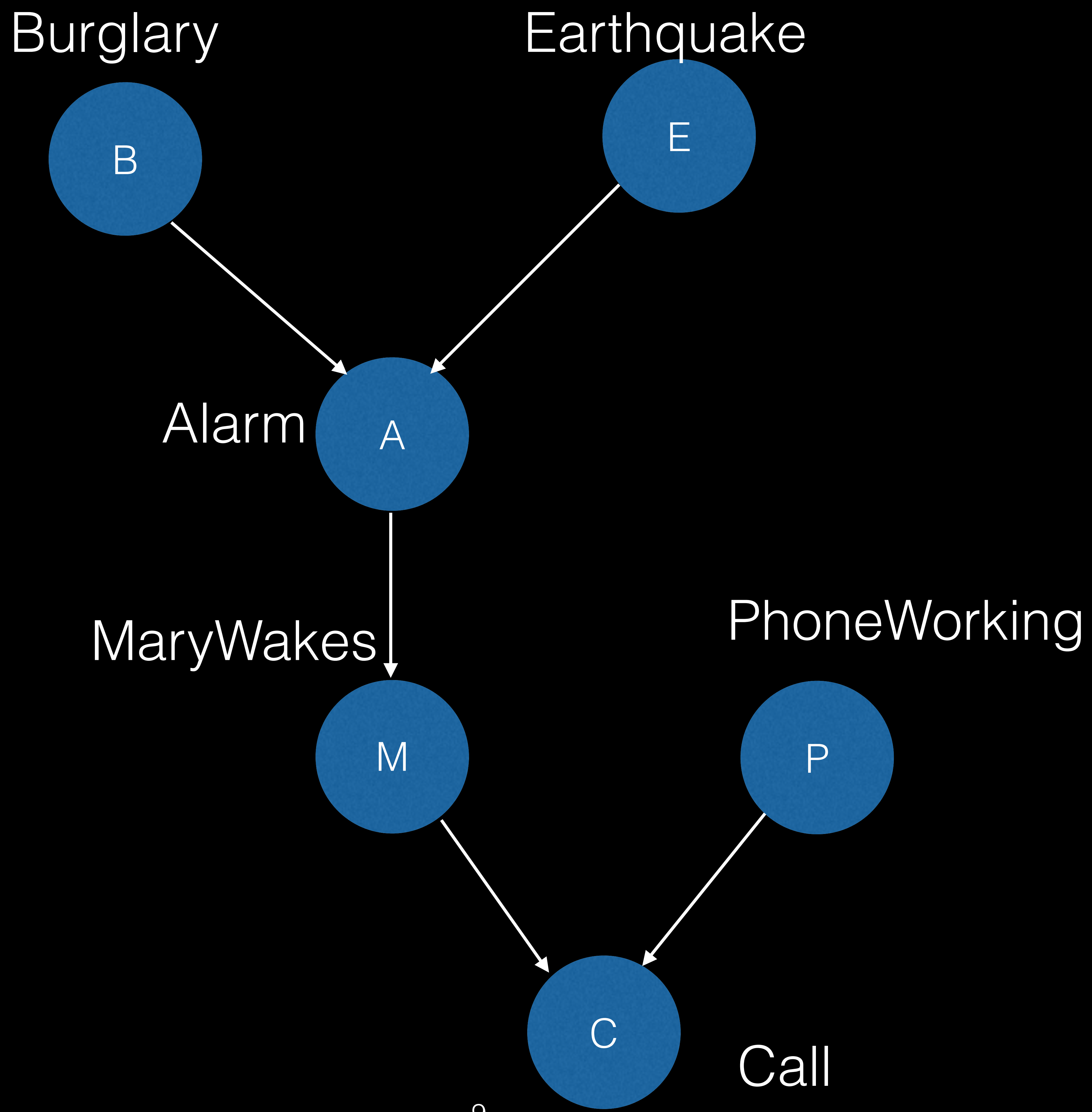
Given that Mary (aged 65) called 911, what is the probability of the burglary in the house?

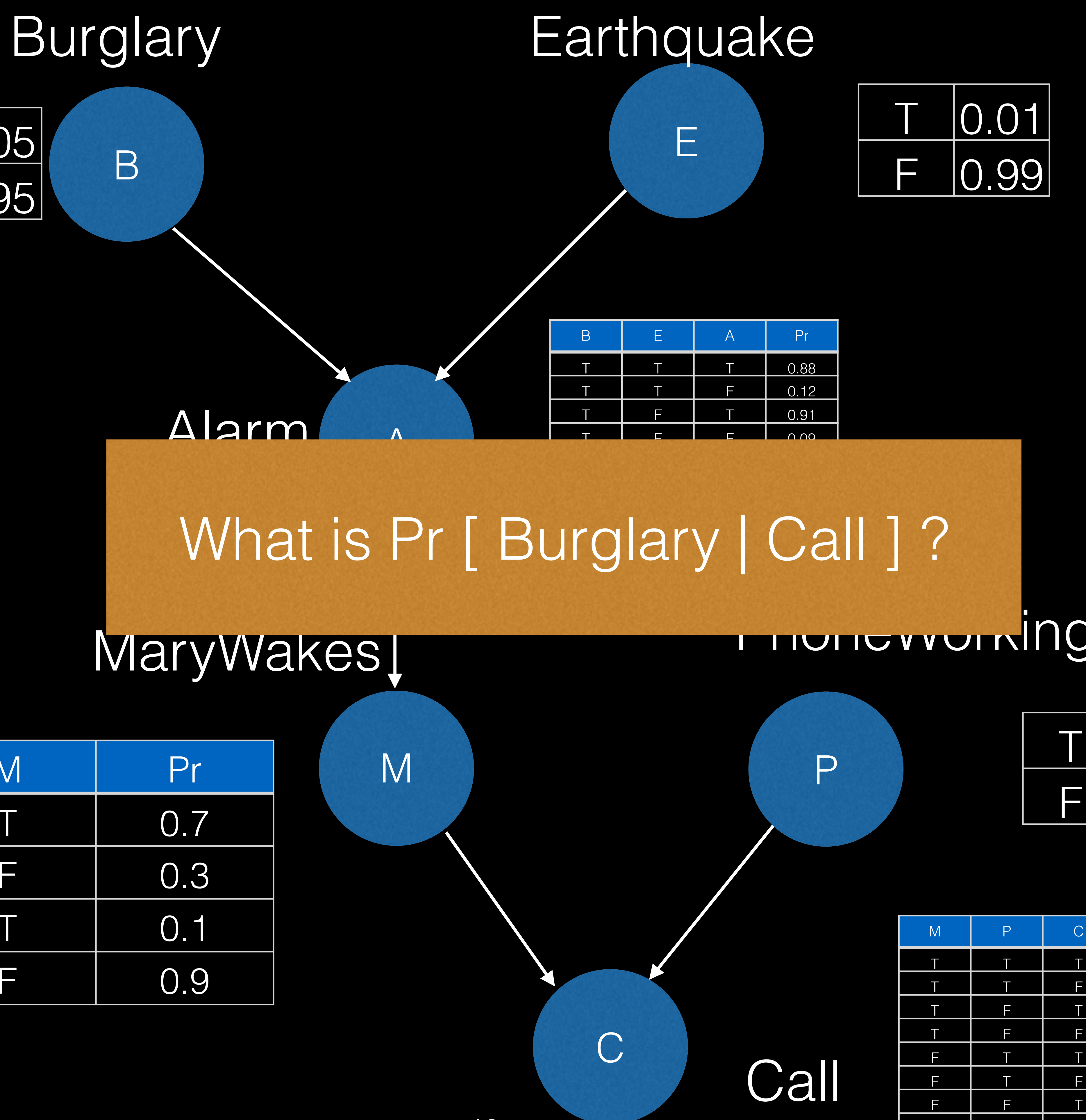
$$\Pr\,[\text{event} \mid \text{evidence}]$$

