Outline

- Satisfiability and Randomisation
  - Randomly Generated Clause Sets
  - Sharp Phase Transition
  - Randomised Algoritms for Satisfiability-Checking

Random Clause Generation

How can one generate a random clause?

Random Clause Generation

How can one generate a random clause?
Let's first generate a random literal.

Random Clause Generation

How can one generate a random clause?
Let's first generate a random literal.

- Fix a number n of boolean variables;

Random Clause Generation

How can one generate a random clause?
Let's first generate a random literal.

- Fix a number n of boolean variables;
- Select a literal among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.

Random Clause Generation

How can one generate a random clause?
Let's first generate a random literal.
A random clause is a collection of random literals.

- Fix a number n of boolean variables;
- Select a literal among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.

Random Clause Generation

How can one generate a random clause?
Let's first generate a random literal.
A random clause is a collection of random literals.

- Fix a number n of boolean variables;
- Select a literal among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.
- Fix the length k of a clause;

Random Clause Generation

How can one generate a random clause?
Let's first generate a random literal.
A random clause is a collection of random literals.

- Fix a number n of boolean variables;
- Select a literal among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.
- Fix the length k of a clause;

Suppose we generate random clauses one after one. How does the set of models of this set change?

SAT and k-SAT

SAT is the problem of satisfiability checking for sets of clauses.

k-SAT is the problem of satisfiability checking for sets of clauses of length k.

SAT and k-SAT

SAT is the problem of satisfiability checking for sets of clauses.

k-SAT is the problem of satisfiability checking for sets of clauses of length k.

- SAT is NP-complete;

SAT and k-SAT

SAT is the problem of satisfiability checking for sets of clauses.

k-SAT is the problem of satisfiability checking for sets of clauses of length k.

- SAT is NP-complete;
- 2-SAT is decidable in linear time;

SAT and k-SAT

SAT is the problem of satisfiability checking for sets of clauses.

k-SAT is the problem of satisfiability checking for sets of clauses of length k.

- SAT is NP-complete;
- 2-SAT is decidable in linear time;
- 3-SAT is NP-complete.

SAT and k-SAT

SAT is the problem of satisfiability checking for sets of clauses.

k-SAT is the problem of satisfiability checking for sets of clauses of length k.

- SAT is NP-complete;
- 2-SAT is decidable in linear time;
- 3-SAT is NP-complete.

There is a simple reduction of SAT to 3-SAT based on the same ideas as used for generating short clausal forms (naming). Take a clause having more than 3 literals:

$$L_1 \vee L_2 \vee L_3 \vee L_4 \ldots$$

And replace it by two clauses:

$$L_1 \vee L_2 \vee n$$
$$\neg n \vee L_3 \vee L_4 \ldots$$

where n is a new variable.

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1

Number of models: 32

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1

Number of models: 32

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 24

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 24

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \lor \neg p_3$
$\neg p_2 \lor p_1$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 20

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 20

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$

p_1	p_2	p_3	p_4	p_5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 20

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 12

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 12

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$

p_1	p_2	p_3	p_4	p_5
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 12

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 9

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 9

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 7

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \lor \neg p_3$
$\neg p_2 \lor p_1$
$\neg p_2 \lor p_2$
$p_1 \lor p_1$
$\neg p_5 \lor p_5$
$p_4 \lor p_5$
$\neg p_5 \lor \neg p_3$
$p_2 \lor \neg p_4$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 7

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \lor \neg p_3$
$\neg p_2 \lor p_1$
$\neg p_2 \lor p_2$
$p_1 \lor p_1$
$\neg p_5 \lor p_5$
$p_4 \lor p_5$
$\neg p_5 \lor \neg p_3$
$p_2 \lor \neg p_4$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 4

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$

$\neg p_2 \vee p_1$

$\neg p_2 \vee p_2$

$p_1 \vee p_1$

$\neg p_5 \vee p_5$

$p_4 \vee p_5$

$\neg p_5 \vee \neg p_3$

$p_2 \vee \neg p_4$

$p_5 \vee \neg p_2$

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1

Number of models: 4

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$
$p_2 \vee \neg p_4$
$p_5 \vee \neg p_2$

