ADA23-HW4

許博翔

December 1, 2023

1 2 SAT, or not 2 SAT, That Is the Question

(a) 2-SAT is in P

A 2-CNF formula is a CNF formula where each clause has exactly 2 literals. For example:

$$\phi_1 = (a_1 \vee \sim a_2) \wedge (a_3 \vee a_1) \wedge (\sim a_1 \vee \sim a_1)$$

$$\phi_2 = (a_1 \vee \sim a_1)$$

are both 2-CNF formulas.

2-SAT (or 2-CNF-SAT) $:= \{\phi | \phi$ is a 2-CNF formula with a satisfying assignment$\}$

For example:
ϕ_1 can be satisfied if $(a_1, a_2, a_3) = (0, 0, 1)$; therefore $\phi_1 \in$ 2-SAT.
ϕ_2 does not have a satisfying assignment; therefore $\phi_2 \notin$ 2-SAT.

A 2-SAT problem is to determine whether a 2-CNF formula ϕ is in 2-SAT or not, and the size of this problem is defined as the number of clauses.

Prove that 2-SAT is in P.

Hint

Let ϕ be a 2-CNF formula with variables $a_1, a_2, \ldots, a_n$. Consider a directed graph G where $V(G) = \{a_1, a_2, \ldots, a_n, \sim a_1, \sim a_2, \ldots, \sim a_n\}$ is the set of all literals. For each clause $x \vee y$ of ϕ, construct directed edges from $\sim x$ to y and from $\sim y$ to x. For example, the following is the graph of ϕ_1 (which is described above):

$$\begin{tikzcd}
a_2 \arrow[r] & a_1 \arrow[d] & \sim a_3 \arrow[l] \\
\sim a_2 & \sim a_1 \arrow[l] \arrow[r] & a_3
\end{tikzcd}$$

First prove that

All clauses can be satisfied.

$$\Updownarrow$$

For all variables x, either x can't reach $\sim x$ or $\sim x$ can't reach x.

Then find an polynomial time complexity algorithm to check if G satisfies the transformed condition, deducing that 2-SAT is in P.

(b) Conditions on MAX-2-SAT

A MAX-2-SAT problem is to find the truth values of the variables such that the number of satisfied clauses in a given 2-CNF formula is maximized. The decision version of MAX-2-SAT problem is to determine whether at least k clauses in a 2-CNF formula can be satisfied. That is:

$$\text{MAX-2-SAT} := \{(\phi, k) | \phi \text{ is a 2-CNF formula with an assignment such that at least } k \text{ clauses can be satisfied}\}$$

It is known that MAX-2-SAT is in NP-complete, but some condition on the clauses may make the problem easier.

Let f be a given single-variable function. The set S_f is defined as the collection of all 2-CNF formulas whose variables can be numbered as $a_1, a_2, \ldots, a_n$ such that for all pairs of variables (a_i, a_j) that are in the same clause, there is $|i - j| < f(n)$. For example, let $f(x) := \sqrt{x}$.

$$\phi_3 = (b_1 \vee b_2) \wedge (\sim b_1 \vee b_3)$$

$$\phi_4 = (b_1 \vee b_2) \wedge (b_2 \vee b_3) \wedge (\sim b_3 \vee b_1)$$

One can number the variables in ϕ_3 by letting $a_1 = b_2, a_2 = b_1, a_3 = b_3$, and ϕ_3 becomes $(a_2 \vee a_1) \wedge (\sim a_2 \vee a_3)$. Since $|2-1| < \sqrt{3}, |2-3| < \sqrt{3}$, there is $\phi_3 \in S_{\sqrt{x}}$. However, no matter how the variables in ϕ_4 are numbered to a_1, a_2, a_3, there always exists a clause that contains both a_1 and a_3. Since $|1-3| \not< \sqrt{3}$, there is $\phi_4 \notin S_{\sqrt{x}}$.

Let's define the MAX-f-CondSAT problem, which is to find the truth values of the variables such that the number of satisfied clauses in a given $\phi \in S_f$ is maximized. The decision version of MAX-f-CondSAT problem is to determine whether at least k clauses can be satisfied. That is:

$$\text{MAX-}f\text{-CondSAT} := \{(\phi, k) | \phi \in S_f \text{ has an assignment such that} \\ \text{at least } k \text{ clauses can be satisfied}\}$$

(b-1)

Let $f(x) := x^{0.001}$. Prove that MAX-f-CondSAT is NP-complete.

Hint

Can you provide a reduction from a known NP-complete problem: MAX-2-SAT?

(b-2)

Let $f(x) := \log x$. Prove that MAX-f-CondSAT is P.

Hint

MAX-1-CondSAT can be easily done by dynamic programming in polynomial time complexity. How about MAX-2-CondSAT, MAX-3-CondSAT, …, and so on? Can this dynamic programming method be generalized to MAX-f-CondSAT?

