2SAT

Andreas Klappenecker

The Problem

Can we make the following boolean formula true?

$(\neg x \vee y) \wedge (\neg y \vee z) \wedge (z \vee y)$

Terminology

A boolean variable is a variable that can be assigned the values true (T) or false (F).

A literal is a boolean variable or the negation of a boolean variable.

A clause is a disjunction (or) of literals.

A boolean formula is said to be in conjunctive normal form if and only if it is a conjunction (and) of clauses.

k-CNF

A boolean formula is in k-conjunctive normal form or k-CNF if and only if its is in conjunctive normal form and each clause contains precisely k literals.

Satisfiability

Given a boolean formula f in k-CNF. The boolean formula is called satisfiable if and only if there exists an assignment of truth values to the boolean variables in f such that f evaluates to true.

Example: $(\neg x \vee y) \wedge (\neg y \vee z) \wedge (z \vee y)$ is satisfiable.

The k-SAT problem is to decide whether a given boolean formula in k-CNF is satisfiable or not.

The problem is difficult if $k>2$.

Randomized Algorithm for 2SAT

Monte Carlo Algorithm for 2SAT

Input: Boolean 2-CNF formula f with n variables

Choose an arbitrary truth assignment for the variables in f

Repeat up to $2mn^2$ times or until f is satisfied

- Choose an arbitrary clause that is not satisfied
- Choose uniformly at random one of the literals and switch the truth assignment of its variable.

if a valid truth assignment has been found then return it

else return "formula is unsatisfiable"

Analysis

Suppose that f is a satisfiable boolean 2-CNF formula.

Let S denote a satisfying assignment of f,

let A_i denote the truth assignment after the i-th iteration

We denote by X_i the number of truth assignments of variables where A_i coincides with S, so X_i counts the number of matches.

If $X_i = n$, then the algorithm stops.

Analysis

If $X_i=0$, then $X_{i+1} = 1$. Hence, $Pr[X_{i+1}=1 \mid X_i=0] = 1$.

If $0 < X_i < n$, then

- $Pr[X_{i+1} = k+1 \mid X_i = k] >= 1/2$ (the probability that the number of matches increases could even be 1).
- $Pr[X_{i+1} = k-1 \mid X_i = k] <= 1/2$ (the probability that the number of matches can decrease might even be 0).

The stochastic process $X_0, X_1, X_2, \ldots$ is a bit complicated, as it might depend on the particular choices of truth assignments in the past, and the fact that one or both variables can lead to an improvement.

Analysis

Let's compare our complicated stochastic process with a related but potentially ``slower'' one:

Form a Markov chain $Y_0, Y_1, Y_2, \ldots$ with

$Y_0 = X_0$.