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	1	0	0	1
1	1	0	1	1

Number of models: 3

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$
$p_2 \vee \neg p_4$
$p_5 \vee \neg p_2$
$p_5 \vee p_2$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1
1	1	0	0	1
1	1	0	1	1

Number of models: 3

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$
$p_2 \vee \neg p_4$
$p_5 \vee \neg p_2$
$p_5 \vee p_2$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1

Number of models: 1

Example (Obtained by a Program) for $n = 5$ and $k = 2$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$
$p_2 \vee \neg p_4$
$p_5 \vee \neg p_2$
$p_5 \vee p_2$
$\neg p_1 \vee \neg p_4$

Number of models: 1

Example (Obtained by a Program) for $n = 5$ and $k = 2$

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$
$p_2 \vee \neg p_4$
$p_5 \vee \neg p_2$
$p_5 \vee p_2$
$\neg p_1 \vee \neg p_4$
$p_5 \vee p_2$

p_1	p_2	p_3	p_4	p_5

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1

Number of models: 1

Example (Obtained by a Program) for $n = 5$ and $k = 2$

	p_1	p_2	p_3	p_4	p_5
$\neg p_2 \vee \neg p_3$					
$\neg p_2 \vee p_1$					
$\neg p_2 \vee p_2$					
$p_1 \vee p_1$					
$\neg p_5 \vee p_5$					
$p_4 \vee p_5$					
$\neg p_5 \vee \neg p_3$					
$p_2 \vee \neg p_4$					
$p_5 \vee \neg p_2$					
$p_5 \vee p_2$					
$\neg p_1 \vee \neg p_4$					
$p_5 \vee p_2$					
$\neg p_1 \vee \neg p_5$					

p_1	p_2	p_3	p_4	p_5
1	0	0	0	1

Number of models: 1

Example (Obtained by a Program) for $n = 5$ and $k = 2$

	p_1	p_2	p_3	p_4	p_5	p_1	p_2	p_3	p_4	p_5

$\neg p_2 \vee \neg p_3$
$\neg p_2 \vee p_1$
$\neg p_2 \vee p_2$
$p_1 \vee p_1$
$\neg p_5 \vee p_5$
$p_4 \vee p_5$
$\neg p_5 \vee \neg p_3$
$p_2 \vee \neg p_4$
$p_5 \vee \neg p_2$
$p_5 \vee p_2$
$\neg p_1 \vee \neg p_4$
$p_5 \vee p_2$
$\neg p_1 \vee \neg p_5$

Number of models: 0

This set of 13 clauses is unsatisfiable.

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Fix:

- Number n of boolean variables;

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Fix:

- Number n of boolean variables;
- Number k of literals per clause, so we will generate k-SAT instances;

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Fix:

- Number n of boolean variables;
- Number k of literals per clause, so we will generate k-SAT instances;
- Number m of clauses.

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Fix:

- Number n of boolean variables;
- Number k of literals per clause, so we will generate k-SAT instances;
- Number m of clauses.

Generate m clauses, each one has k literals randomly generated among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Fix:

- Number n of boolean variables;
- Number k of literals per clause, so we will generate k-SAT instances;
- Number m of clauses.

Generate m clauses, each one has k literals randomly generated among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.

Note that the probability is a monotone function: the more clauses we generate, the higher chance we have that the set is unsatisfiable.

Random Clause Generation

We are interested in the probability that a set of clauses of a given size is unsatisfiable.

Fix:

- Number n of boolean variables;
- Number k of literals per clause, so we will generate k-SAT instances;
- Number m of clauses. Real number r: ratio of clauses per variable.

Generate $\lceil rn \rceil$ clauses, each one has k literals randomly generated among $p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n$ with an equal probability.

Note that the probability is a monotone function: the more clauses we generate, the higher chance we have that the set is unsatisfiable.

Roulette

SAT Roulette

We will generate random instances of 2-SAT with 5-variables.
You will bet on whether the resuting set of clauses is satisfiable or not.

SAT Roulette

We will generate random instances of 2-SAT with 5-variables.
You will bet on whether the resuting set of clauses is satisfiable or not.

- What would you bet on if we generate 5 clauses?

SAT Roulette

We will generate random instances of 2-SAT with 5-variables.
You will bet on whether the resuting set of clauses is satisfiable or not.