Graphical Models

Bayesian Networks


Burglary
B
Earthquake
E
Alarm
A
MaryWakes
M
PhoneWorking
P
C
Call


Burglary
Earthquake
B
E
Alarm
A
MaryWakes
PhoneWorking
M
P
C
Call



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

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

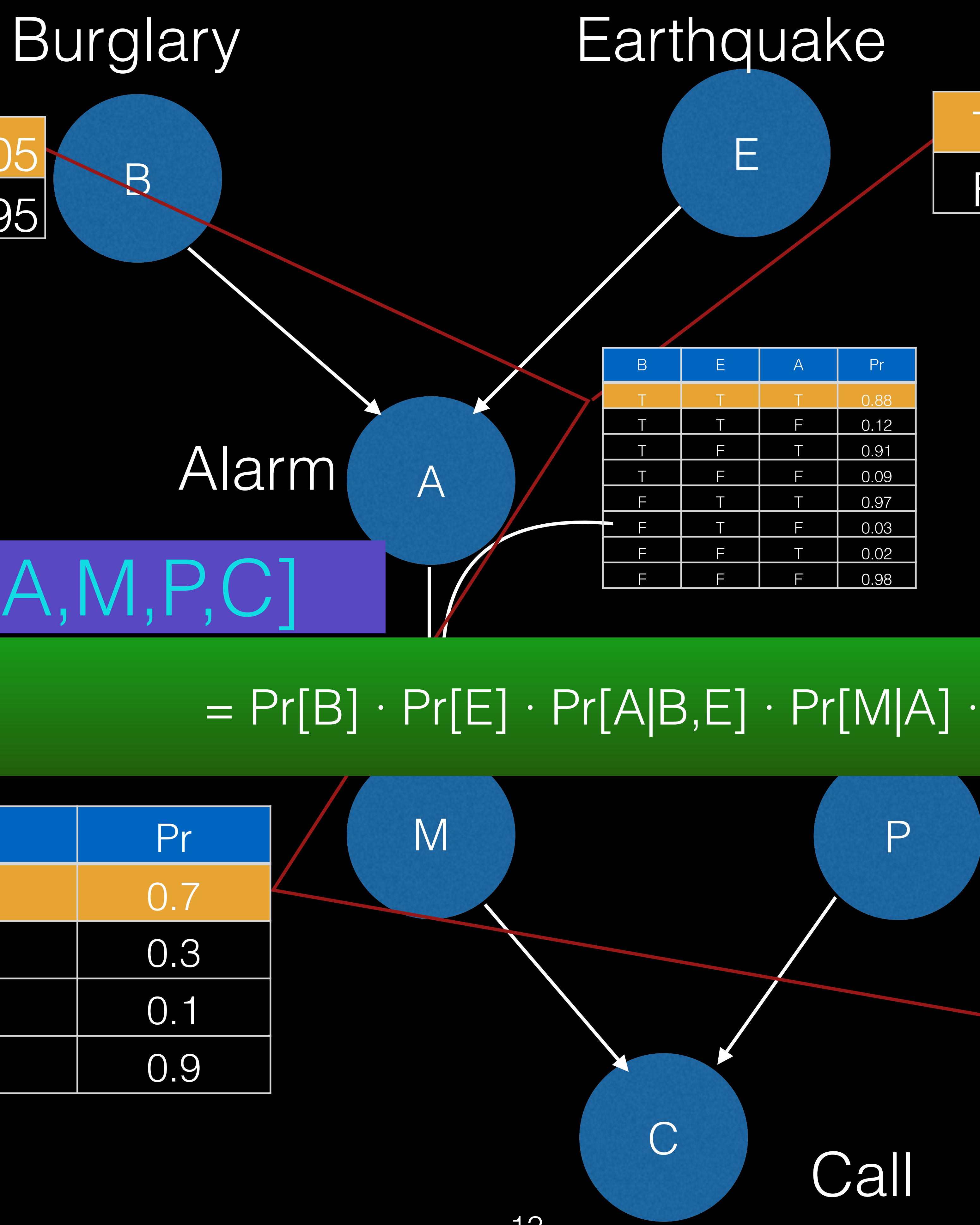
Bayes' Rule to the Rescue

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

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

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

$$= Pr[B] \cdot Pr[E] \cdot Pr[A|B,E] \cdot Pr[M|A] \cdot Pr[C|M,P]$$



T	0.05
F	0.95

T	0.01
F	0.99

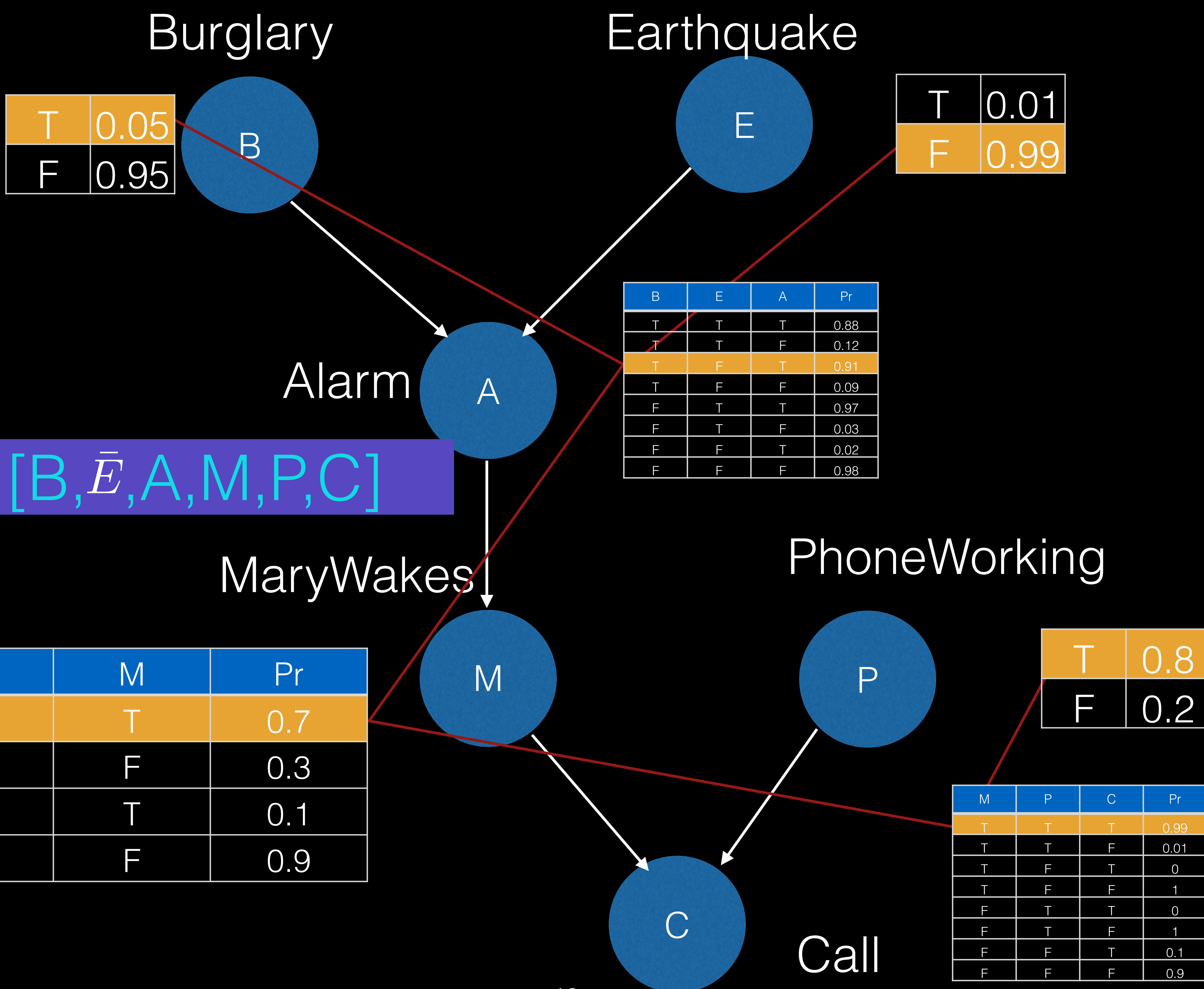
B	E	A	Pr
T	T	T	0.88
T	T	F	0.12
T	F	T	0.91
T	F	F	0.09
F	T	T	0.97
F	T	F	0.03
F	F	T	0.02
F	F	F	0.98

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

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



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

So we are done?

- We can enumerate all paths
- Compute probability for every path
- Sum up all the probabilities

Where is the catch?

Exponential number of paths

Prior Work


BP, MCMC
C ? f
Scalability
C = f
Exact Methods
Quality/Guarantees

Our Contribution

Approximation Guarantees

Input: ϵ , δ

Output: C

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

An Idea for a new paradigm?

Partition space of paths into
“small” and “equal weighted” cells

# of paths in a cell is not large (bounded by a constant)

“equal weighted”: All the cells have equal weight

Outline

- Reduction to SAT
- 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)

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

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

$\mathbf{|R_F| = 3}$

Weighted Model Counting

Given:

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

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 ∨ b); W(a=0) = 0.4; W(a = 1) = 1-0.4 = 0.6
  W(b=0) = 0.3; W(b = 1) = 0.7
- W[(0,1)] = W(a = 0) W(b = 1) = 0.4 0.7 = 0.28

Reduction to W-SAT

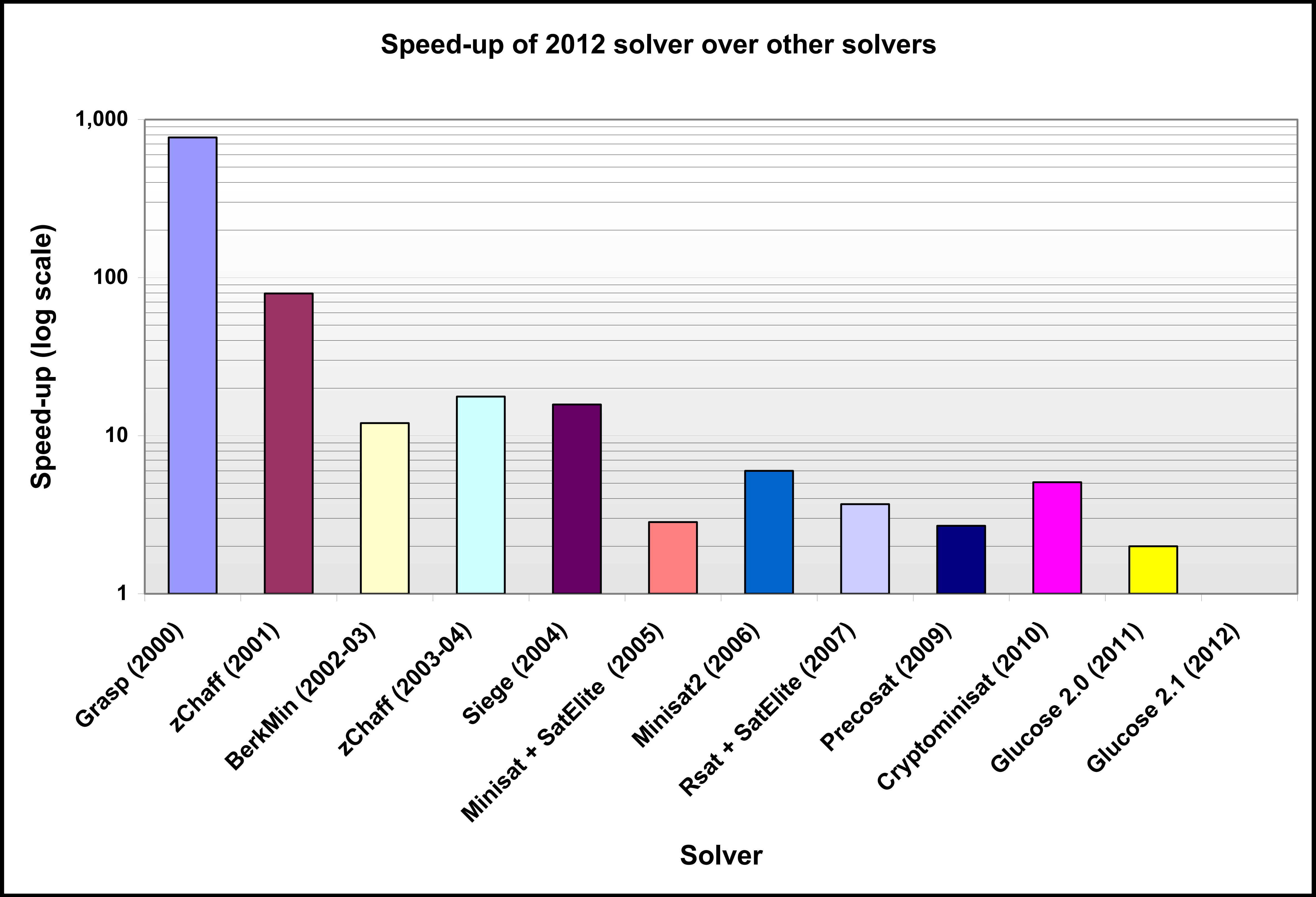
Bayesian Network	SAT Formula
Nodes	Variables
Rows of CPT	Variables
Probabilities in CPT	Weights
Event and Evidence	Constraints

