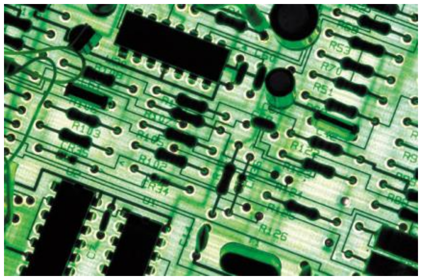
A Scalable and Nearly Uniform Generator of SAT Witnesses

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

[1]Indian Institute of Technology Bombay, India
[2]Department of Computer Science, Rice University

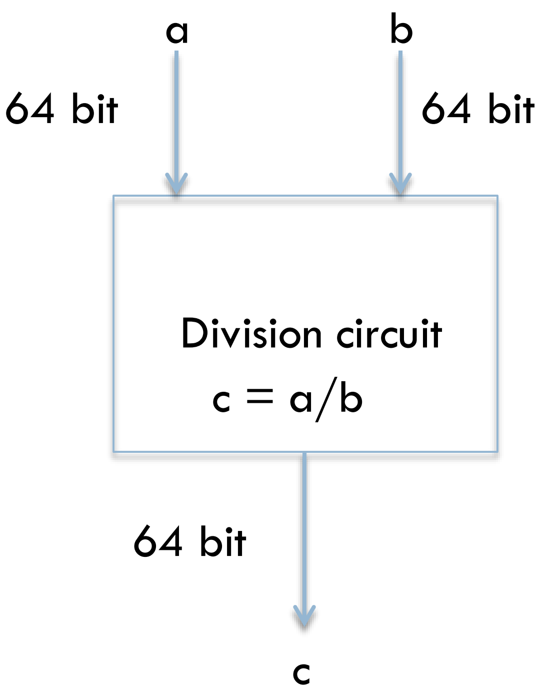
CAV 2013

Life in the 21st Century!



How do we guarantee that the systems work *<u>correctly</u>* ?

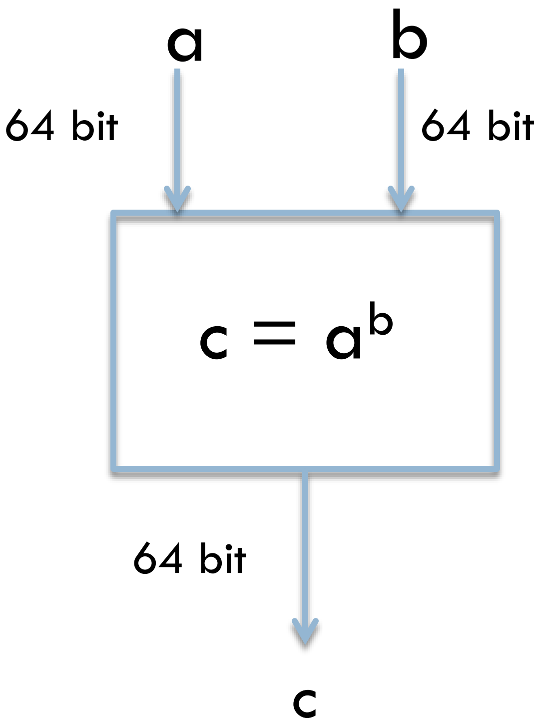
Motivating Example



How do we verify that this circuit works ?

- Formal Verification – Not Scalable!
- Randomly sample some a's and b's
  - Wait! None of the circuits in the past faulted when $10 < b < 40$
  - Finite resources!
- Lets sample from regions where it is likely to fault

Constraints Design



Designing Constraints

- Designers:
  1. $100 < b < 200$
  2. $300 < a < 451$
  3. $40 < a < 50$ and $30 < b < 40$
- Past Experience:
  1. $400 < a < 2000$
  2. $120 < b < 230$
- Users:
  1. $1000<a < 1100$
  2. $20000 < b < a < 22000$

Problem: How can we uniformly sample the values of a and b satisfying the above constraints?

Uniform Generation of SAT-Witnesses

Set of Constraints

SAT Formula

Given a SAT formula, can one uniformly sample solutions without enumerating all solutions

Uniform Generation of SAT-Witnesses

Uniform Generation of SAT-Witnesses

Set of Constraints

↓

SAT Formula

↓

Given a SAT formula, can one uniformly sample solutions without enumerating all solutions while scaling to real world problems?

Scalable Uniform Generation of SAT-Witnesses

Overview

- Prior Work & Our Approach
- Theoretical Results
- Experimental Results
- Where do we go from here?

Prior Work

BDD-based
Guarantees: strong
Performance: weak

SAT-based heuristics
Guarantees: weak
Performance: strong

INDUSTRY

Theoretical Work
Guarantees: strong
Performance: weak

Heuristic Work
Guarantees: weak
Performance: strong

ACADEMIA

BGP Algorithm

XORSample'

