ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

M.Sc. Program in Computer Science
Department of Informatics

Design and Analysis of Algorithms

Satisfiability

Vangelis Markakis

markakis@gmail.com

(CNF) SAT variants

2-SAT: Each clause has 2 literals

Horn SAT: Each clause has at most one positive literal

3-SAT: Each clause has 3 literals

Q: Is there a satisfying truth assignment?

NAE 3-SAT

I: A 3-CNF formula

Q: is there a TA with at least one true and one false literal in each clause ?

1 in 3 3-SAT

I: A 3-CNF formula

Q: is there a TA with exactly one true literal in each clause?

MAX 2-SAT

I: A 2-CNF formula of m clauses and an integer $B \leq m$

Q: is there an assignment satisfying at least B clauses ?

MAX k-SAT

I: A k-CNF formula of m clauses and an integer $B \leq m$

Q: is there an assignment satisfying at least B clauses ?

2-SAT

2-SAT

I: A 2-CNF formula ϕ

Q: is there a truth assignment for ϕ ?

E.g. $\phi = (x \vee \neg y) \wedge (x \vee z) \wedge (\neg x \vee \neg z) \wedge (\neg y \vee w)$

# of variables of ϕ = n

# of clauses of ϕ = m

We will solve 2-SAT through a graph-theoretic approach

Construct a directed graph G=(V,E), with

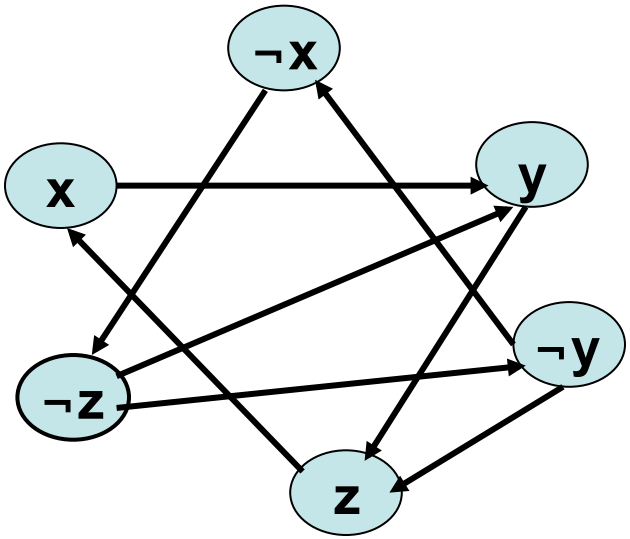
V = { a, $\neg$a | a is a variable of ϕ}

E = { ($\neg$a,b), ($\neg$b,a) : (a $\vee$ b) is a clause of ϕ}

|V|= 2n, |E|= 2m

2-SAT

Example: $\phi = (\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$



$V = \{ a, \neg a \mid a \text{ is a variable of } \phi \}$

$E = \{ (\neg a, b), (\neg b, a) \mid (a \vee b) \text{ is a clause of } \phi \}$

If $(x, y) \in E$ then $(\neg x \vee y) \in C$ or $(y \vee \neg x) \in C$ => $(\neg y, \neg x) \in E$

Therefore, if there is a path $x_1, x_2, \ldots, x_{i-1}, x_i$ in G,
then there is also a path $\neg x_i, \neg x_{i-1}, \ldots, \neg x_2, \neg x_1$ (symmetry of G)

2-SAT

Theorem: ϕ is unsatisfiable iff there is a variable x s.t. there are paths from x to $\neg x$ and from $\neg x$ to x in G

<u>If there is a variable x s.t. both paths exist => there is no TA</u>

- assume that there is a TA, T such that ϕ=true
- assume T(x) = true, then T($\neg$x) = false (same arguments in the other case)
- => there is a path from x to $\neg x$ and T(x)=true, T($\neg$x) = false
- => there is some edge (a, b) s.t. T(a)=true and T(b)=false
- => there is a clause $(\neg a \vee b) \in C$
- then, $T(\neg a) \vee T(b)$ = false $\vee$ false = false
- Thus, ϕ = false, a contradiction

2-SAT

Theorem: ϕ is unsatisfiable iff there is a variable x s.t. there are paths from x to $\neg x$ and from $\neg x$ to x in G

```
Algorithm 2-SAT(φ)
Construct G;
If there is a variable x s.t. there are
   paths from x to ¬x and from ¬x to x
   then answer NO
   else answer YES
```

2 questions:

- Q1: How do we find a truth assignment, when ϕ is satisfiable?
- Q2: How can we check efficiently the if condition ?