- What would you bet on if we generate 5 clauses?
- What would you bet on if we generate 100 clauses?

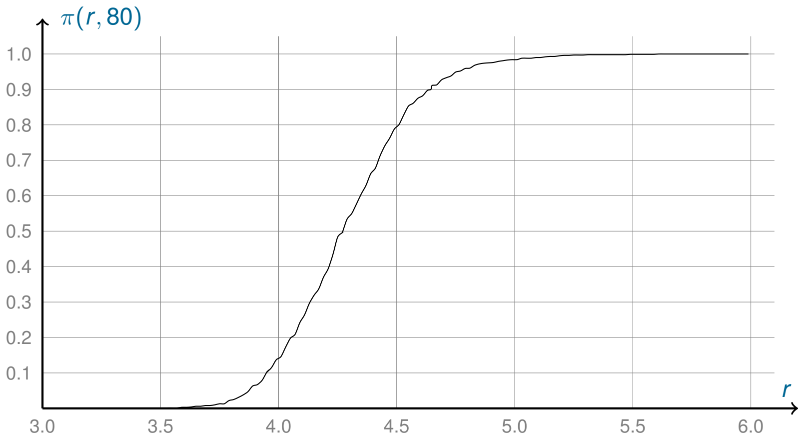
SAT Roulette

We will generate random instances of 2-SAT with 5-variables.
You will bet on whether the resuting set of clauses is satisfiable or not.

- What would you bet on if we generate 5 clauses?
- What would you bet on if we generate 100 clauses?
- What would you bet on if we generate 15 clauses?

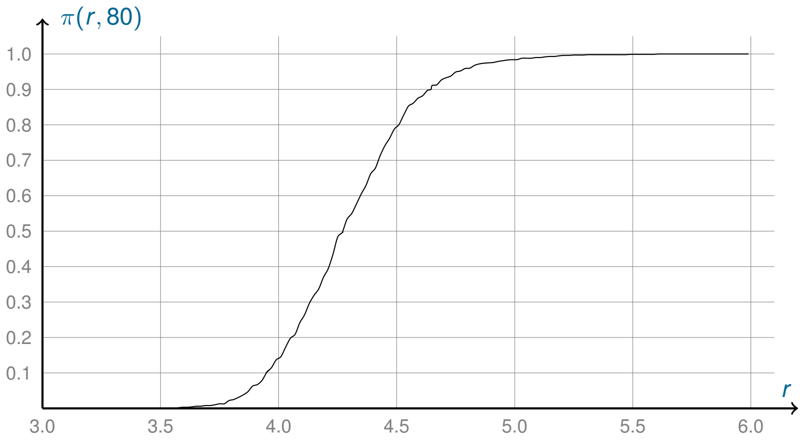
Probability of Obtaining an Unsatisfiable Set

This probablity is a monotone function: the more clauses we generate, the higher chance to obtain an unsatisfiable set.



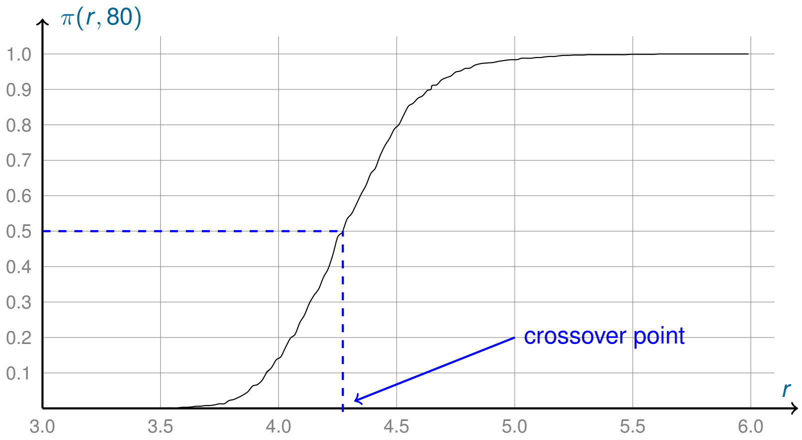
Probability of Obtaining an Unsatisfiable Set

This probablity is a monotone function: the more clauses we generate, the higher chance to obtain an unsatisfiable set.
Crossover point: the value of r at which the probability crosses 0.5.