(c) 2-SAT with Generalized Boolen Values

In a 2-SAT problem, all variables take values in $\{True, False\}$, but what if the variables take values in a set with size larger than 2? Will the problem become harder?

Let c be a positive integer. Define the notation $[c] := \{1, 2, \ldots, c\}$.

(c-1)

The set T_c is defined as the collection of all formulas ϕ that is the disjunction of some clauses, and each clause of ϕ is the conjunction of two literals, where each literal is of the form $a = b$. a is a variable taking values in $[c]$, while b is an element of $[c]$.
For example, let $c = 3$.

$$\phi_5 = (a_1 = 3 \lor a_2 = 2) \land (a_3 = 1 \lor a_2 = 3)$$

$$\phi_6 = (\sim (a_1 = 3) \lor a_2 = 2)$$

$\phi_5 \in T_c$, but $\phi_6 \notin T_c$ because $\sim (a_1 = 3)$ is not of the form $a = b$ for a variable a and a constant b taking values in $[c]$.

Let's define the c-GenSAT problem, which is to determine whether all clauses in a given $\phi \in T_c$ can be satisfied. That is:

$$c\text{-GenSAT} := \{\phi | \phi \in T_c \text{ has a satisfying assignment}\}$$

Prove that c-GenSAT is P for all positive integers c.

Hint

Can you provide a reduction to a known P problem: 2-SAT?

(c-2)

The set U_c is defined as the collection of all formulas ϕ that is the disjunction of some clauses, and each clause of ϕ is the conjunction of two literals, where each literal is of the form $a = B$. a is a variable taking values in $[c]$, while B is a subset of $[c]$.
For example, let $c = 3$.

$$\phi_7 = (a_1 \in \{1, 2\} \lor a_2 \in \{3\}) \land (a_3 \in \{1, 2, 3\} \lor a_2 \in \emptyset)$$

$$\phi_8 = (a_1 \in \{3,4\} \vee a_2 \in \{2,3\})$$

$\phi_7 \in T_c$, but $\phi_8 \notin T_c$ because $\{3,4\}$ is not a subset of $[c]$.

Let's define the c-GenSetSAT problem, which is to determine whether all clauses in a given $\phi \in U_c$ can be satisfied. That is:

$$c\text{-GenSetSAT} := \{\phi | \phi \in U_c \text{ has a satisfying assignment}\}$$

Prove that c-GenSetSAT is NP-complete for all integers $c \geq 3$.

Hint

Can you provide a reduction from a known NP-complete problem: 3-colorability problem?

(a)

Claim: All clauses can be satisfied $\iff$ for all variables x, either x can't reach $\sim x$ or $\sim x$ can't reach x.

Proof:

$(\Rightarrow)$:

Since $a \vee b$ is true $\iff \sim a \to b$ is true, and if $a \to b$ and $b \to c$, then $a \to c$.

$\therefore$ a can reach b on the graph means that $a \to b$ is true.

If x can reach $\sim x$, then $\sim x \vee \sim x$ (which is $x \to \sim x$) is true.

$\Rightarrow x = 0$.

Similarly, if $\sim x$ can reach x, then $x = 1$.

$\therefore$ either x can't reach $\sim x$ or $\sim x$ can't reach x.

$(\Leftarrow)$:

Consider the DAG of SCC.

Repeat choosing SCC with no out degree and assign all vertices in the SCC with true, and then remove that SCC from the DAG until we can't do that (a is assigned to be true but $\sim a$ is in the SCC). Then we assign all left SCC with false.

Proof of $a, \sim a$ won't be false simultanously: if a can't be true, it means that there is b where a can walk to b such that b is false and it's because of $\sim b$ is assigned true. Since this graph is symmetric ($a \to b \iff \sim b \to \sim a$), $\sim b$ can walk to $\sim a$ and therefore $\sim a$ has been assigned true.

It is equivalent to check if for each a, $\sim a$ is in the same SCC or not by the claim, which can be done by using Kosaraju's algorithm to find all SCC (with time complexity $O(n \log n)$) and then check the SCC that a, $\sim a$ are in for each a (with time complexity $O(n)$).
This is an algorithm that runs in polynomial time, and therefore 2-SAT is in P.