Our Contribution

BDD-based
Guarantees: strong
Performance: weak

SAT-based heuristics
Guarantees: weak
Performance: strong

INDUSTRY

UniWit
Guarantees : strong
Performance: strong

Theoretical Work
Guarantees: strong
Performance: weak

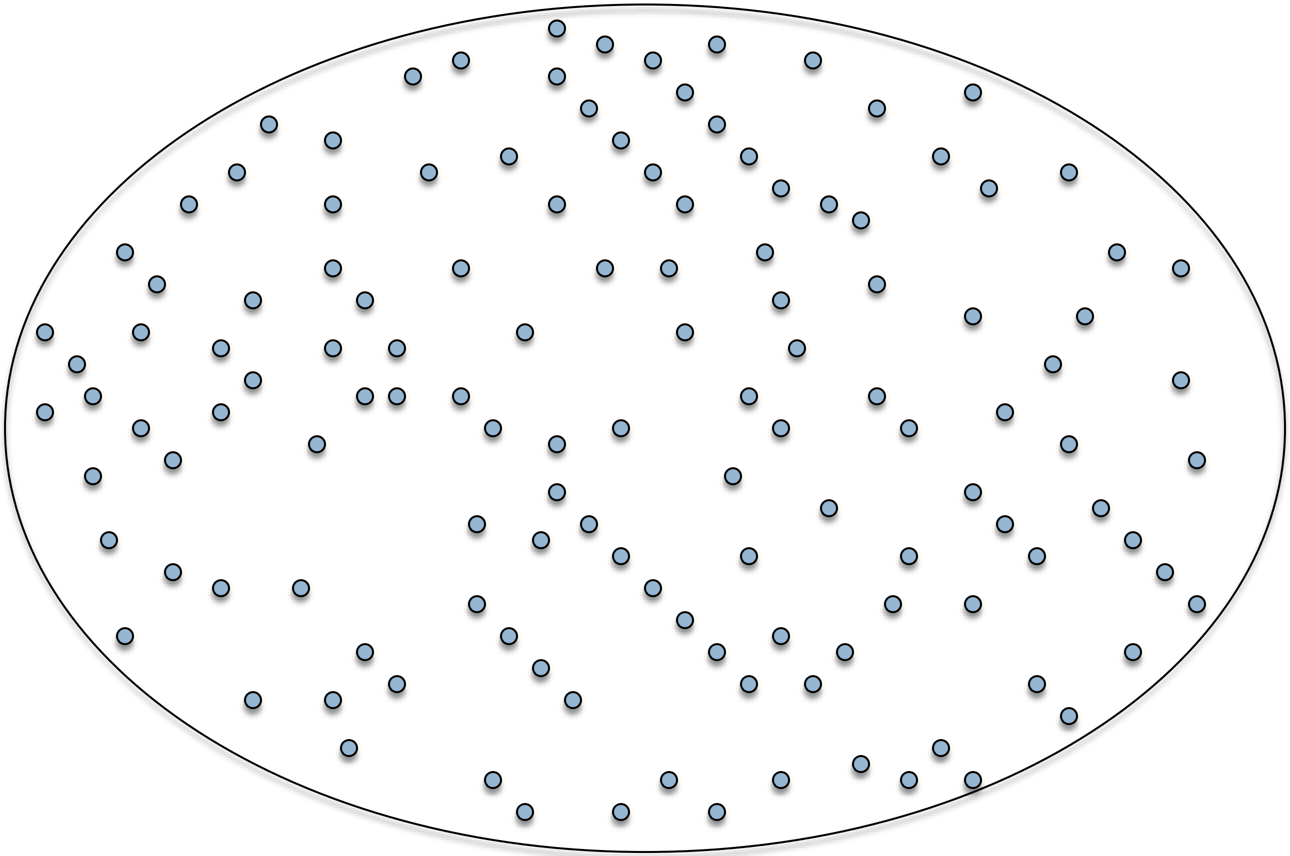
Heuristic Work
Guarantees: weak
Performance: strong

ACADEMIA

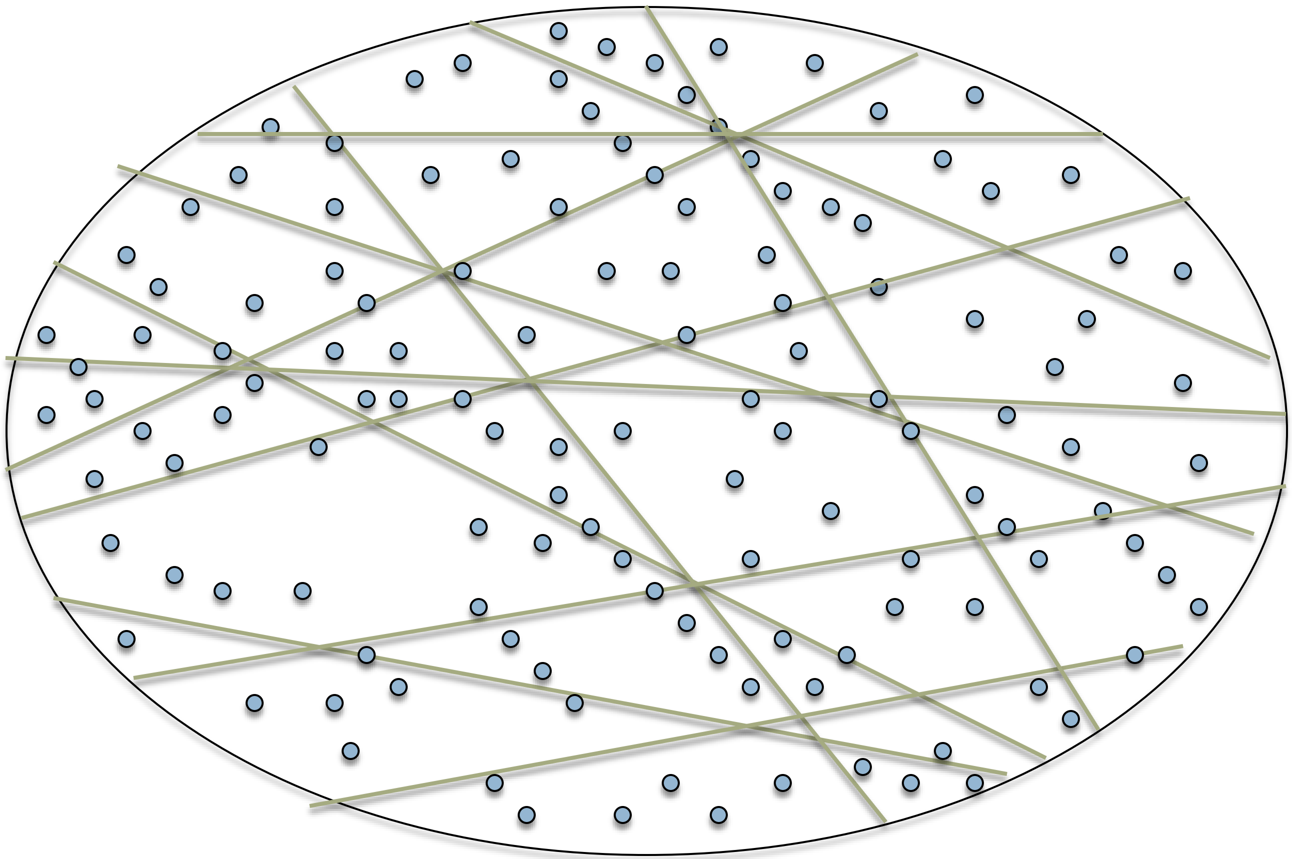
BGP Algorithm

XORSample'

Central Idea



Partitioning into equal "small" cells



How to Partition?