Probability of Obtaining an Unsatisfiable Set

This probablity is a monotone function: the more clauses we generate, the higher chance to obtain an unsatisfiable set.
Crossover point: the value of r at which the probability crosses 0.5.



ϵ-Window

Take a (small) number $\epsilon > 0$. ϵ-window is the interval of values of r where the probability is between ϵ and $1 - \epsilon$.

ϵ-Window

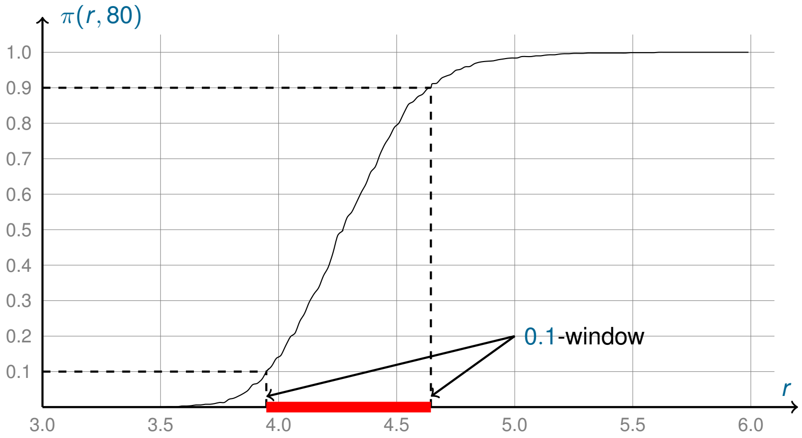
Take a (small) number $\epsilon > 0$. ϵ-window is the interval of values of r where the probability is between ϵ and $1 - \epsilon$.

For example, take $\epsilon = 0.1$.

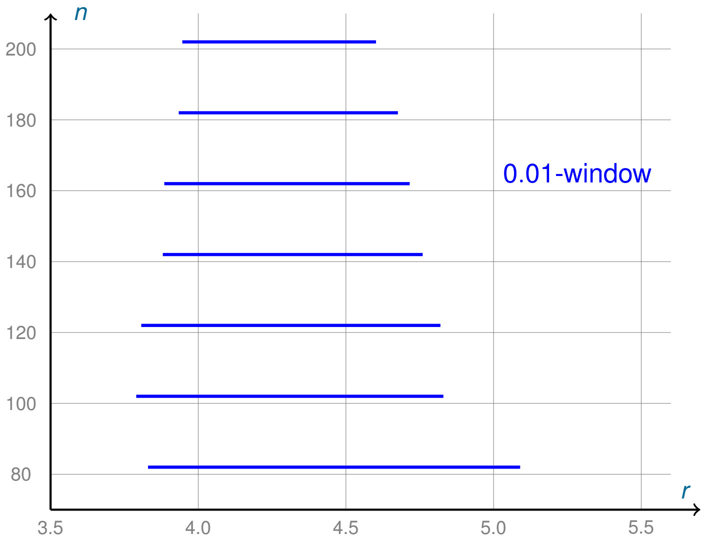
ϵ-Window

Take a (small) number $\epsilon > 0$. ϵ-window is the interval of values of r where the probability is between ϵ and $1 - \epsilon$.

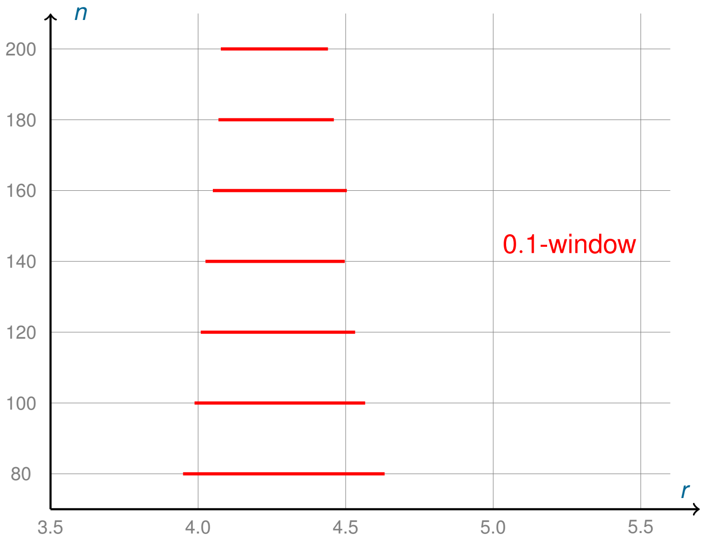
For example, take $\epsilon = 0.1$.



