Next

Next: Randomized satisfiability algorithms.

SAT as a Decision Problem

Decision problem: any collection of problems that have a yes-no answer. Each element of this collection is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance is 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 an 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 small witnesses: interpretations.

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

SAT as a Decision Problem

Decision problem: any collection of problems that have a yes-no answer. Each element of this collection is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance is 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 an 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 small witnesses: interpretations.

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

SAT as a Decision Problem

Decision problem: any collection of problems that have a yes-no answer. Each element of this collection is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance is 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 an 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 small witnesses: interpretations.

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

SAT as a Decision Problem

Decision problem: any collection of problems that have a yes-no answer. Each element of this collection is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance is 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 an 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 small witnesses: interpretations.

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

SAT as a Decision Problem

Decision problem: any collection of problems that have a yes-no answer. Each element of this collection is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance is 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 an 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 small witnesses: interpretations.

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

SAT as a Decision Problem

Decision problem: any collection of problems that have a yes-no answer. Each element of this collection is called an instance of this problem.

Example: solvability of systems of linear inequalities over integers.

- an instance is 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 an 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 small witnesses: interpretations.

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

Satisfiability 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
    I := random interpretation
    if $I \models S$ **then return** I
  return *don't know*
end

Randomized satisfiability algorithms:

- random search for a satisfying assignment;
- cannot establish unsatisfiability;
- may return "don't know"

Satisfiability 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
    I := random interpretation
    if $I \models S$ **then return** I
  return *don't know*
end

Randomized satisfiability algorithms:

- random search for a satisfying assignment;
- cannot establish unsatisfiability;
- may return "don't know"

Satisfiability 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
    I := random interpretation
    if $I \models S$ **then return** I
  return *don't know*
end

Randomized satisfiability algorithms:

- random search for a satisfying assignment;
- cannot establish unsatisfiability;
- may return "don't know"

Satisfiability 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
    I := random interpretation
    if $I \models S$ **then return** I
  return *don't know*
end

Randomized satisfiability algorithms:

- random search for a satisfying assignment;
- cannot establish unsatisfiability;
- may return "don't know"

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.

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

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

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.

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.

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.

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

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

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.

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

In other words, the interpretation $\mathit{flip}(I,p)$ 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*
parameters: integers *MAX-TRIES*, *MAX-FLIPS*
begin
  repeat *MAX-TRIES* times
    I := random interpretation
    if $I \models S$ **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 $I \models S$ **then return** I
  return *don't know*
end

GSAT is a local search algorithm, it tries to maximise the number of satisfied clauses by local changes.

Note: there can be several candidates for flipping; we pick p uniformly at random among such candidates.

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 ⊨ 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 is a local search algorithm, it tries to maximise the number of satisfied clauses by local changes.

Note: there can be several candidates for flipping; we pick p uniformly at random among such candidates.

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
    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 $I \models S$ **then return** I
  return *don't know*
end

GSAT is a local search algorithm, it tries to maximise the number of satisfied clauses by local changes.

Note: there can be several candidates for flipping; we pick p uniformly at random among such candidates.

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
    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 $I \models S$ **then return** I
  return *don't know*
end

GSAT is a local search algorithm, it tries to maximise the number of satisfied clauses by local changes.

Note: there can be several candidates for flipping; we pick p uniformly at random among such candidates.

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
    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 $I \models S$ **then return** I
  return *don't know*
end

GSAT is a local search algorithm, it tries to maximise the number of satisfied clauses by local changes.

Note: there can be several candidates for flipping; we pick p uniformly at random among such candidates.

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
    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 $I \models S$ **then return** I
  return *don't know*
end

GSAT is a local search algorithm, it tries to maximise the number of satisfied clauses by local changes.

Note: there can be several candidates for flipping; we pick p uniformly at random among such candidates.

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	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 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	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 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	p_2, p_3	p_3
3	0	1	0	4	5	4	4	p_1	p_1
	1	1	0	5					

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	p_2, p_3	p_3
3	0	1	0	4	5	4	4	p_1	p_1
	1	1	0	5					

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	p_2, p_3	p_3
3	0	1	0	4	5	4	4	p_1	p_1
	1	1	0	5					

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	5	4	4	p_1	p_1
	1	1	0	5					

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	5	4	4	p_1	p_1
	1	1	0	5					

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	5	4	4	p_1	p_1
	1	1	0	5					

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 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 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					

Advantages: Can quickly find a satisfying assignment in large problems.

Issues: during the inner loop GSAT can get stuck in a ”plateau” optimum point, where further flips do not change the number of satisfied clauses.

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

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

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

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

Walk SAT (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 ⊨ 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
```

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT (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
    repeat *MAX-FLIPS* times
      randomly select a clause $C \in S$ such that $I \not\models C$
      randomly select a variable p in C
      $I = flip(I, p)$
      if $I \models S$ **then return** I
  return *don't know*
end

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT (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
    repeat *MAX-FLIPS* times
      randomly select a clause $C \in S$ such that $I \not\models C$
      randomly select a variable p in C
      $I = flip(I, p)$
      if $I \models S$ **then return** I
  return *don't know*
end

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT (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
    repeat *MAX-FLIPS* times
      randomly select a clause $C \in S$ such that $I \not\models C$
      randomly select a variable p in C
      $I = flip(I, p)$
      if $I \models S$ **then return** I
  return *don't know*
end

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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	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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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 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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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 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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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 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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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 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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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	p_3
	1	1	0			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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	p_3
	1	1	0			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Walk SAT 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			

Note: first uniformly at random pick an unsatisfied clause and then uniformly at random pick a variable in such clause.

Summary

Probabilistic analysis of satisfiability:

- randomly generated clauses
- main parameter: $r = m/n$; n– number of variables; m number of clauses
- sharp threshold transition of $\pi(r, m)$ from sat to unsat
- crossover point $r \simeq 4.25$
- hard random problems are around the crossover point

Randomized algorithms

- random search for satisfying assignments
- one-sided – can not be used to show unsatisfiability
- GSAT local search
- GSAT with random walks;
- Walk SAT (WSAT)