

Let us now investigate some variants of the SAT problem and see how various *parameters* may affect the complexity of a problem.

1

Goal:

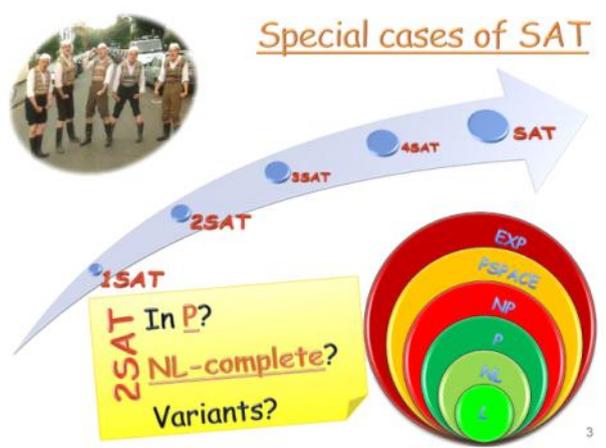
- Discuss the complexity of variants of SAT

Plan:

- General
- 2SAT
- Max2SAT

In particular, we study the *2SAT problem* and its optimization version.

2



Special cases of SAT

1SAT In P?
2SAT NL-complete?
Variants?

3SAT 4SAT SAT

EXP
PSPACE
NP
P
NL
L

We can limit the *number of literals* in each *clause* of the SAT problem.

We have already seen that with 3 literals the problem is NP-complete.

This is clearly still the case when the number of literals is even larger.

What about 2?

PAP 184-185

2SAT

2SAT Instance:

- a 2-CNF formula ϕ EG $(\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3)$

Decision Problem:

- is ϕ satisfiable?

Theorem:

- $2SAT \in P$

Proof:

- Reduce **2SAT** to a graph problem in **P**: construct G_ϕ -- then specify problem

A 2SAT instance is a 2-CNF formula, which is good (accepted) if it can be satisfied by some assignment to its variables.

We'll now show *it is in P* by *reducing* it to a graph problem in P.

Implication graph $G_\varphi = (V_\varphi, E_\varphi)$

- V_φ**
 - 1 vertex for every literal of φ
- E_φ**
 - note** edges: $(\alpha, \beta) \in E_\varphi \iff (\neg\beta, \neg\alpha) \in E_\varphi$
paths: $\alpha \mapsto \beta \iff \neg\beta \mapsto \neg\alpha$
 - edge $(\alpha, \beta) \iff \varphi$ contains clause $(\neg\alpha \vee \beta)$
- Theorem:**
 - note** $\alpha \mapsto \beta \iff \alpha \mapsto \beta$
 - φ is unsatisfiable $\iff \exists x$ s.t. $x \mapsto \neg x$ and $\neg x \mapsto x$ in G_φ

An *implication graph* for a given formula has a vertex for every literal of the formula. Namely, 2 for each variable.

For a clause to be satisfied, if one of its literals is FALSE, the other one must be TRUE. Edges in the graph correspond to such restrictions, i.e., 2 for each clause.

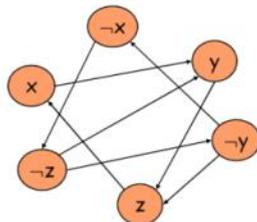
To satisfy all clauses, if a literal is assigned TRUE, all the outgoing edges from its corresponding vertex, enforce the adjacent literals to be TRUE as well.

A problematic cycle is a cycle that contains both x and $\neg x$ for some variable x .

We'll now prove that the formula can be satisfied if and only if its implication graph has no problematic cycle.

Implication graph : Example

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



Com

6

Here is an example of the implication graph of a simple formula.

Correctness

Completeness:

- $x \mapsto \neg x$ can't assign TRUE to x
- $\neg x \mapsto x$ can't assign FALSE to x

Soundness:

- Repeat
Pick an x ; if $x \mapsto \neg x$, $\alpha = \neg x$ o/w $\alpha = x$ -
no $\alpha \mapsto \neg \alpha$, hence assign TRUE to α ;
Then, \forall literal β s.t. $\alpha \mapsto \beta$:
assign TRUE to β and FALSE to $\neg \beta$

- No inconsistencies!

note $\alpha \mapsto \beta \wedge \alpha \mapsto \neg \beta$
 $\Rightarrow \alpha \mapsto \neg \alpha$

The completeness proof is straightforward: if a problematic cycle contains x and $\neg x$, then x can be assigned neither TRUE nor FALSE. Hence a formula that can be satisfied results in a graph with no problematic cycles.