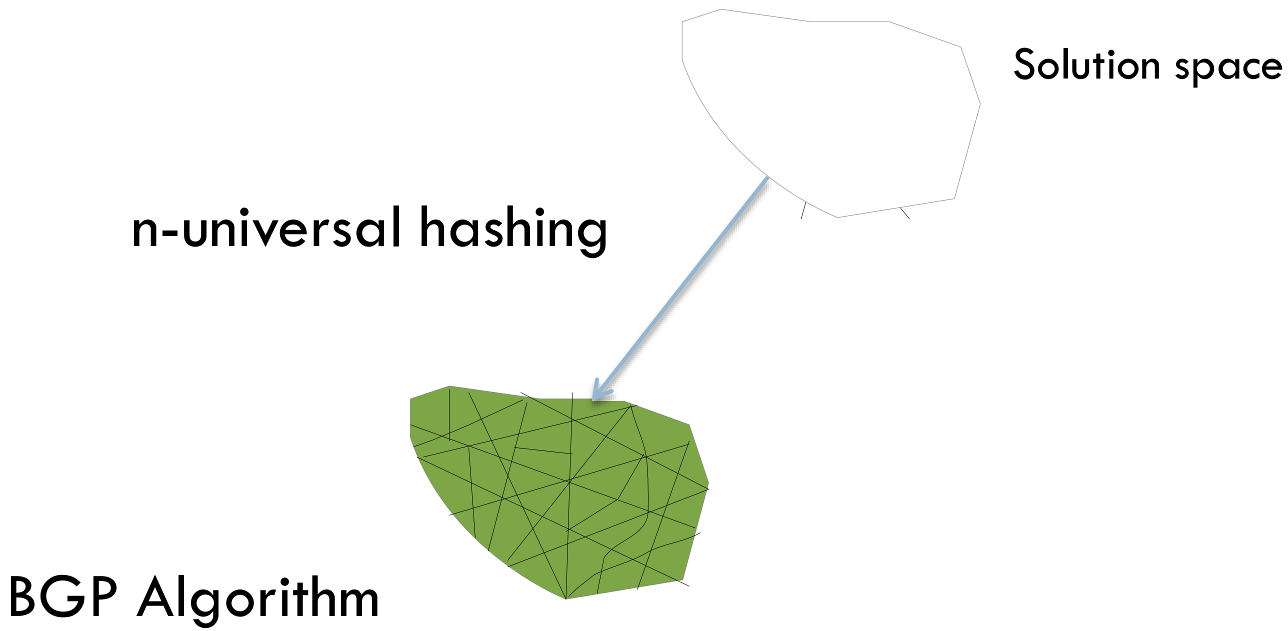
How to partition into roughly equal small cells of solutions without knowing the distribution of solutions?

Universal Hashing
[Carter-Wegman 1979, Sipser 1983]

Lower Universality ➡ 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 range of size of cells
- r-wise universality => Polynomials of degree r-1
- Lower universality => lower complexity

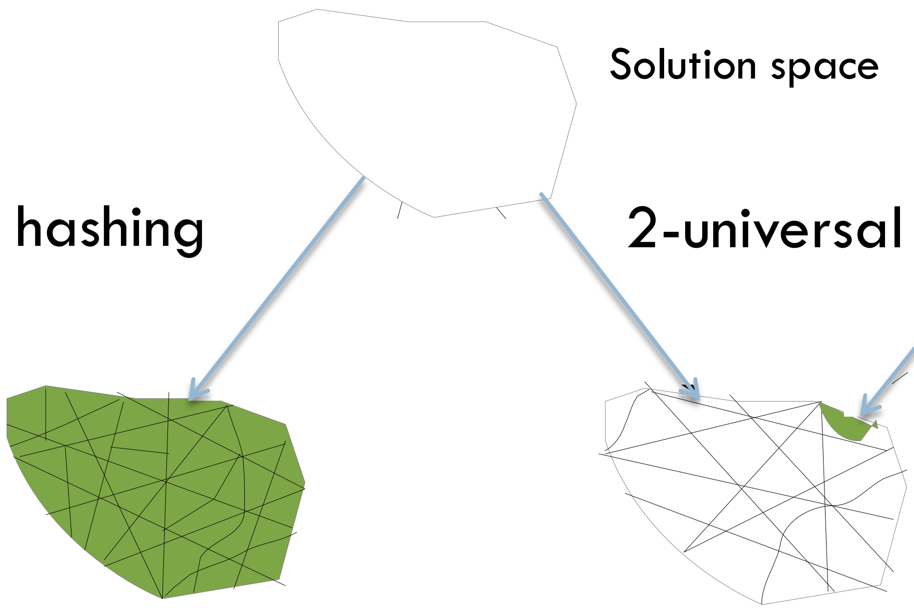
Hashing-Based Approaches



All cells should be small

Uniform Generation

Scaling to Thousands of Variables



BGP Algorithm

UniWit

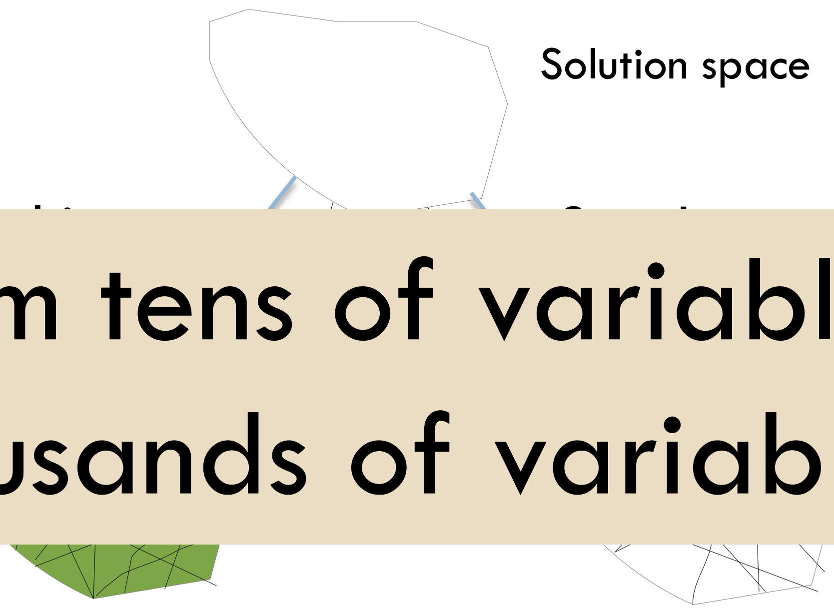
All cells should be small

Only a randomly chosen cells needs to be "small"

Uniform Generation

Near Uniform Generation

Scaling to Thousands of Variables



Solution space

n-univ

From tens of variables to thousands of variables!