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


Speed-up of 2012 solver over other solvers
Speed-up (log scale)
1,000
100
10
1
Grasp (2000)
zChaff (2001)
BerkMin (2002-03)
zChaff (2003-04)
Siege (2004)
Minisat + SatElite (2005)
Minisat2 (2006)
Rsat + SatElite (2007)
Precosat (2009)
Cryptominisat (2010)
Glucose 2.0 (2011)
Glucose 2.1 (2012)
Solver

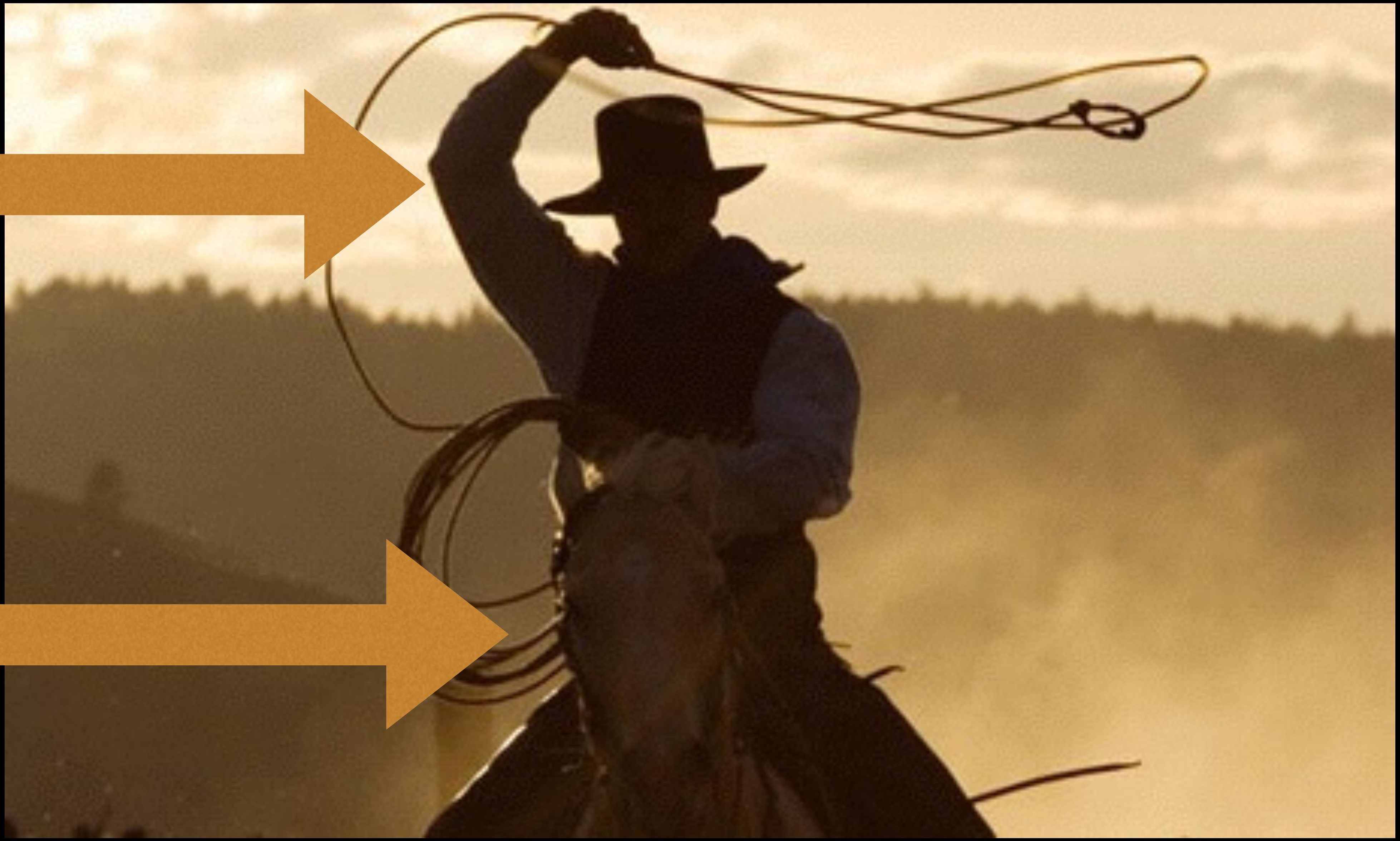
Why SAT?

- 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

Riding the SAT revolution

Probabilistic
Inference

SAT



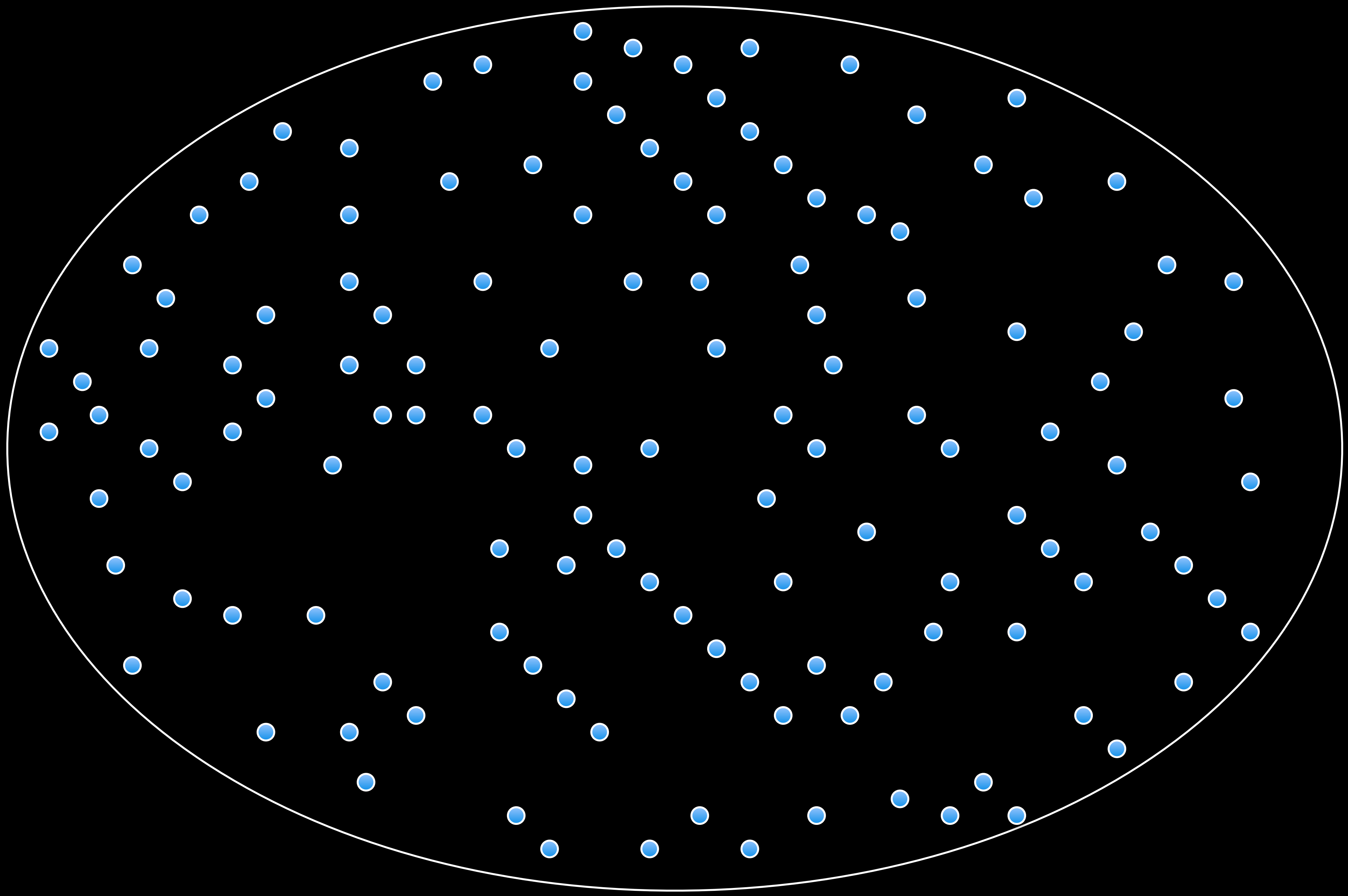
Where is the catch?

- Model counting is very hard (#P hard)
  - #P: Harder than whole polynomial hierarchy
- Exact algorithms do not scale to large formulas
- Approximate counting algorithms do not provide theoretical guarantees