2-SAT

Q1: How do we find a truth assignment, when ϕ is satisfiable?

There is no variable s.t. both paths exist => there is a TA
We construct a TA by assigning repeatedly values to nodes of G:

Repeat

- **Pick an unassigned node x of G s.t. there is no path from x to ¬x ;** **(1)**
- **T(x) = true**
- **Assign true to all vertices reachable from x (successors of x)**
- **Assign false to their negations;**
  (they correspond to the nodes from which ¬x is reachable, i.e., the predecessors of ¬x) **(2)**

Until all vertices are assigned values

2-SAT

No edge of G goes from true to false and, hence, all clauses are satisfied

- Suppose that there are paths from x to both y and ¬y.
  Then, there are also paths from both ¬y and y to ¬x
  => there is also a path from x to ¬x, a contradiction to (1)
  Hence, the algorithm will not assign true to a literal and its negation

- Suppose that there is a path from x to a node y already assigned false.
  Then, x is a predecessor of y and was also assigned false by (2)

  Whenever a node is assigned true
  all its successors are also assigned true
  Whenever a node is assigned false
  all its predecessors are also assigned false

Hence, there is no edge from true to false and T is a TA satisfying ϕ.

2-SAT

- Q2: How can we check efficiently the if condition ? (i.e., if there is a path from x to ¬x and vice versa)

- We have seen how to find strongly connected components
- We can run the algorithm for finding them and also assign a number to each of them
- Let scc(x) = id of strongly connected component that contains x

Algorithm for checking the "if" condition

```
Construct G;
Find the SCCs of G and assign to each node u a SCC
    number scc(u);
For each pair of variables x, ¬x
    if scc(x) =scc(¬x) then answer NO and halt
Answer YES (and construct a TA)
```

Complexity: O(n+m)

2-SAT

Summarizing...

Theorem: 2-SAT can be solved in polynomial time, namely in time O(n+m). Furthermore, a truth assignment can also be computed in polynomial time when a formula is satisfiable

MAX SAT

The optimization version of SAT problems:

MAX SAT (optimization version)
I: A CNF formula ϕ of m clauses
Q: find a truth assignment satisfying the maximum possible number of clauses

Restrictions of MAX SAT:

MAX k-SAT (optimization version)
I: A k-CNF formula ϕ of m clauses
Q: find a truth assignment satisfying the maximum possible number of clauses

We can also have weights on the clauses and try to maximize total weight

Theorem: Even for k =2, MAX k-SAT is NP-complete

Randomized Algorithms

- All the algorithms we have seen so far are deterministic
- For a given input a deterministic algorithm will run in the same way and have the same output

Randomized algorithms:

- Algorithms that flip coins during their execution
- They generally do not produce the same output if you run them twice on the same input
- What may be affected by the randomization:
- Running time: Can be polynomial time or polynomial time in expectation or polynomial time with high probability
- Output: Can be wrong (with a small probability)
- Approximability: can be in expectation

Overall a very powerful tool

- For some problems we only know randomized efficient algorithms
- Sometimes deterministic algorithms can be obtained by first designing a randomized algorithm

MAX k-SAT

A randomized approximation algorithm for MAX k-SAT:

Independently set each variable to: $\begin{cases} 1, & \text{with probality } \frac{1}{2} \\ 0, & \text{with probability } \frac{1}{2} \end{cases}$

Each clause contains c_i, $1 \le b \le c_i \le k$, distinct literals

- important: no clause contains a variable and its negation
- Hence, the truth assignment to each literal is independent of the other ones

$$\Pr[C_i = 0] = \frac{1}{2} \cdot \frac{1}{2} \cdots \frac{1}{2} = \frac{1}{2^{c_i}}, \qquad \Pr[C_i = 1] = 1 - \frac{1}{2^{c_i}} \ge 1 - \frac{1}{2^b} \ge \frac{1}{2}$$

MAX k-SAT

$X = \#$ of true clauses (random variable)

$$X = X_1 + X_2 + ... + X_m = \sum_{i=1}^{m} X_i, \qquad X_i = \begin{cases} 1, & \text{if } C_i = 1 \\ 0, & \text{if } C_i = 0 \end{cases}$$

$$E[X_i] = 1 \cdot \Pr[C_i = 1] + 0 \cdot \Pr[C_i = 0] = 1 - \frac{1}{2^{c_i}} \geq 1 - \frac{1}{2^b}$$

$$E[X] = E\left[\sum_{i=1}^{m} X_i\right] = \sum_{i=1}^{m} E[X_i] \geq \sum_{i=1}^{m} (1 - \frac{1}{2^b}) = (1 - \frac{1}{2^b})m$$