BGP Algorithm

UniWit

All cells should be small

Only a randomly chosen cells needs to be “small”

Uniform Generation

Near Uniform Generation

Highlights

- Employs XOR-based hash functions instead of computationally infeasible algebraic hash functions
- Uses off-the-shelf SAT solver CryptoMiniSAT (MiniSAT+XOR support)

Strong Theoretical Guarantees

- Uniformity

For every solution y of R_F

Pr [y is output] = $1/|R_F|$

Strong Theoretical Guarantees

- Near Uniformity

For every solution y of R_F

Pr [y is output] >= $^1/8 \times 1/|R_F|$

- Success Probability

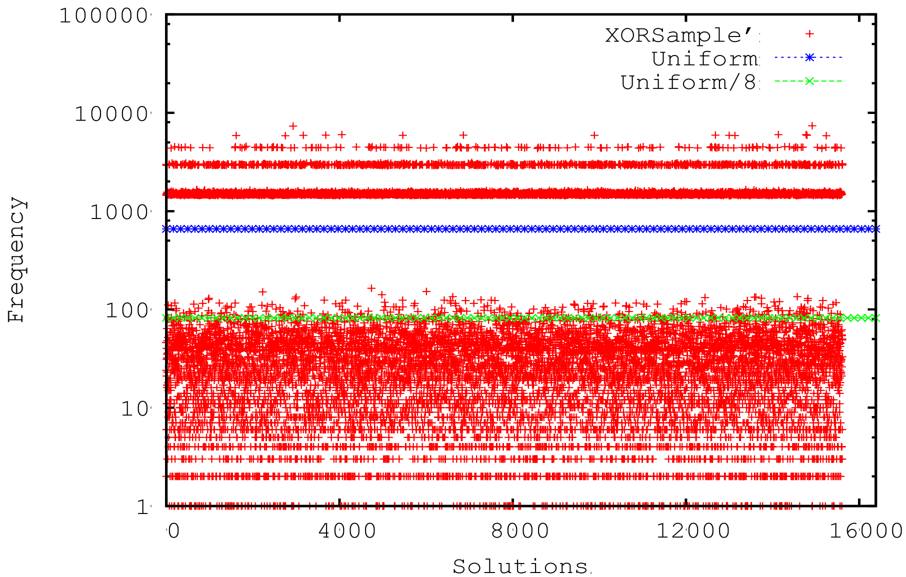
Algorithm UniWit succeeds with probability at least 1/8

- Polynomial: $O(n^{3/2})$ calls to SAT Solver

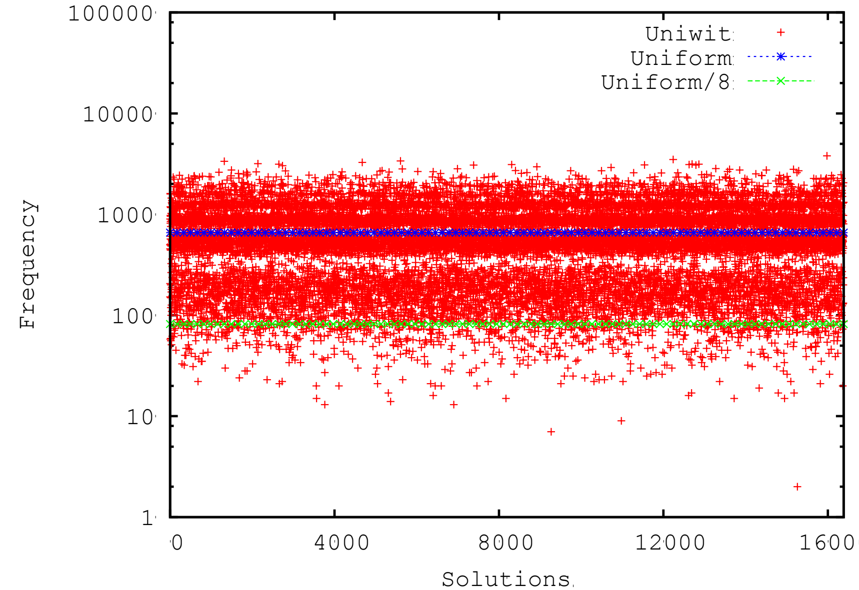
Experimental Methodology

- Benchmarks (over 200)
  - Bit-blasted versions of word level constraints from VHDL designs
  - Bit-blasted versions from SMTLib version and ISCAS'85
- Objectives
  - Comparison with algorithms **BGP** & **XORSample'**
    - **Uniformity**
    - **Performance**