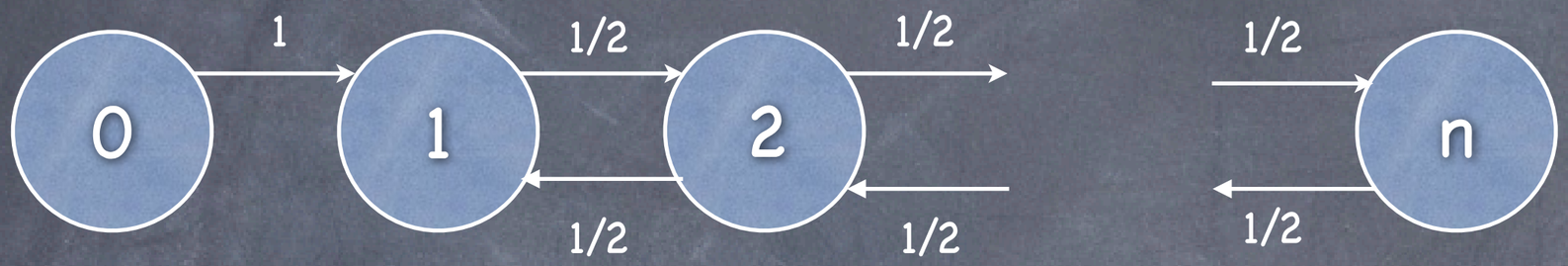
$\Pr[Y_{i+1}=1 \mid Y_i=0] = 1$

$\Pr[Y_{i+1} = k+1 \mid Y_i = k] = 1/2$

$\Pr[Y_{i+1} = k-1 \mid Y_i = k] = 1/2$

This is a pessimistic version of the stochastic process $X_0, X_1, X_2, \ldots$

Random Walk



Let h_j denote the expected number of steps to reach n given that you are in state j (the mean hitting time).

$h_n = 0$

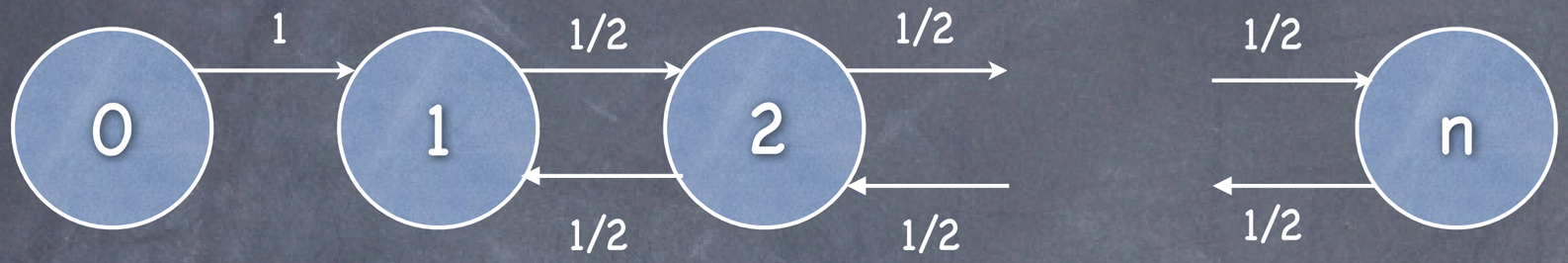
$h_j = 0.5\ h_{j-1} + 0.5\ h_{j+1} + 1$

$h_0 = 1 + h_1$

Solution:

$h_{j+1} = h_j + 2j - 1$

Random Walk



Thus, the expected number of steps to get all n truth assignments correct is

$h_0 = 1 + h_1 = 1 + 3 + h_2 = 1 + 3 + 5 + h_3$

or $h_0 = 1 + 3 + 5 + \ldots + (2n - 1) = n^2$

Conclusion

If the formula f is unsatisfiable, then the algorithm returns correctly that f is unsatisfiable.

If f is satisfiable, then the algorithm returns with probability $1-2^{-m}$ a valid truth assignment.

[Let Z denote the r.v. counting the number of steps until satisfying truth assignment is found. Then $Pr[Z>=2n^2] <= n^2/2n^2 =1/2$ by Markov's inequality. The algorithm repeats this m times.]

Deterministic Algorithm for 2SAT

2-SAT

The problem 2-SAT can be decided in polynomial time.

Proof (1)

Given our boolean 2-CNF formula f, we construct a graph G as follows:

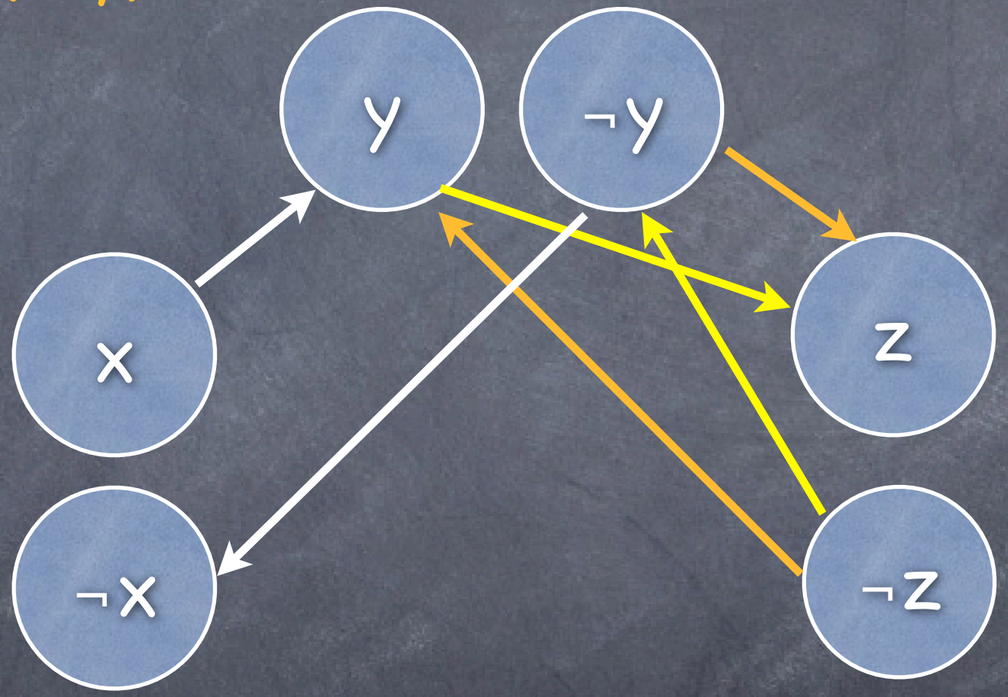
- For each variable x in f, we create two vertices x and ¬x in G.

- For each clause (x ∨ y), we create two edges ¬x -> y and ¬y -> x

We call G the implication graph of f.

Example

$f = (\neg x \vee y) \wedge (\neg y \vee z) \wedge (z \vee y)$



Strongly Connected Components

Compute the strongly connected components of the implication graph G.

[Recall that the strongly connected components form a directed acyclic graph, and can be topologically sorted with partial order $\preccurlyeq$

Let C denote the function that assigns to a variable its strongly connected component. If $x \rightsquigarrow y$, then $C(x) \preccurlyeq C(y)$.]

Unsatisfiable f

If there exists a strongly connected component C of G containing a variable x and its negation ¬x, then f is unsatisfiable.

Since x and ¬x are in C, there exists a cycle <x,…, ¬x, …, x> in G.

If we assign x to be true, then all variables on the cycle must be true, contradicting the fact that ¬x must be false.

If we assign x to be false, then ¬x must be true, and all variables on the cycle must be true, contradicting the fact that x is false.

Satisfiable f

If $C(x) \neq C(\neg x)$ for all variables in f, then assign

- x = true if $C(\neg x) < C(x)$
- x = false if $C(x) < C(\neg x)$

We claim that f evaluates to true under this truth assignment.

Seeking a contradiction, let us assume that f evaluates to false under this truth assignment. Then there must exist a clause $(x \vee y)$ in f that evaluates to false, so both x and y must evaluate to false.

Satisfiable f

Since x and y are assigned false, we have by construction

$C(x) < C(\neg x)$ and $C(y) < C(\neg y)$.

As G contains the edges $\neg x \rightarrow y$ and $\neg y \rightarrow x$, we have

$C(\neg x) \leqslant C(y)$ and $C(\neg y) \leqslant C(x)$.

Therefore, $C(x) < C(\neg x) \leqslant C(y) < C(\neg y) \leqslant C(x)$, a contradiction.

Thus, f must evaluate to true as claimed.

Algorithm

- Create implication graph in adjacency list representation from f in linear time

- Calculate strongly connected components in linear time

- Foreach strongly connected component (taken in topological order), assign truth values to variables as explained. Again, this can be done in linear time.

=> This is a linear time algorithm!

Conclusions

- We showed that there exists a Monte Carlo algorithm for 2SAT that can find a satisfying assignment with probability $1-2^{-m}$ in $2mn^2$ steps.
- We showed that there exists a deterministic linear time algorithm.