$OPT =$ maximum # of true clauses, $\quad OPT \leq m$

MAX k-SAT

$$\frac{E[X]}{OPT} \geq \frac{E[X]}{m} = 1 - \frac{1}{2^b} \geq \frac{1}{2}$$ randomized approximate solution

If we know that each clause contains exactly k literals, then we can guarantee a better estimate:

$$\text{If } b = c_i = k, \forall i, \text{ then } \frac{E[X]}{OPT} \geq \frac{E[X]}{m} = 1 - \frac{1}{2^k}$$

MAX 3-SAT

For 3-CNF formulas:

$$\frac{E[X]}{OPT} = 1 - \frac{1}{2^3} = \frac{7}{8} = 0.875$$

Fact 1: for every instance of MAX 3-SAT, the expected # of clauses satisfied by a random assignment is at least 7/8 of the optimal

MAX 3-SAT

Fact 1: for every instance of MAX 3-SAT, the expected # of clauses satisfied by a random assignment is at least 7/8 of the optimal

But, for any random variable, there is some point at which it assumes a value at least as large as its expectation

Thus:

Fact 2: for every instance of 3-SAT, there is an assignment satisfying at least 7/8 of all clauses !

And this implies:

Fact 3: every instance of 3-SAT with at most $m \leq 7$ clauses is satisfiable !

Proof:

- By Fact 2, there is an assignment satisfying at least $t = 7/8m$ clauses
- For $m < 8$, it holds that $t = 7/8m > m-1$,
- As t is an integer, it follows that $t = m$!

MAX 3-SAT

Fact 2: for every instance of 3-SAT, there is an assignment satisfying at least 7/8 of all clauses !

How can we find such an assignment? What is the complexity ?
How long will it take until we find one by random trials ?

Waiting for the first success: Z= # of trials until success (a random variable)

p = Pr[a random assignment satisfies at least 7/8m clauses]

Pr[Z=j] = probability for success in the j-th trial

Pr[Z=j] = $(1-p)^{j-1}p$

$$E[Z] = \sum_{j=1}^{\infty} j \Pr[Z = j] = \sum_{j=1}^{\infty} j(1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=1}^{\infty} j(1-p)^{j}$$

$$= \frac{p}{1-p} \frac{(1-p)}{p^2} = \frac{1}{p}$$

In expectation 1/p random trials suffice

MAX 3-SAT

Can we bound 1/p ?

X = # of satisfied clauses by a random assignment; recall E[X] = 7/8m

p_j = Pr[a random assignment satisfies exactly j clauses]

Let m' = the largest integer less than 7/8m, m' < m

$$\sum_{j<7m/8} p_j = 1-p, \quad \sum_{j\geq 7m/8} p_j = p$$

$$\frac{7}{8}m = E[X] = \sum_{j=1}^{m} jp_j = \sum_{j<7m/8} jp_j + \sum_{j\geq 7m/8} jp_j \leq \sum_{j<7m/8} m' p_j + \sum_{j\geq 7m/8} mp_j$$

$$= m'(1-p) + mp = m' + (m-m')p \leq m' + mp$$

MAX 3-SAT

$$\frac{7}{8}m \le m' + mp \Rightarrow p \ge \frac{7/8m - m'}{m}$$

$$m' = \text{largest integer} < 7/8m \Rightarrow 7/8m - m' \ge 1/8$$

$$\text{Thus, } p \ge \frac{1}{8m} \Rightarrow \frac{1}{p} \le 8m$$

Theorem: there is a randomized algorithm with expected complexity O(m) for finding an assignment satisfying at least 7/8 of all clauses of a 3-SAT instance !

MAX 3-SAT

- Actually it can get even better
- We can “derandomize” the algorithm
- Again use the fact that there is always a truth assignment that achieves at least what the expectation says
- Method of conditional expectations:
  - Try to set a variable x to 0 or 1
  - One of the two conditional expectations (i.e., given that x =0 or x = 1) has to be at least as large as the original expectation
  - If we know how to compute conditional expectations we are done essentially

Hence: there exists a deterministic 7/8-approximation for MAX 3-SAT