Better Uniformity than State-of-art Generators



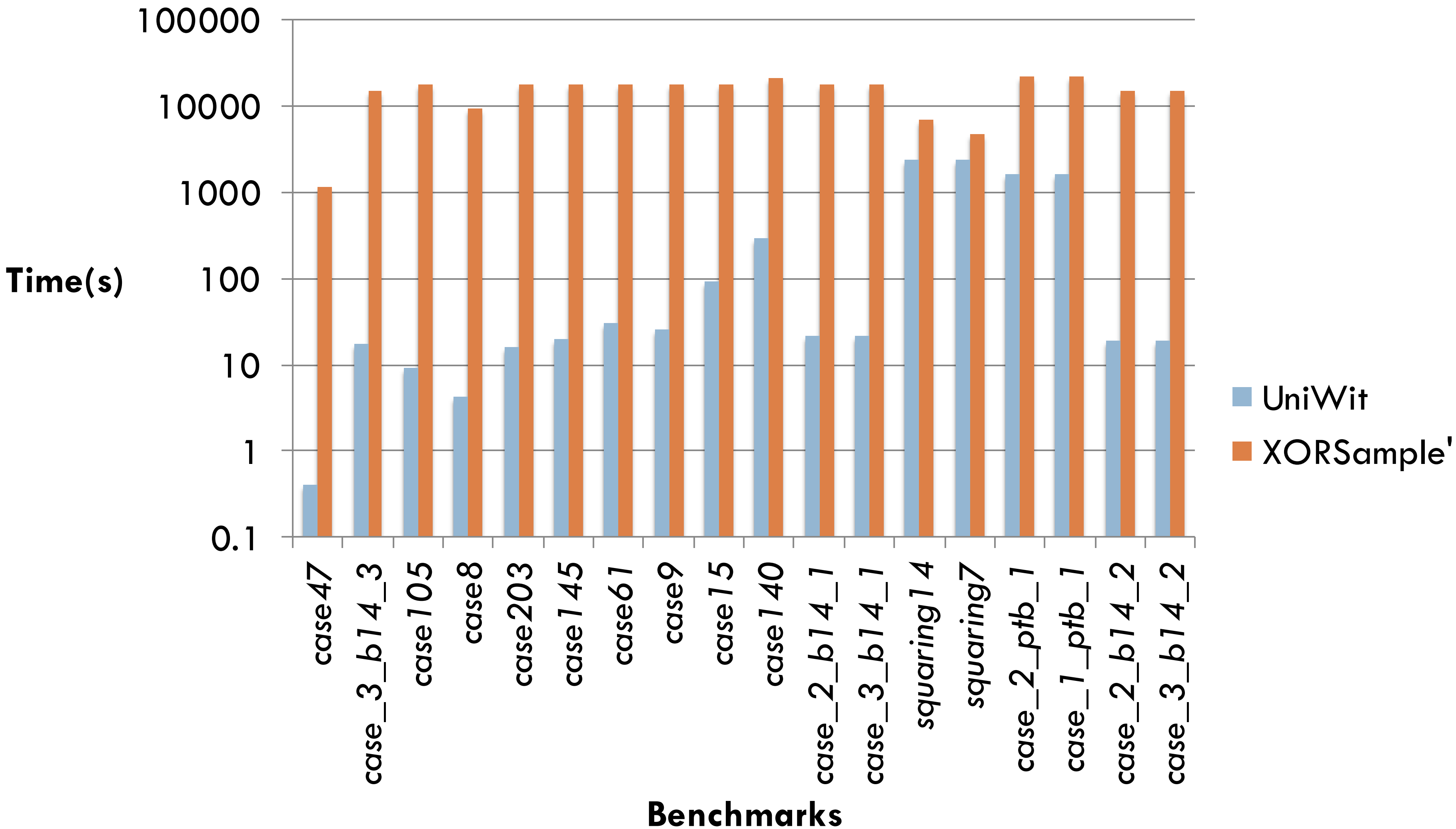
XORSample'