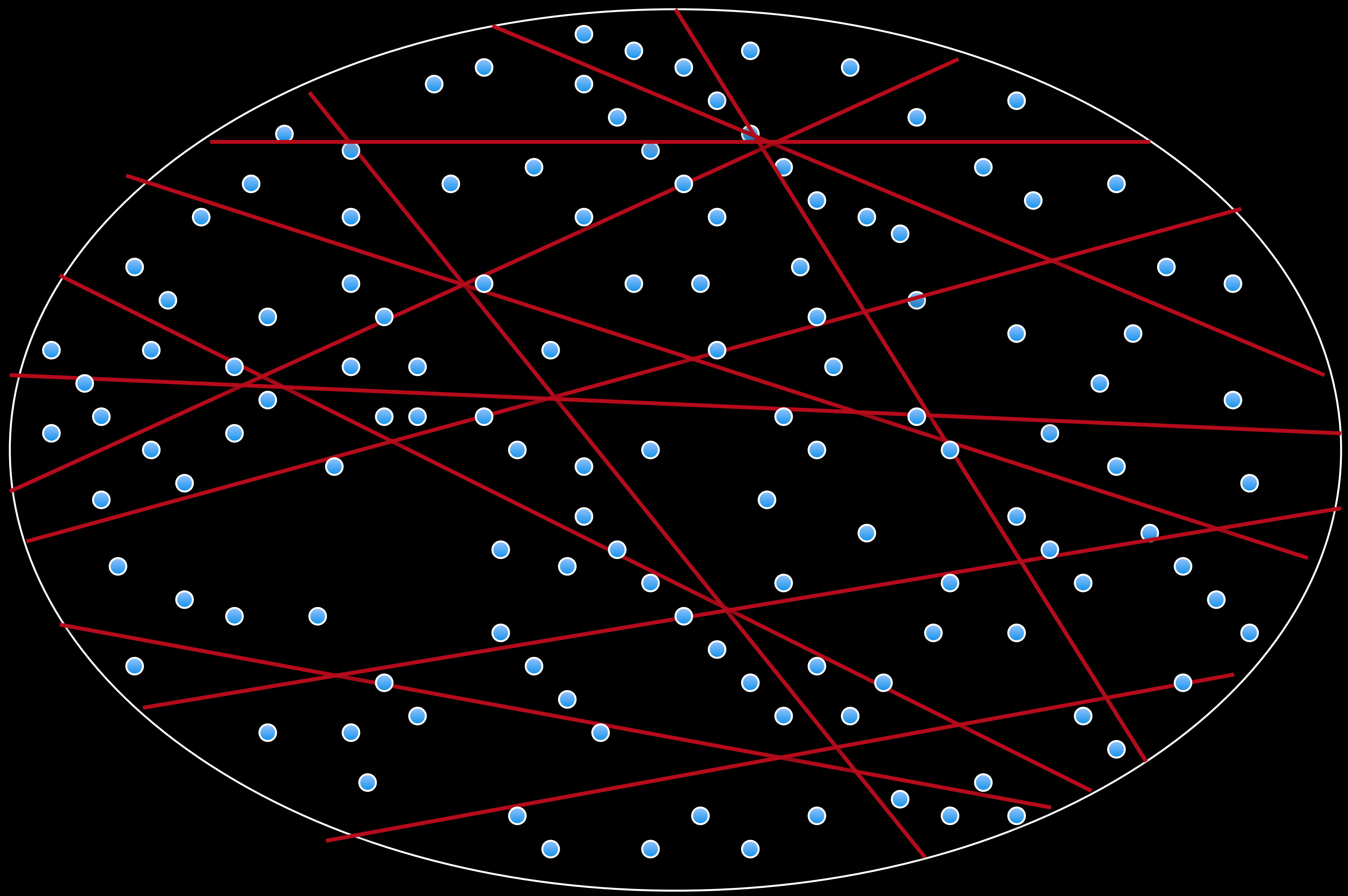
Outline

- Reduction to SAT
- <u>Approximate Weighted Model Counting</u>
- Looking forward