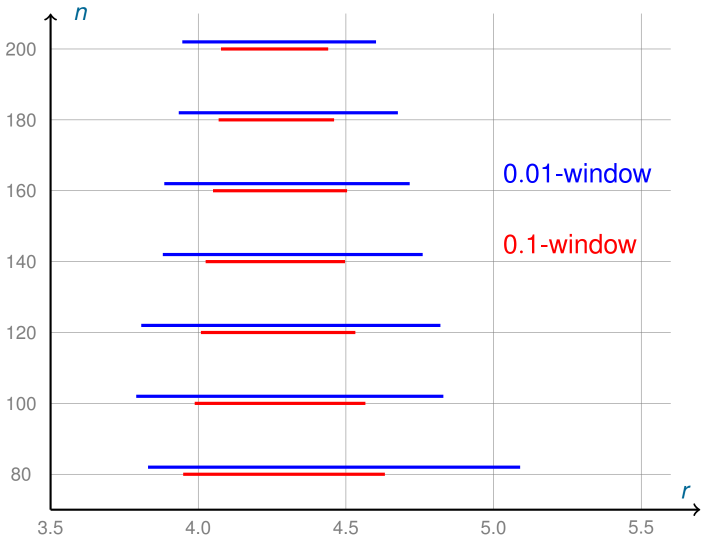
Scaling Window Effect



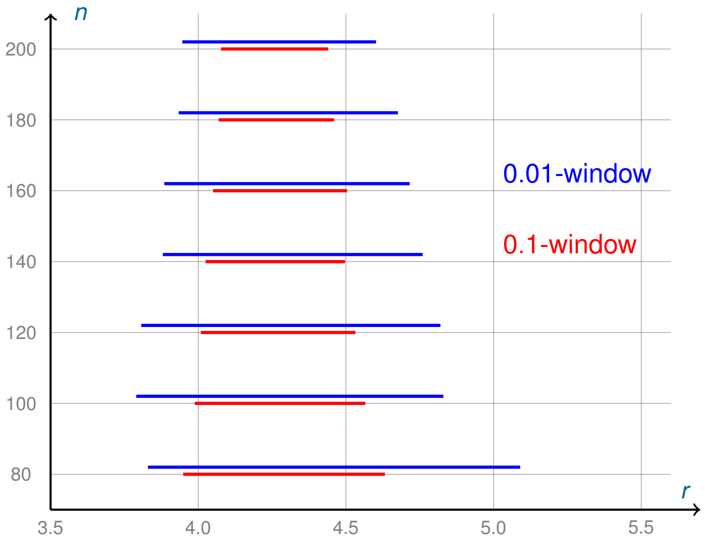
Scaling Window Effect



Scaling Window Effect

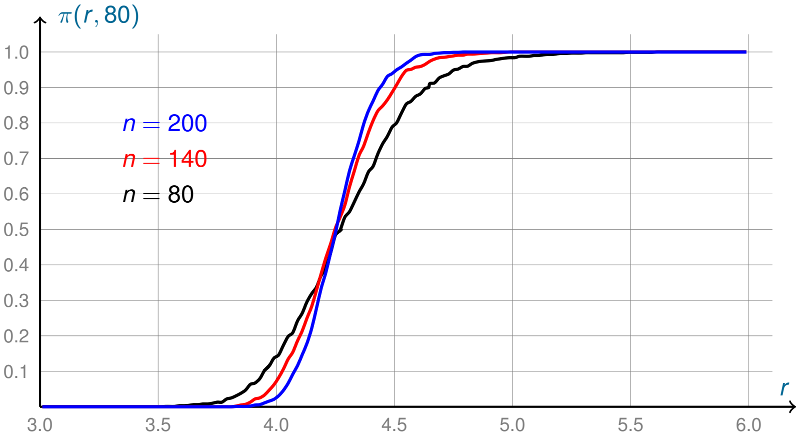


Scaling Window Effect

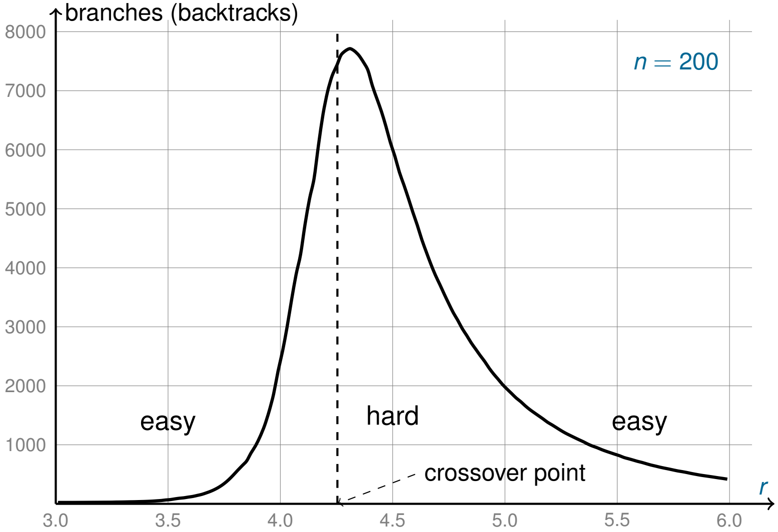


Conjecture: for $n \to \infty$ every ϵ-window "degenerates into a point".

Sharp Phase Transition



Easy-Hard-Easy Pattern



End of Lecture 8

Slides for lecture 8 end here . . .

Satisfiability-Checking Algorithm that Cannot Establish Unsatisfiability

Satisfiability-Checking Algorithm that Cannot Establish Unsatisfiability

procedure $CHAOS(S)$
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*

Satisfiability-Checking Algorithm that Cannot Establish Unsatisfiability

procedure *CHAOS*(S)
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: positive integer *MAX-TRIES*
begin
  repeat *MAX-TRIES* times

end

Satisfiability-Checking Algorithm that Cannot Establish Unsatisfiability

```
procedure CHAOS(S)
input: set of clauses S
output: interpretation I such that I ⊨ S or don't know
parameters: positive integer MAX-TRIES
begin
  repeat MAX-TRIES times
    I := random interpretation
    if I ⊨ S then return I
  return don't know
end
```

SAT as a Decision Problem

Decision problem: any problem on any infinite domain, that has a yes-no answer. Each element of this domain is called an instance of this problem.

SAT as a Decision Problem

Decision problem: any problem on any infinite domain, that has a yes-no answer. Each element of this domain is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance in a system of linear inequalities;
- an answer is yes if it has a solution.

SAT as a Decision Problem