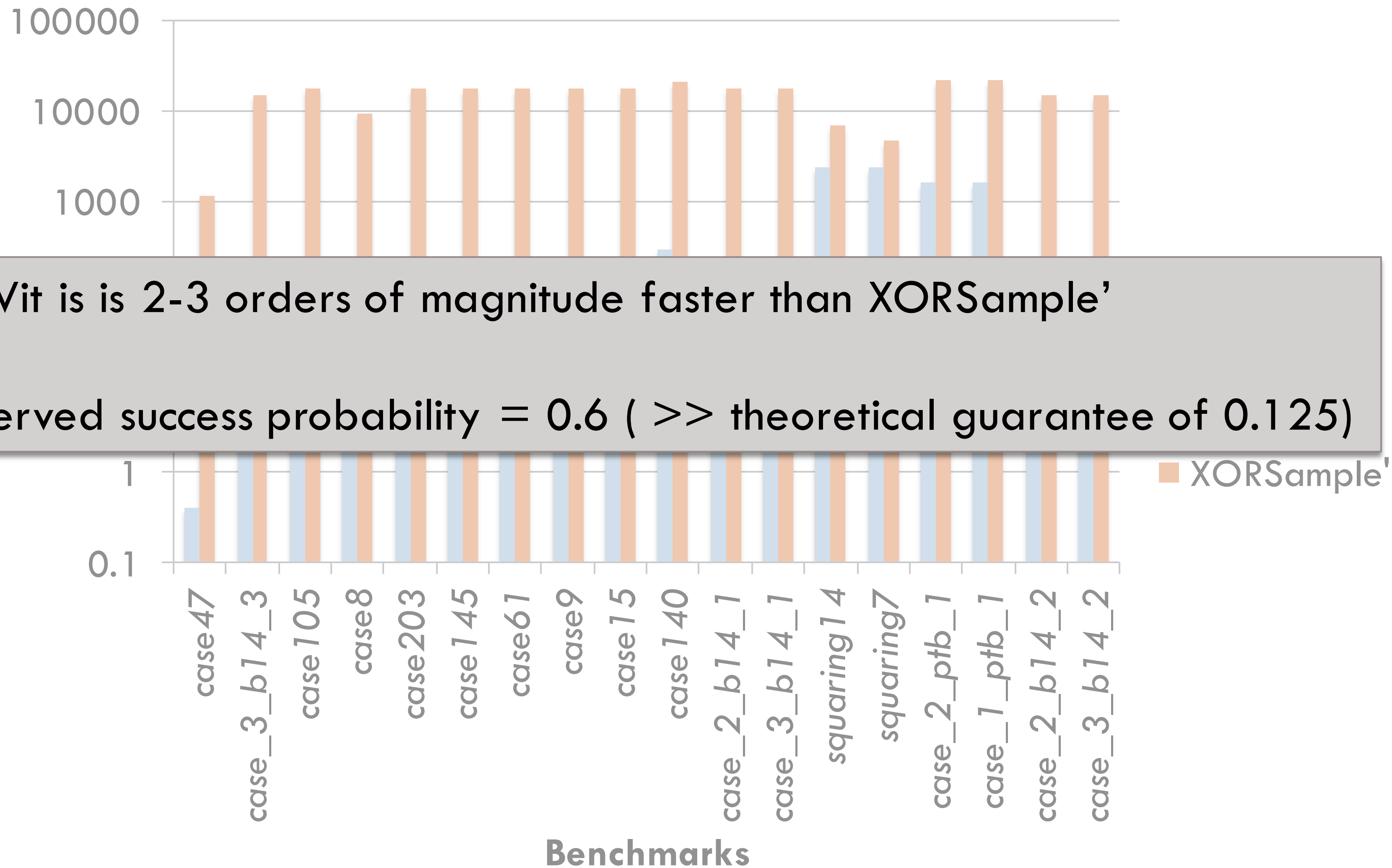
UniWit

- Benchmark: case110.cnf; #var: 287; #clauses: 1263
- Total Runs: **1.08×10^8**; Total Solutions : **16384**
- **XORSample'** could not find 772 solutions and more than 250 solutions were generated only once

2-3 Orders of Magnitude Faster



2-3 Orders of Magnitude Faster



- UniWit is is 2-3 orders of magnitude faster than XORSample'
- Observed success probability = 0.6 (>> theoretical guarantee of 0.125)

Key Takeaways

- Uniform sampling is an important problem
- Prior work didn't scale or offered weak guarantees
- We use 2-wise independent hash function to divide solution space into "small" partitions
- Only a randomly chosen partition has to be small
- Theoretical guarantees of near uniformity
- Major improvements in running time and uniformity compared to the existing generators
- Tool is available at http://www.cfdvs.iitb.ac.in/reports/UniWit/

Where Do We Go From Here?

- Extension to SMT
- Extending the technique to model counting (CP'13)
- Stronger Guarantees
- Efficient hash functions

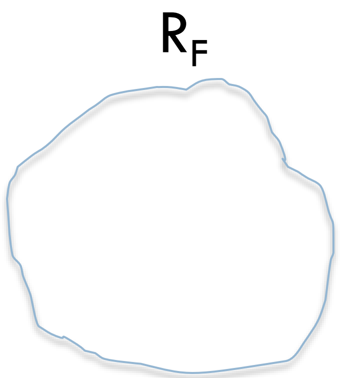
Discussion

Acknowledgments

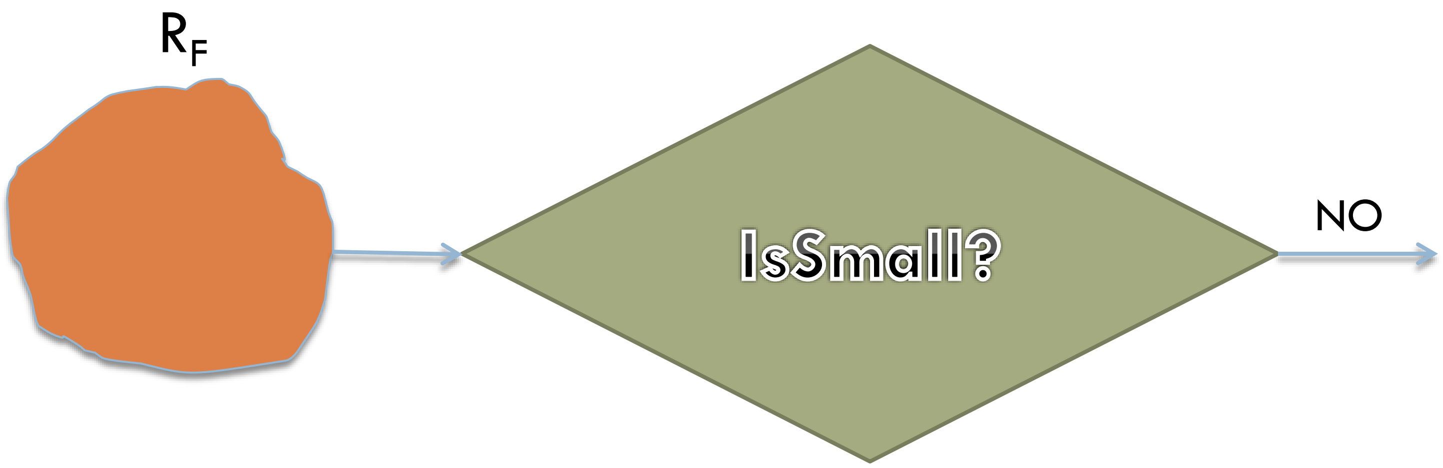
- NSF
- ExCAPE
- Intel
- BRNS, India
- Sun Microsystems
- Sigma Solutions,Inc

Thank You for your attention!