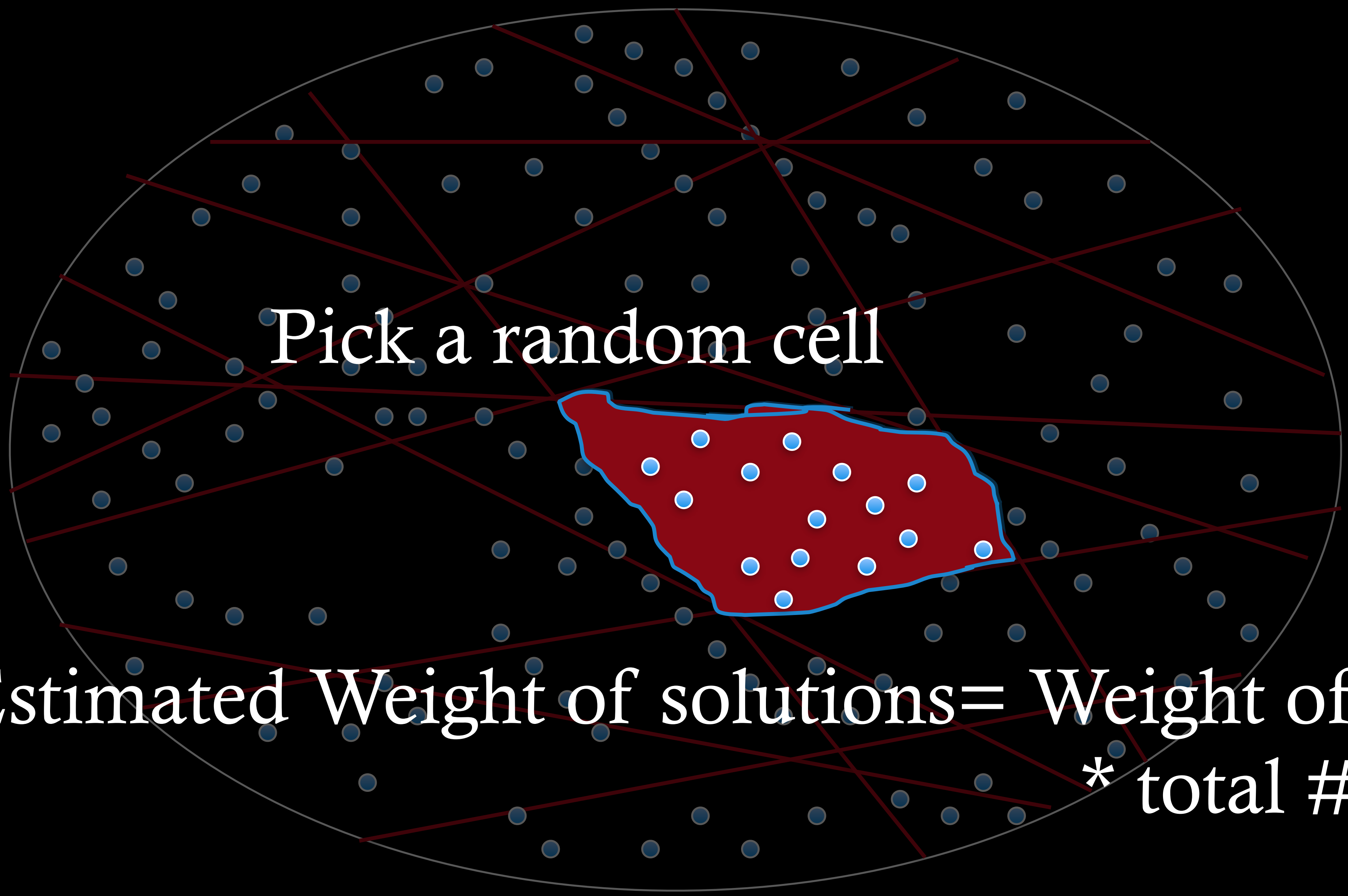
Counting through Partitioning



Counting through Partitioning



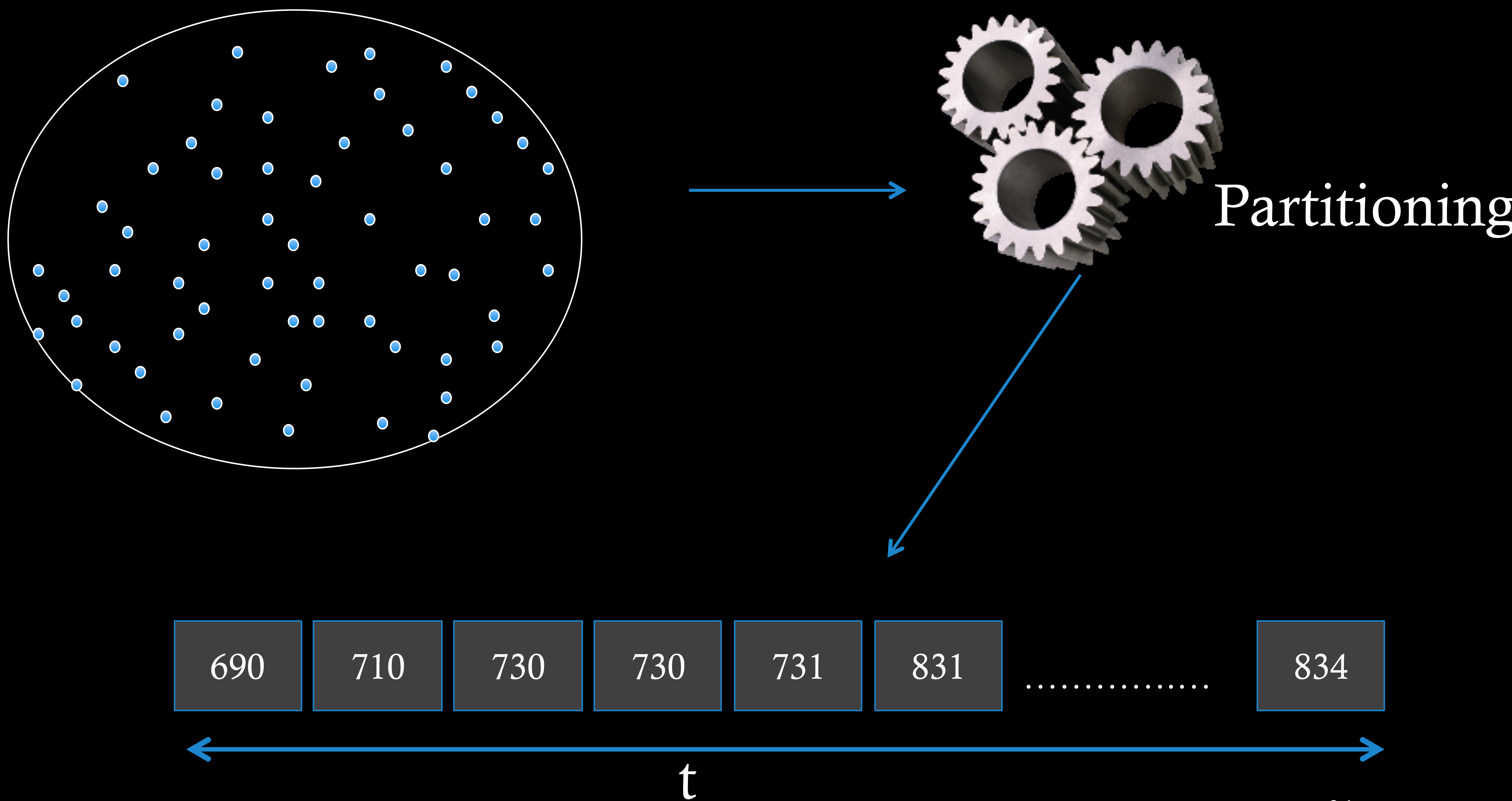
Counting through Partitioning



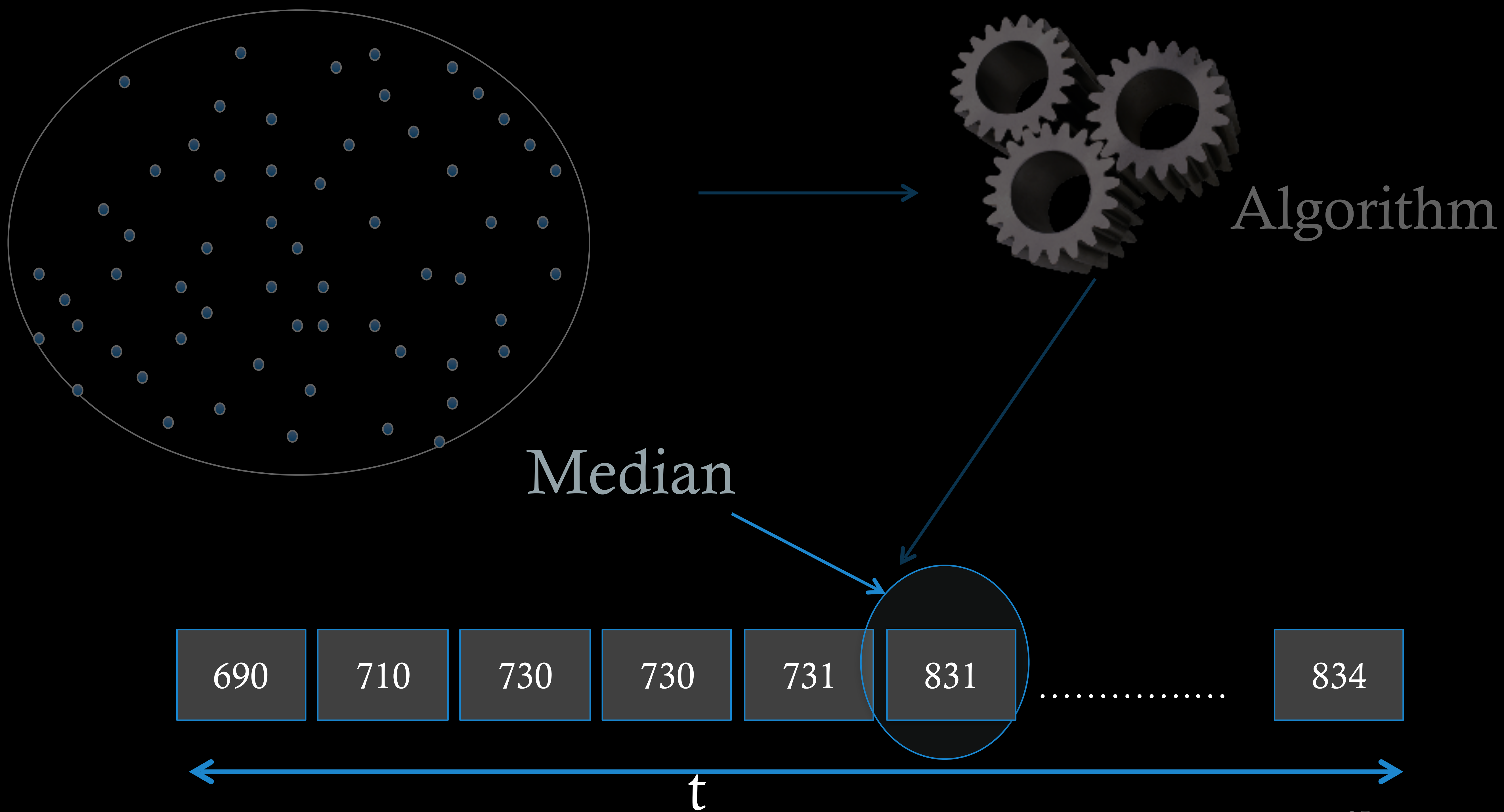
Pick a random cell

Estimated Weight of solutions= Weight of the cell
* total # of cells

Approximate Counting



Scaling the confidence



How to Partition?

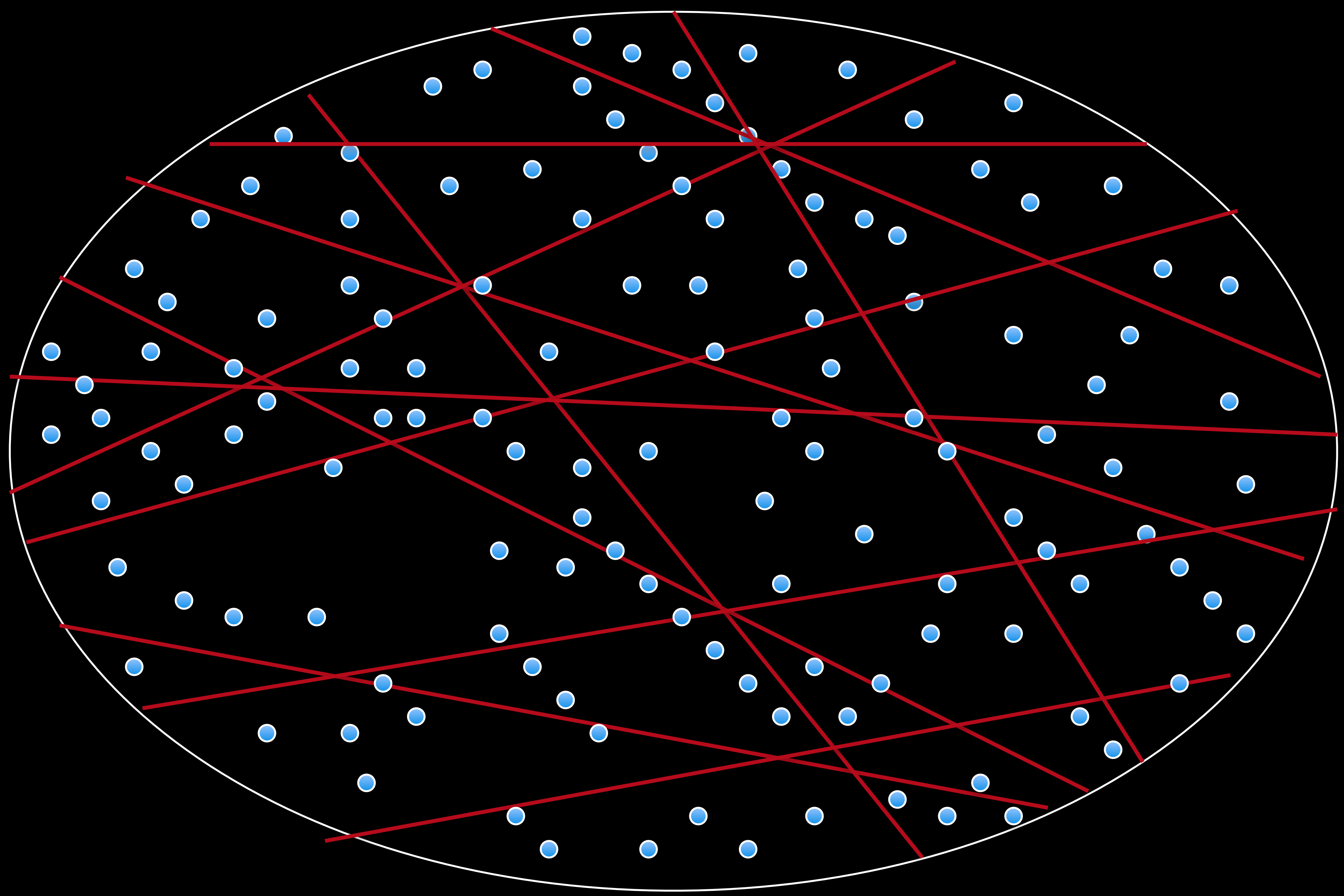
How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

3-Universal Hashing
[Carter-Wegman 1979, Sipser 1983]

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 the cells should be?

Size of cell

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

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

Dependence on distribution

- Normalized weight of a solution y = $W(y)/W_{max}$
- Maximum weight of a cell = pivot
- Maximum # of solutions in cell = $pivot*W_{max}/W_{min}$
- Tilt = Wmax/Wmin