Decision problem: any problem on any infinite domain, that has a yes-no answer. Each element of this domain is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance in a system of linear inequalities;
- an answer is yes if it has a solution.

SAT is a decision problem:

- an instance is a finite set of clauses.
- it has a yes-no answer: yes (satisfiable) or no (unsatisfiable)

SAT as a Decision Problem

Decision problem: any problem on any infinite domain, that has a yes-no answer. Each element of this domain is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance in a system of linear inequalities;
- an answer is yes if it has a solution.

SAT is a decision problem:

- an instance is a finite set of clauses.
- it has a yes-no answer: yes (satisfiable) or no (unsatisfiable)

Witness for a instance I: any data D such that, given D, one can check in polynomial time (in D) that I has a yes-answer.

SAT as a Decision Problem

Decision problem: any problem on any infinite domain, that has a yes-no answer. Each element of this domain is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance in a system of linear inequalities;
- an answer is yes if it has a solution.

SAT is a decision problem:

- an instance is a finite set of clauses.
- it has a yes-no answer: yes (satisfiable) or no (unsatisfiable)

Witness for a instance I: any data D such that, given D, one can check in polynomial time (in D) that I has a yes-answer.

Satisfiability has short witnesses: interpretations.

SAT as a Decision Problem

Decision problem: any problem on any infinite domain, that has a yes-no answer. Each element of this domain is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance in a system of linear inequalities;
- an answer is yes if it has a solution.