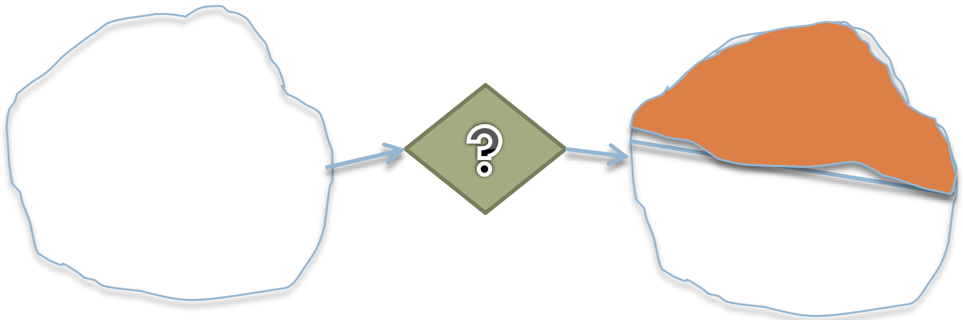
UniWit



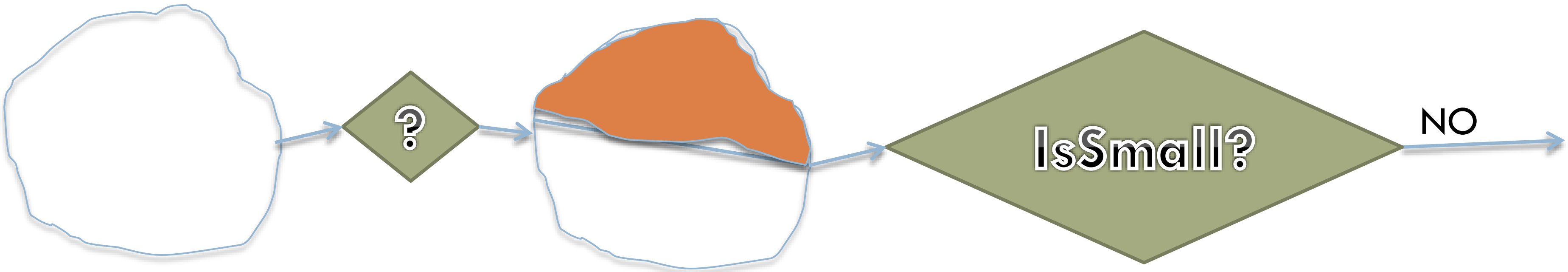
UniWit



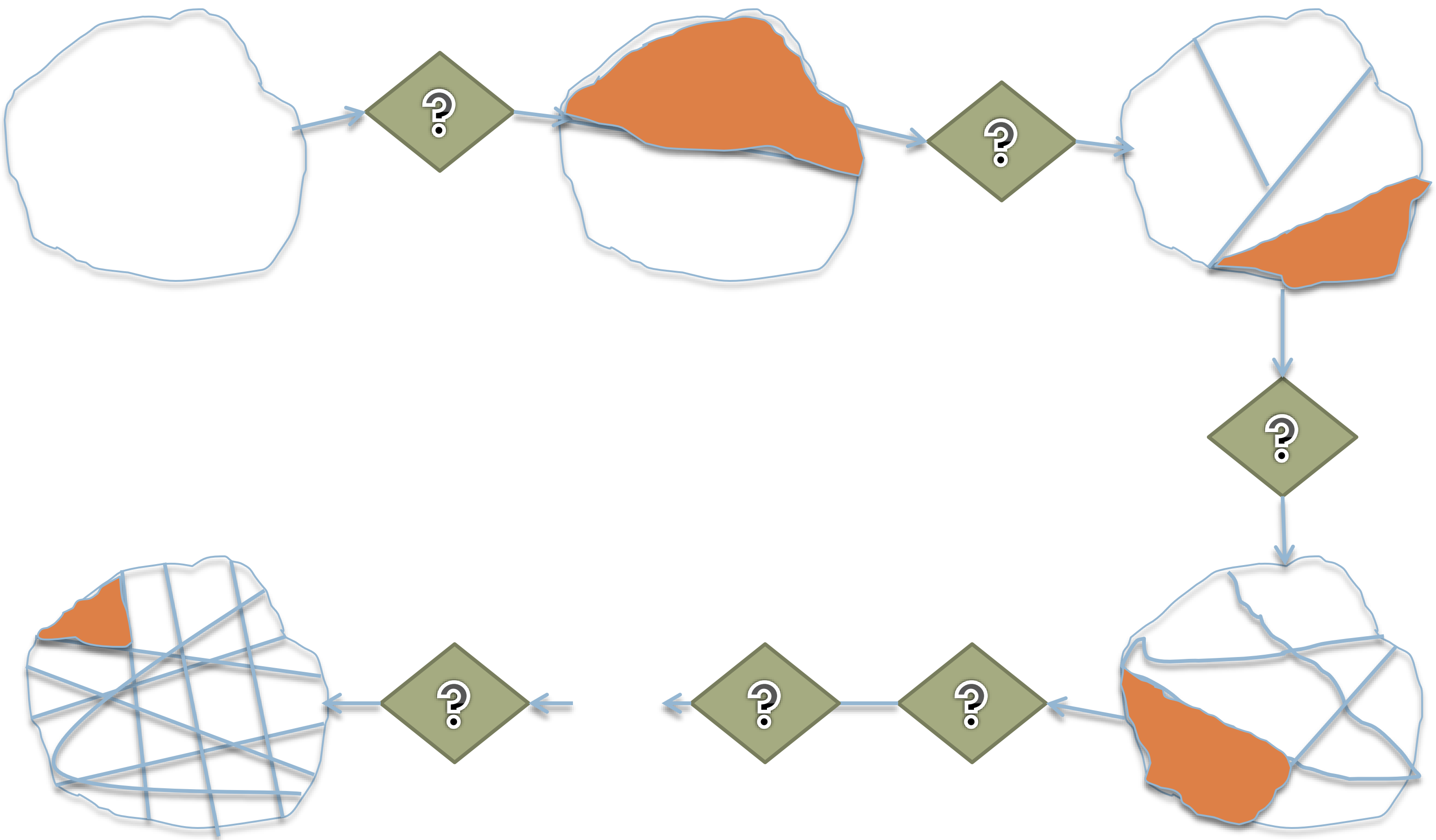
UniWit



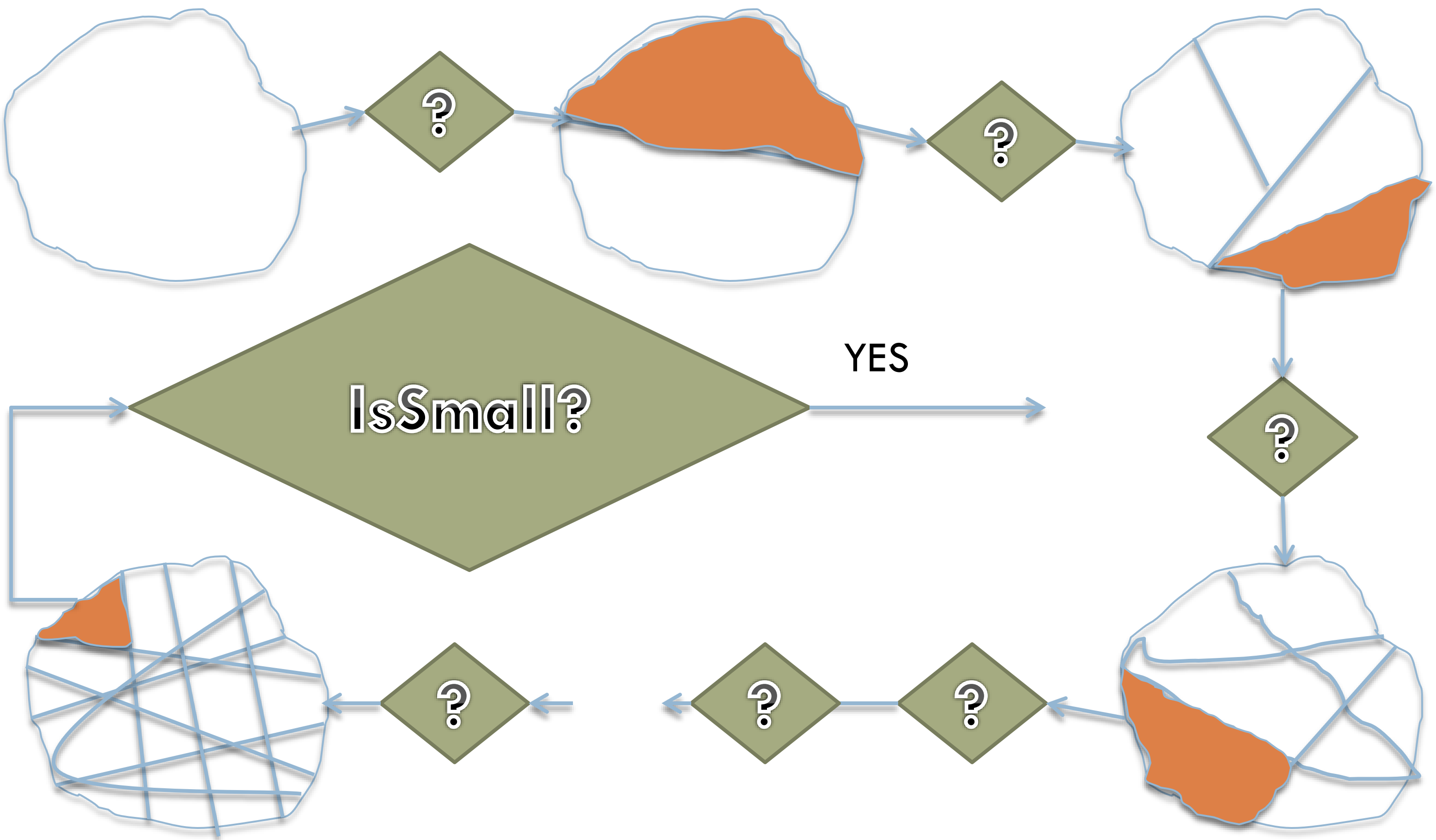