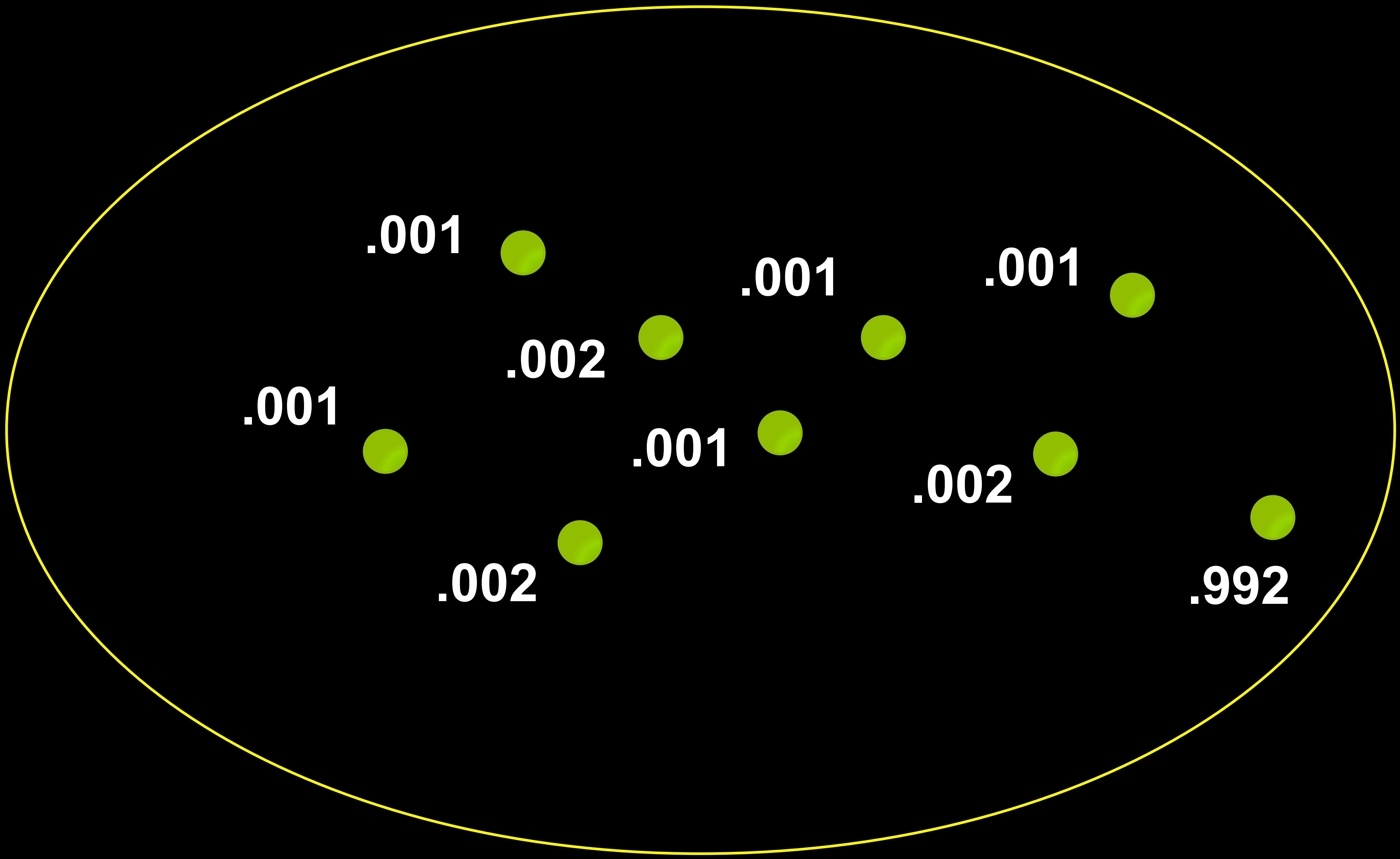
Strong Theoretical Guarantees

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

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

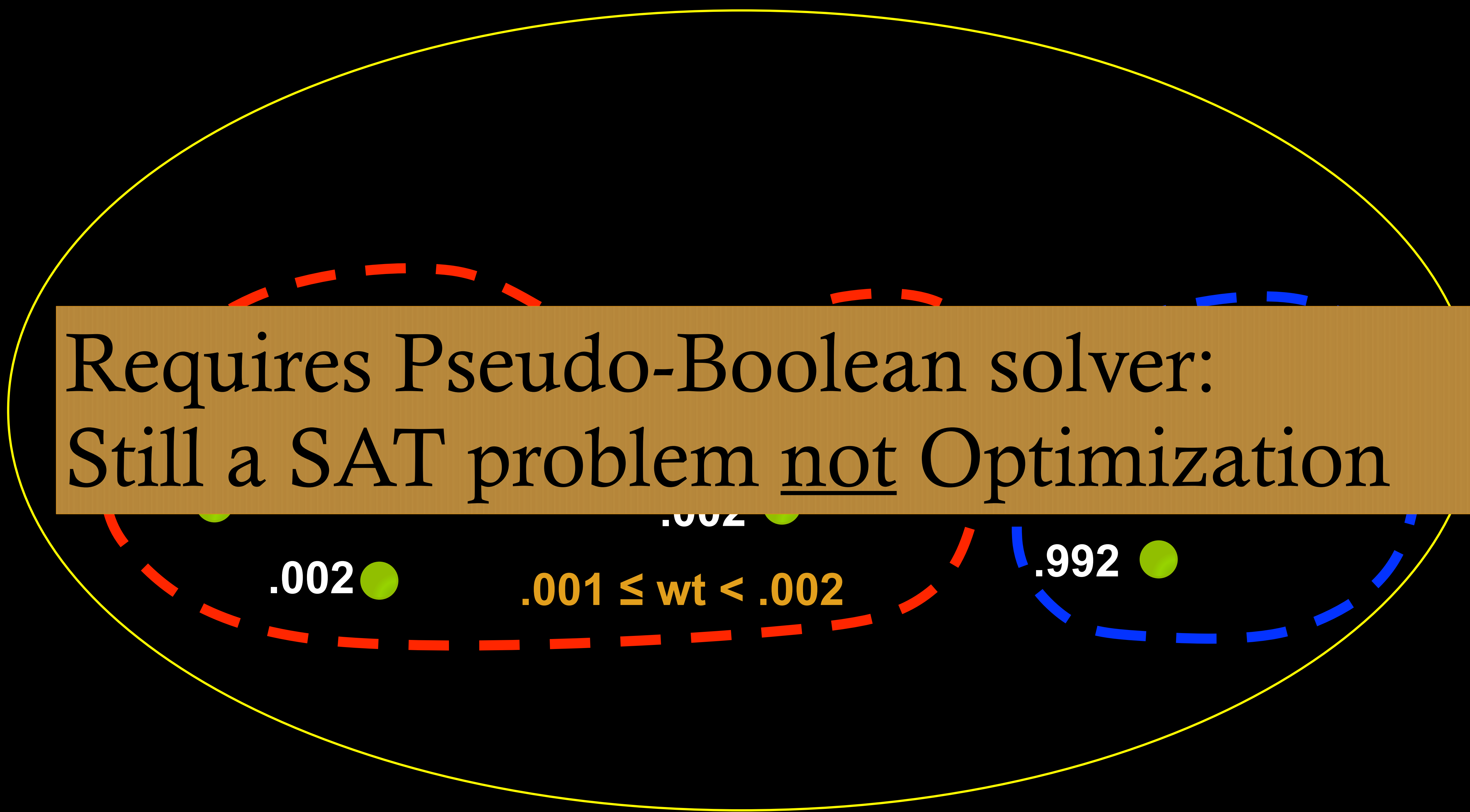
- **Complexity:** # of calls to SAT solver is linear in ρ and polynomial in $\log \delta^{-1}, |F|, 1/\varepsilon$

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

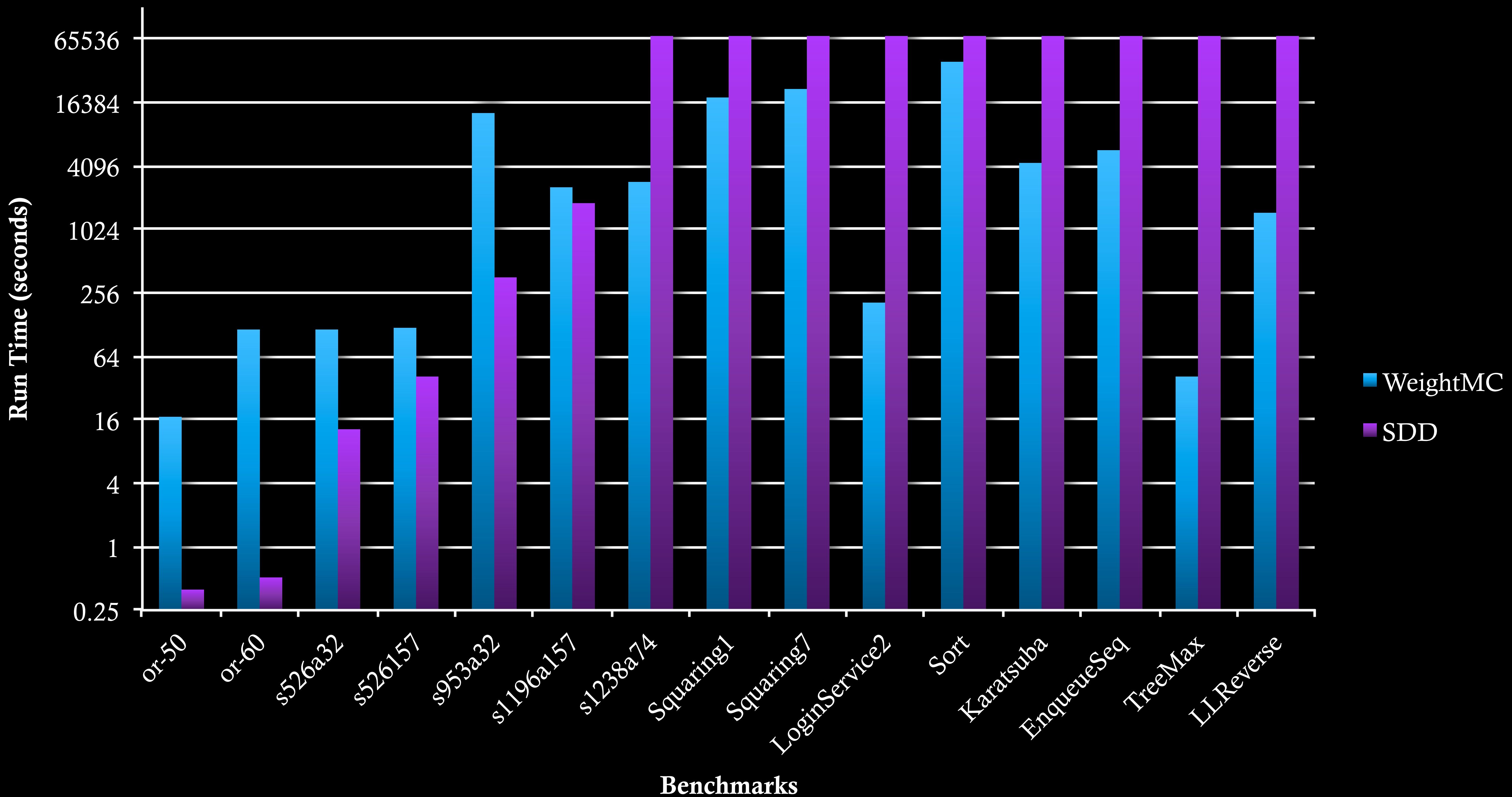
Strong Theoretical Guarantees

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

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

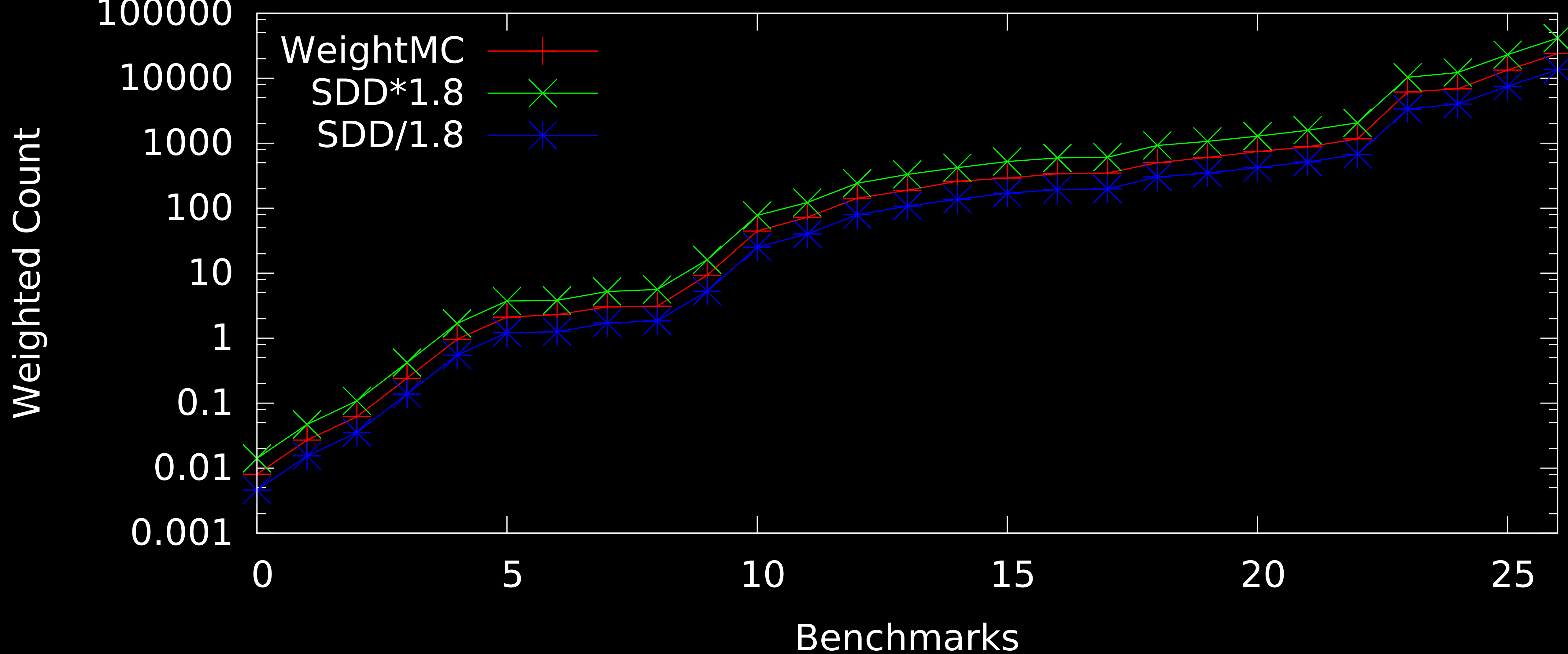
- **Complexity:** # of calls to SAT solver is linear in $\log \rho$ and polynomial in $\log \delta^{-1}, |F|, 1/\varepsilon$