SAT is a decision problem:

- an instance is a finite set of clauses.
- it has a yes-no answer: yes (satisfiable) or no (unsatisfiable)

Witness for a instance I: any data D such that, given D, one can check in polynomial time (in D) that I has a yes-answer.

Satisfiability has short witnesses: interpretations.

Unsatisfiability has no polynomial-size witnesses, unless $NP = coNP$.

Randomised Algorithms for SAT

- Choose a random interpretation.

Randomised Algorithms for SAT

- Choose a random interpretation.
- If this interpretation is not a model, repeatedly choose a variable and change its value in the interpretation (flip the variable).

Randomised Algorithms for SAT

- Choose a random interpretation.
- If this interpretation is not a model, repeatedly choose a variable and change its value in the interpretation (flip the variable).

The flipped variables are chosen using heuristics or randomly, or both.

Randomised Algorithms for SAT

- Choose a random interpretation.
- If this interpretation is not a model, repeatedly choose a variable and change its value in the interpretation (flip the variable).

The flipped variables are chosen using heuristics or randomly, or both.

$$flip(I, p)(q) = \begin{cases} I(q), & \text{if } p \neq q; \\ 1, & \text{if } p = q \text{ and } I(p) = 0; \\ 0, & \text{if } p = q \text{ and } I(p) = 1. \end{cases}$$

Randomised Algorithms for SAT

- Choose a random interpretation.
- If this interpretation is not a model, repeatedly choose a variable and change its value in the interpretation (flip the variable).

The flipped variables are chosen using heuristics or randomly, or both.

$$flip(I,p)(q) = \begin{cases} I(q), & \text{if } p \neq q; \\ 1, & \text{if } p = q \text{ and } I(p) = 0; \\ 0, & \text{if } p = q \text{ and } I(p) = 1. \end{cases}$$

In other words, the interpretation $flip(I,p)$ is obtained from I by changing its value on p.

GSAT

procedure $GSAT(S)$
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*

GSAT

procedure $GSAT(S)$
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*

GSAT

procedure $GSAT(S)$
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*
begin
  repeat *MAX-TRIES* times
    I := random interpretation
    if $I \models S$ **then return** I

end

GSAT

```
procedure GSAT(S)
input: set of clauses S
output: interpretation I such that I ⊨ S or don't know
parameters: integers MAX-TRIES, MAX-FLIPS
begin
 repeat MAX-TRIES times
  I  :=  random interpretation
  if I ⊨ S then return I
  repeat MAX-FLIPS times
   p  :=  a variable such that flip(I, p) satisfies
          the maximal number of clauses in S
   I = flip(I, p)
   if I ⊨ S then return I
 return don't know
end
```

GSAT Example

0		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

GSAT Example

0		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4					

GSAT Example

0		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip	interpretation			satisfied clauses				candidates	flipped
no.	p_1	p_2	p_3		p_1	p_2	p_3	for flipping	variable
1	0	0	1	4	3	4	4		

GSAT Example

0		1		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1						

GSAT Example

0		1		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation p_1	p_2	p_3	satisfied clauses	p_1	p_2	p_3	candidates for flipping	flipped variable
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4					

GSAT Example

0		1		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4	3	4	4		

GSAT Example

0		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4	3	4	4	p_2, p_3	p_3
3	0	1	0						

GSAT Example

0		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4	3	4	4	p_2, p_3	p_3
3	0	1	0	4					

GSAT Example

0		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip	interpretation			satisfied clauses				candidates	flipped
no.	p_1	p_2	p_3		p_1	p_2	p_3	for flipping	variable
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4	3	4	4	p_2, p_3	p_3
3	0	1	0	4	5	4	4		

GSAT Example

1		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4	3	4	4	p_2, p_3	p_3
3	0	1	0	4	5	4	4	p_1	p_1
	1	1	0						

GSAT Example

1		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			satisfied clauses				candidates for flipping	flipped variable
	p_1	p_2	p_3		p_1	p_2	p_3		
1	0	0	1	4	3	4	4	p_2, p_3	p_2
2	0	1	1	4	3	4	4	p_2, p_3	p_3
3	0	1	0	4	5	4	4	p_1	p_1
	1	1	0	5					