(b)

(b-1)

Let ϕ be any 2-CNF formula, and $n(\phi) :=$ the number of variables in ϕ.
Let ϕ' be a 2-CNF with $n(\phi)^{1000}$ variables, and ϕ''s clauses := the union of the clauses in ϕ and $\{x \vee \sim x : \text{for all variables } x \text{ of } \phi'\}$.
One can see that $\phi' \in S_{x^{0.001}}$ by numbering the first $n(\phi)$ variables being those in ϕ.
Since $x \vee \sim x$ is always true, the number of satisfied clauses in ϕ = the number of satisfied clauses in $\phi' - n(\phi)^{1000}$.
$\therefore (\phi, k) \in$ MAX-2-SAT $\iff (\phi', k + n(\phi)^{1000}) \in$ MAX-f-CondSAT.
Also, ϕ' can be constructed from ϕ in complexity $n(\phi)^{1000}$, which is polynomial complexity.
$\therefore (\phi, k) \to (\phi', k + n(\phi)^{1000})$ is a polynomial reduction. Since MAX-2-SAT is NP-complete, MAX-f-CondSAT is also NP-complete.

(b-2)

Sort all clauses by the maximum index of its variables.
Let $k := \lfloor f(n) \rfloor$, and $j = (j_k j_{k-1} \cdots j_0)_2$.
Let $dp[i][j] :=$ the maximum number of clauses with both its variables in $\{a_1, a_2, \ldots, a_i\}$ are satisfied given the condition $a_{i-l} = j_l,\ \forall 0 \le l \le k$.
$dp[i][j] = \max(dp[i-1][j/2], dp[i-1][j/2+1]) +$ (the number of clauses containing a_i, a_{i-l} are satisfied when $a_i = j_0$ and $a_{i-l} = j_l$ for all l).

After calculating the values of all $dp[i][j]$ in the order $i = 1$ to n, $j = 0$ to $2^{k+1} - 1$, find the maximum value of $dp[n][j]$ among all j.

The time complexity of calculating a single $dp[i][j]$ is $O(k)$.

Suppose that $f(n) < c \log n$ for all n.

$\therefore$ the total time complexity $= O(n \times 2^k \times k) = O(n \times 2^{c \log n} c \log n) = O(n^{c+1} \log n)$, which is a polynomial time complexity.

(c)

(c-1)

Suppose that the variables used in ϕ are $a_1, a_2, \ldots, a_n$.

Let ϕ' contains boolen variables $a_{i,j}$ for $i \in [n]$ and $j \in [c]$.

For all i and $j_1 \neq j_2$, add the clause $\sim a_{i,j_1} \vee \sim a_{i,j_2}$ to ϕ'. These clauses are called type 1.

For each clause $a_i = b_1 \vee a_j = b_2$ of ϕ, add the clause $a_{i,b_1} \vee a_{j,b_2}$ to ϕ'. These clauses are called type 2.

For an assignment $a_1, a_2, \ldots, a_n$ such that ϕ is satisfied, set $a_{i,j} = \mathbb{I}\{a_i = j\}$. One can see that the clauses in type 2 are satisfied, and the clauses in type 1 are satisfied because $a_i = j_1, a_i = j_2$ won't be true simultanously for $j_1 \neq j_2$.

For an assignment of all $a_{i,j}$ such that ϕ' is satisfied, set $a_i = \begin{cases} j, \text{ if } \exists j \text{ s.t. } a_{i,j} \text{ is true} \\ 1, \text{otherwise} \end{cases}$.

Such existence of j is unique because $\sim a_{i,j_1} \vee \sim a_{i,j_2}$ for all $j_1 \neq j_2$. If $a_{i,j}$ is false for all j, then assigning $a_{i,1} = 1$ won't change any clause of ϕ' to false.

Since $a_{i,b_1} \vee a_{j,b_2}$ is true implies $a_i = b_1 \vee a_j \vee b_2$ is true, all clauses in ϕ are satisfied.

$\therefore \phi \in c-$GenSAT $\iff \phi' \in$ 2-SAT.

Also, constructing ϕ' from ϕ takes polynomial time complexity.

$\therefore \phi \to \phi'$ is a polynomial reduction. Since 2-SAT is in P, c-GenSAT is also in P.

(c-2)

Let's have a reduction from 3-colorability problem to c-GenSetSAT.

For a graph G with vertices $v_1, v_2, \ldots, v_n$, construct a correspond formula ϕ with variables $a_1, a_2, \ldots, a_n$. If $v_iv_j \in G$, add the three clauses $a_i \in \{1\} \vee a_j \in \{2,3\}, a_i \in \{2\} \vee a_j \in \{3,1\}, a_i \in \{3\} \vee a_j \in \{1,2\}$ to ϕ.

One can see that the above three clauses are all satisfied iff $a_i \neq a_j$ and $a_i, a_j \in \{1,2,3\}$.

Therefore, ϕ is true iff $a_i \neq a_j$ for all edges v_iv_j iff we can color v_i in a_i for some a_i.

$\therefore G$ can be 3-colored iff $\phi \in$ 3-GenSetSAT.

Also, constructing ϕ from G takes polynomial time complexity.

$\therefore G \to \phi$ is a polynomial reduction. Since c-colorability problem is NP-complete, 3-GenSetSAT is also NP-complete.