Significantly Faster than SDD



# of variables ——>

Mean Error: 4% (Allowed: 80%)



Outline

- Reduction to SAT
- Weighted Model Counting
- Looking forward

Distribution-Aware Sampling

Given:

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

Problem (Sampling):

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

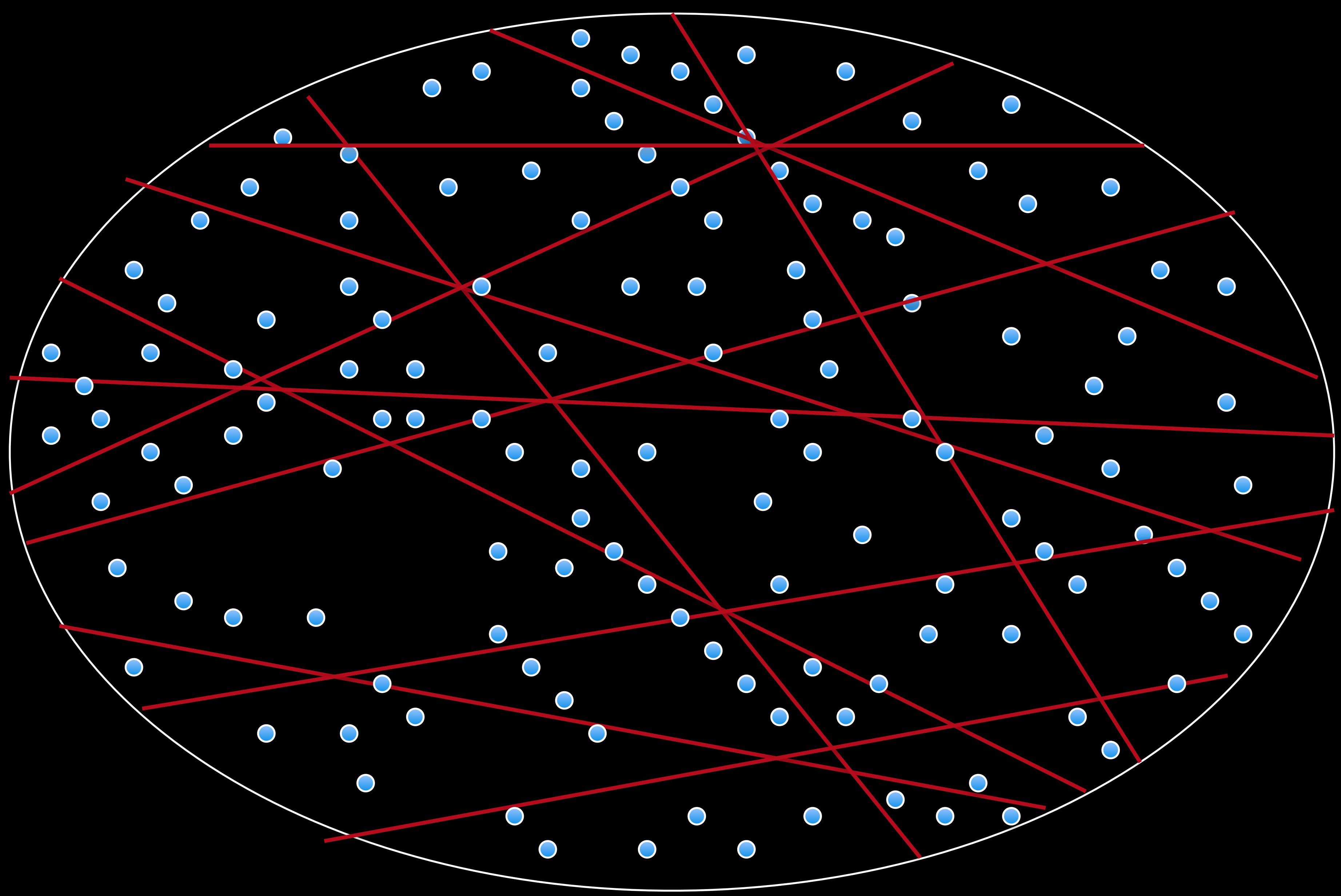
Example:

F = (a ∨ b); R_F = {[0,1], [1,0], [1,1]}

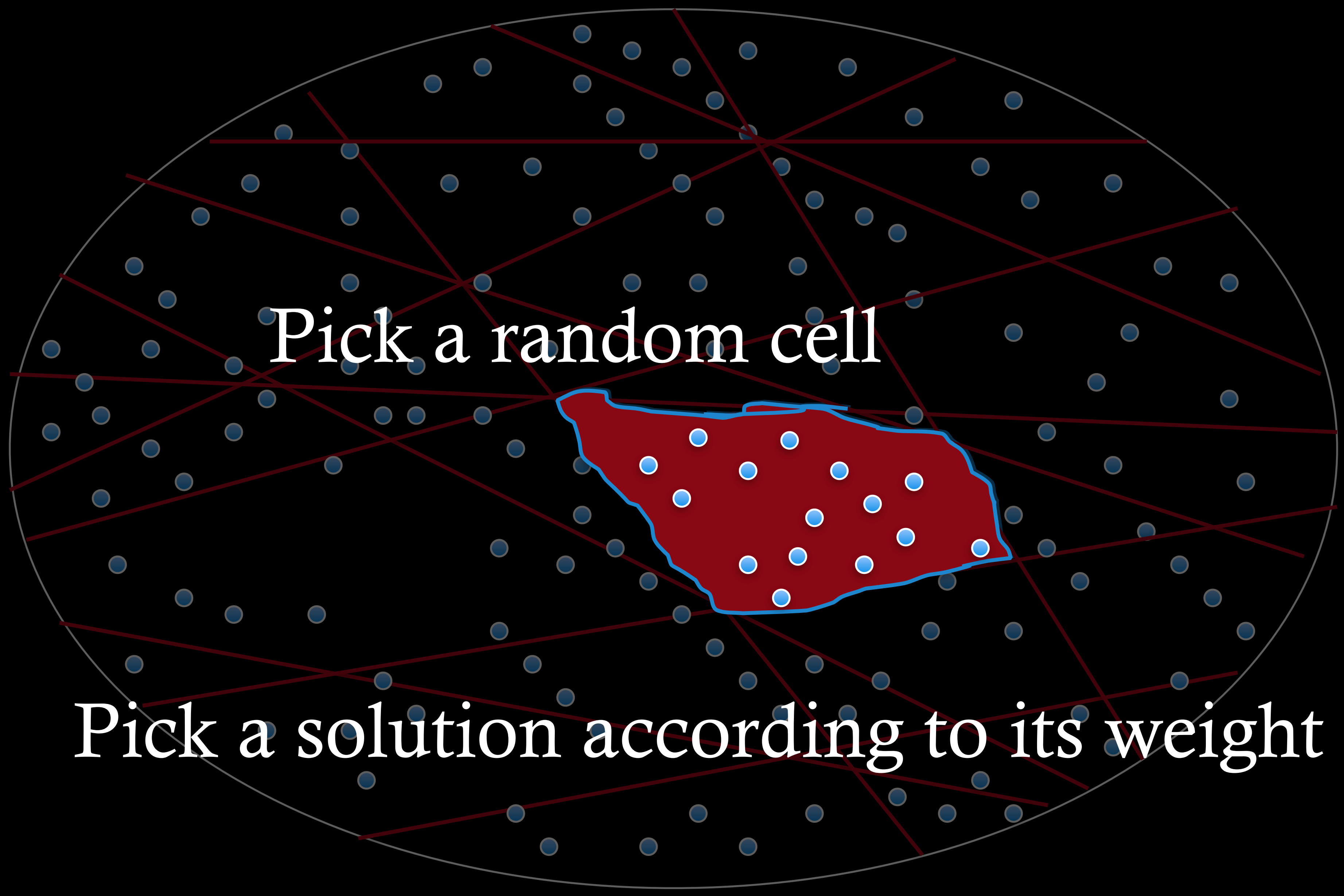
W([0,1]) = W([1,0]) = 1/3 W([1,1]) = W([0,0]) = 1/6