UniWit



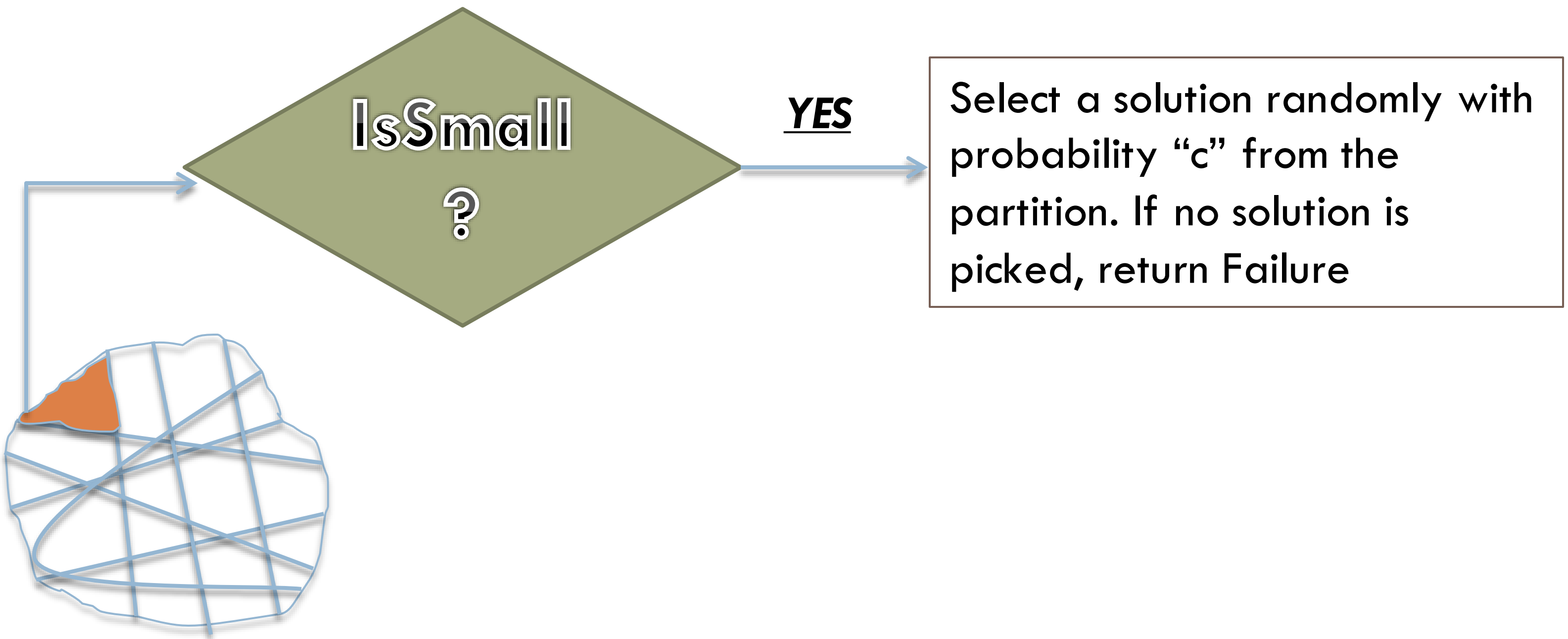
UniWit



UniWit



UniWit


IsSmall
?
YES
Select a solution randomly with probability “c” from the partition. If no solution is picked, return Failure