GSAT with Random Walks

procedure *GSATwithWalks*(S)
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*

GSAT with Random Walks

procedure *GSATwithWalks*(S)
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*
real number $0 \leq \pi \leq 1$ (probability of a sideways move),

GSAT with Random Walks

procedure *GSATwithWalks*(S)
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*
real number $0 \leq \pi \leq 1$ (probability of a sideways move),
begin
repeat *MAX-TRIES* times
I := random interpretation ;
if $I \models S$ **then** **return** I

end

GSAT with Random Walks

procedure *GSATwithWalks*(S)
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*
real number $0 \leq \pi \leq 1$ (probability of a sideways move),
begin
repeat *MAX-TRIES* times
I := random interpretation ;
if $I \models S$ **then** **return** I
repeat *MAX-FLIPS* times
with probability π
p := a variable such that $flip(I, p)$ satisfies the maximal number of clauses in S
with probability $1 - \pi$
randomly select p among all variables occurring in clauses false in I
$I = flip(I, p)$;
if $I \models S$ **then** **return** I
return *don't know*
end

WSAT

procedure $WSAT(S)$
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*

WSAT

procedure *WSAT(S)*
input: set of clauses S
output: interpretation I such that $I \models S$ or *don't know*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*
begin
  repeat *MAX-TRIES* times
    I := random interpretation
    if $I \models S$ **then** **return** I

end

WSAT

```
procedure WSAT(S)
input: set of clauses S
output: interpretation I such that I ⊨ S or don't know
parameters: integers MAX-TRIES, MAX-FLIPS
begin
 repeat MAX-TRIES times
  I := random interpretation
  if I ⊨ S then return I
  repeat MAX-FLIPS times
   randomly select a clause C ∈ S such that I ⊭ C
   randomly select a variable p in C
   I = flip(I, p)
   if I ⊨ S then return I
 return don't know
end
```

WSAT Example

0		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

WSAT Example

0		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1			

WSAT Example

0		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	

WSAT Example

1		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	p_1
2	1	0	1			

WSAT Example

1		0		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	p_1
2	1	0	1	$\neg p_1 \vee \neg p_3$ $\neg p_1 \vee p_2$	p_1, p_2, p_3	

WSAT Example

1		1		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	p_1
2	1	0	1	$\neg p_1 \vee \neg p_3$ $\neg p_1 \vee p_2$	p_1, p_2, p_3	p_2
3	1	1	1			

WSAT Example

1		1		1
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	p_1
2	1	0	1	$\neg p_1 \vee \neg p_3$ $\neg p_1 \vee p_2$	p_1, p_2, p_3	p_2
3	1	1	1	$\neg p_2 \vee \neg p_3$ $\neg p_1 \vee \neg p_3$	p_1, p_2, p_3	

WSAT Example

1		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation p_1	p_2	p_3	unsatisfied clauses	candidates for flipping	flipped variable
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	p_1
2	1	0	1	$\neg p_1 \vee \neg p_3$ $\neg p_1 \vee p_2$	p_1, p_2, p_3	p_2
3	1	1	1	$\neg p_2 \vee \neg p_3$ $\neg p_1 \vee \neg p_3$	p_1, p_2, p_3	p_3
	1	1	0			

WSAT Example

1		1		0
p_1	$\vee$	$\neg p_2$	$\vee$	p_3
		$\neg p_2$	$\vee$	$\neg p_3$
$\neg p_1$			$\vee$	$\neg p_3$
$\neg p_1$	$\vee$	p_2		
p_1	$\vee$	p_2		

flip no.	interpretation			unsatisfied clauses	candidates for flipping	flipped variable
	p_1	p_2	p_3			
1	0	0	1	$p_1 \vee p_2$	p_1, p_2	p_1
2	1	0	1	$\neg p_1 \vee \neg p_3$ $\neg p_1 \vee p_2$	p_1, p_2, p_3	p_2
3	1	1	1	$\neg p_2 \vee \neg p_3$ $\neg p_1 \vee \neg p_3$	p_1, p_2, p_3	p_3
	1	1	0			

End of Lecture 9

Slides for lecture 9 end here . . .