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

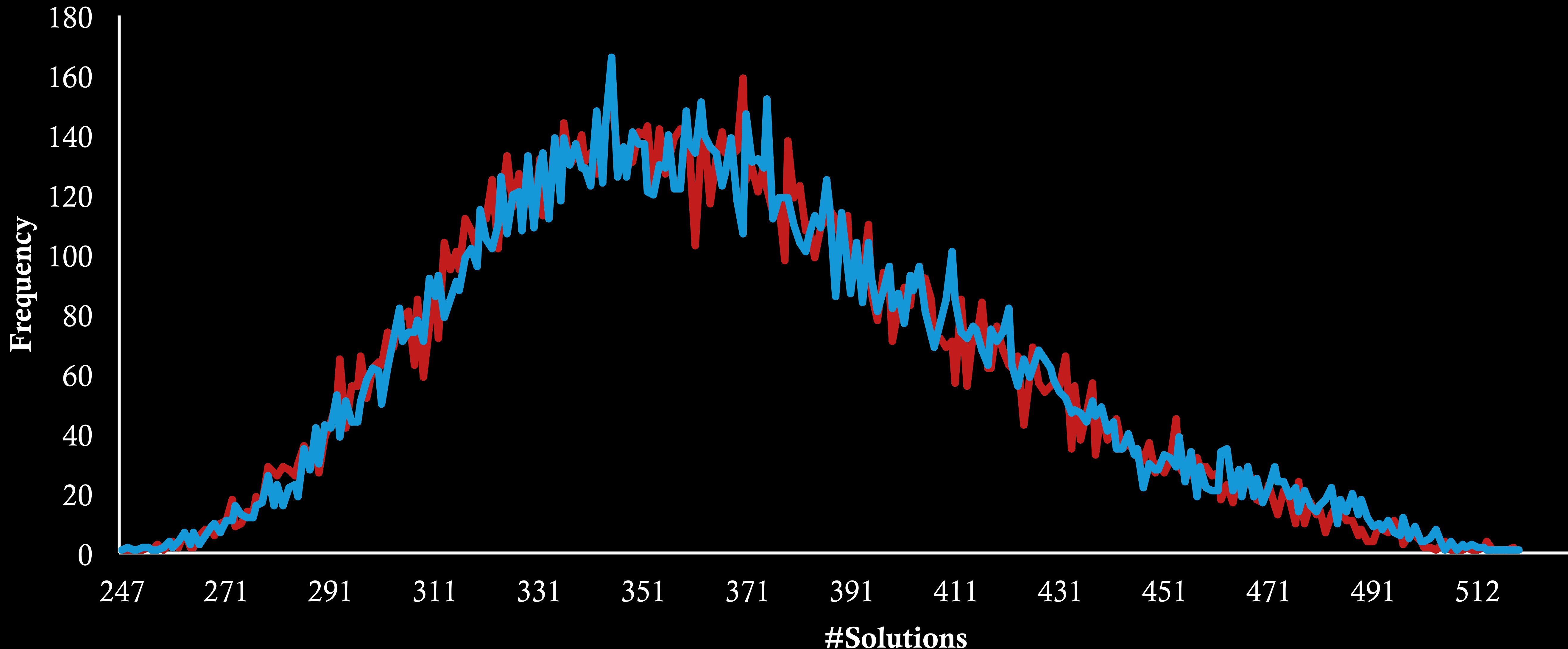
Partitioning into equal (weighted) “small” cells



Partitioning into equal (weighted) “small” cells

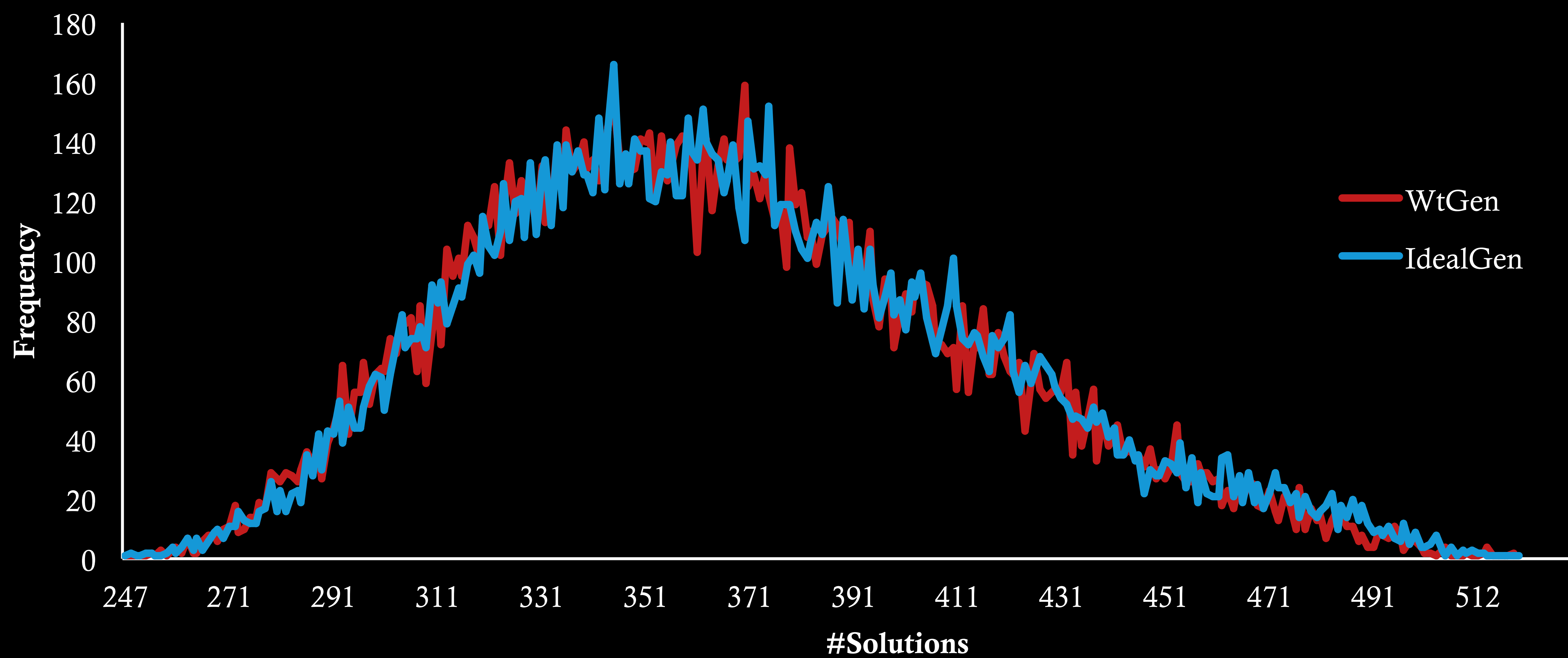


Sampling Distribution



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

Sampling Distribution



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

Classification

- What kind of problems have small tilt?
- How to predict tilt?

Tackling Tilt

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

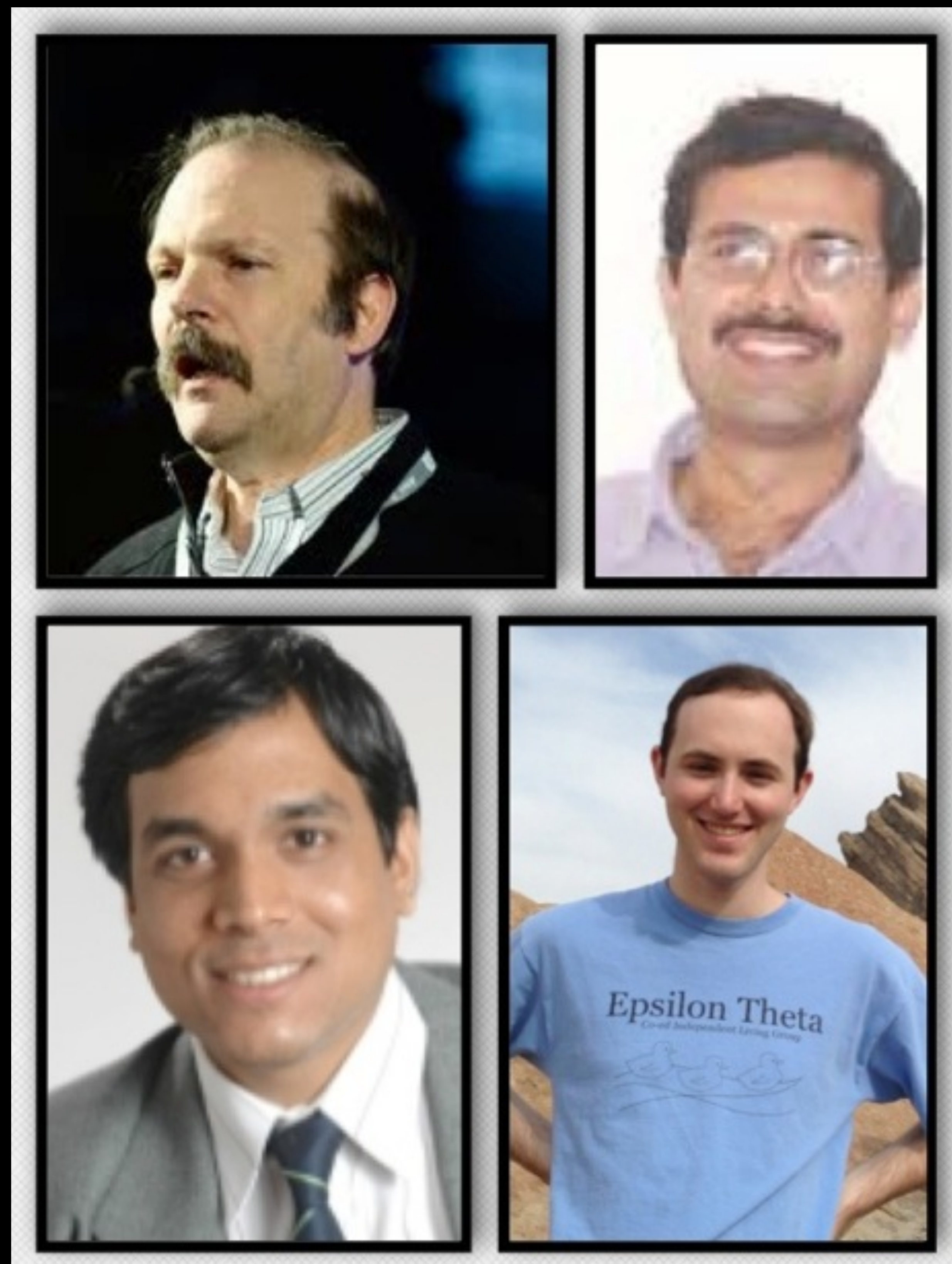
- Solvers to handle F + Hash efficiently
  - CryptoMiniSAT has fueled progress for SAT domain
  - Similar solvers for other domains?

Conclusion

- Inference is key to the Internet of Things (IoT)
- Current inference methods either do not scale or do not provide any approximation guarantees
- A novel scalable approach that provides theoretical guarantee of approximation
- Significantly better than state-of-the-art tools
- Exciting opportunities ahead!

To sum up ….

Collaborators



EXTRA SLIDES

Complexity

- Tilt captures the ability of hiding a large weight solution.
- Is it possible to remove tilt 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
- Discussion on hashing
- 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)