As to soundness, with no problematic cycles, one can construct a satisfying assignment: Assign an arbitrary x with a value that is not contradicted by a path from x to $\neg x$ or vice versa. Now assign all values that are implied by this.

With no problematic cycles there can be no contradictions in this process.

If not done, pick another variable and proceed in the same manner.

Graph Connectivity (CONN)

CONN Instance:

- a directed graph $G=(V, E)$ and 2 vertices $s, t \in V$

Decision Problem:

- Is there is a path from s to t in G ?

Theorem:

- $CONN \in P$ Apply some search algorithm (DFS/BFS)

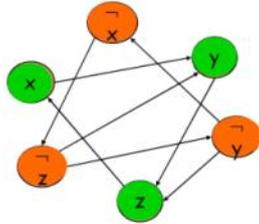
Corollary:

- " $\exists x$ s.t. $x \mapsto \neg x$ and $\neg x \mapsto x$ in G_0 " $\in P$ ■

The property we have reduced our problem to, constitutes of a set of *connectivity problems*. Therefore, it is in P .

An Assignment: example

- Construct an assignment as follows:



Com

9

Here is an example of the assignment being constructed.

Max-2-SAT

Instance:

- a 2-CNF formula φ

Maximization Problem:

- Find the **maximum** # of clauses satisfied by an assignment to φ

Instance (decis. ver.):

- a 2-CNF formula φ and a *threshold* K

Decision Problem:

- Is there an assign. satisfying $\geq K$ clauses of φ ?

Now consider the problem of, given a 2-CNF formula, *maximize* the number of clauses satisfied.

To better analyze this problem we translate it into a decision problem. We add to the input a threshold K , and simply ask whether there exists an assignment that satisfies at least K of the clauses.

We call this problem *Max-2-SAT*.

Max2SAT NPC

Theorem:

- Max2SAT is NP-hard

note clearly
Max2SAT ∈ NP

Proof: 3SAT_φ → Max2SAT

- Replace each $C=(\alpha\vee\beta\vee\gamma)$ of ϕ w/10 clauses in ϕ' :
 $(\alpha)\wedge(\beta)\wedge(\gamma)\wedge(w_c) \wedge (\neg\alpha\vee\neg\beta)\wedge(\neg\beta\vee\neg\gamma)\wedge(\neg\gamma\vee\neg\alpha) \wedge$
 $(\alpha\vee\neg w_c)\wedge(\beta\vee\neg w_c)\wedge(\gamma\vee\neg w_c).$

- Set $K=7|\phi|$.

note $w_c = "a=b=\gamma=TRUE?"$
maximizes satisfiab.

Completeness:

- $C=(\alpha\vee\beta\vee\gamma)$ satisfied \Rightarrow 7/10 clauses satisfied

Soundness:

- $C=(\alpha\vee\beta\vee\gamma)$ unsatisfied $\Rightarrow \leq 6/10$ clauses satisfied

This problem is shown to be *NP-hard* by a simple reduction from 3SAT.

Let us replace every clause by 10 clauses (plus an extra variable), so that at the most 7 of the 10 can be satisfied. And so that, in case the original clause is satisfied, exactly 7 of the 10 can be satisfied. While, in case the original clause is not satisfied, exactly 6 clauses can be satisfied.

To be convinced this is indeed the case, note that setting the auxiliary variable to TRUE only if all 3 literals are TRUE maximized the number of satisfied clauses. Now consider cases according to the number of literals TRUE in the original clause.

Synopsis



Discussed variants of SAT
Also: Maximization Problems



Special cases of NPC problems may be in P: SAT vs. 2SAT
Optimization versions of problems in P may be hard: 2SAT vs. Max-2-SAT

We have discussed some SAT problems.

More importantly, we have introduced the notion of an *optimization* problem, and have investigated the complexity of the optimization version of 2SAT.

We'll come back to this notion later and study it extensively.

SAT	Max-2-SAT	NPC
2SAT	Max-2-SAT	NPC
	NL Complete	NP-Hard
Complexity Classes	NP	NL
P	L	co-NP
EXPTIME	PSPACE	

WWinex



Papadimitriou,
Christos

13

- [SAT](#)
- [Max-2-SAT](#)
- [NPC](#)
- [2SAT](#)
- [NL Complete](#)
- [Complexity Classes](#)
- [P](#)
- [NP](#)
- [co-NP](#)
- [EXPTIME](#)
- [L](#)
- [NL](#)
- [PSPACE](#)
- [NPC](#)
- [NP-Hard](#)
- [Max-2-SAT](#)