CS 3510 — Algorithms, Spring 2022

Lecture 18: SAT Problem

March 17, 2022

Lecturer: Frederic Faulkner *Scribe: Aditya Diwakar*

1 SAT$\rightarrow$ 3SAT

In the last lecture, we showed that every SAT problem can be converted into a 3SAT problem. Hence, if we have some polynomial solution for 3SAT, then we can get a polynomial time solution for SAT.

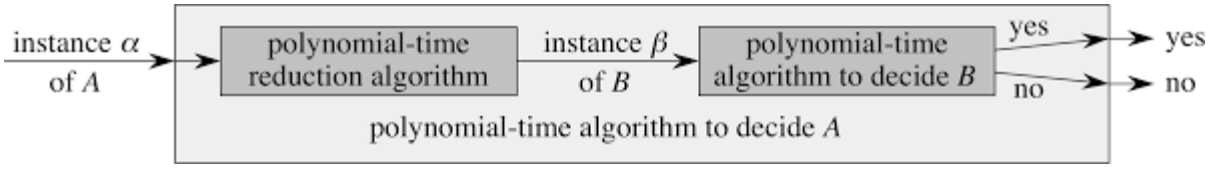
Presume we had some polynomial time solver for 3SAT and want to design a solver for the SAT problem. We take as input some SAT problem and convert it to a 3SAT and then simply run the 3SAT polynomial time algorithm on this modified input. We can simply adjust the output and return that assignment.

All of these steps are polynomial in nature. Converting from SAT to 3SAT is polynomial and we presumed that the 3SAT solver was polynomial, hence the entire algorithm here is polynomial.

Unfortunately, there is no known 3SAT polynomial time solver.

2 Reductions

The process of converting one problem to another is called a reduction. This is where we have a problem of type A and an algorithm for type B, so we find some way to convert the type A problem into type B.



Theorem. *If exists a polytime reduction from $A \to B$, then $B \in P \implies A \in P$.*

Definition. A problem is considered NP-Complete if every problem in NP can be reduced to A and $A \in NP$.

Definition. A problem A is NP-Hard if every problem in NP can be reduced to A in polynomial time.

Theorem. *If there is a polynomial time algorithm for A and A is NP-Complete, then $P = NP$ as every problem in NP can be reduced to A and $A \in P$ so $\forall x \in NP, x \in P \implies P = NP$.*

In other words, since every problem can be reduced to A and A can be solved in polynomial time, then every NP problem can be solved in polynomial time and therefore the NP and P sets are the same $(P = NP)$.

Theorem. *CNF-SAT is NP-Complete.*

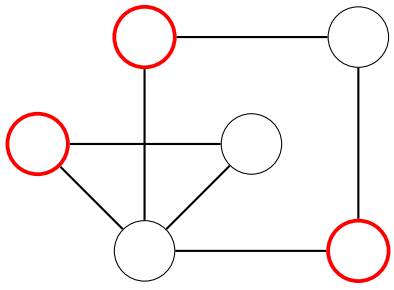
Proof. Omitted.

Corollary. *3-SAT is NP-Complete.*

Proof. By definition, a problem is considered NP-Complete if every problem in NP reduces to this problem and this problem is in NP. We know that SAT reduces to 3-SAT and 3-SAT is in NP. Since every problem reduces to SAT, then every problem reduces to 3SAT, hence 3-SAT is NP-Complete.

3 Independent Set

We say $S \subseteq V$ is an independent set if no edges connect any of the vertices in S. In other words, S is a set of vertics in G such that no edge of G has both endpoints in S. Formally: for $u, v \in S \implies (u, v) \notin E$. For example, the red vertices on the graph on the nezt page form an independent set for the graph.



Problem Given a graph G and a goal g, is there an independent set of size $\geq g$?

Hence, we have written the problem as a decision problem. Now, let us prove that this problem is NP-Complete.

Theorem 3.1. *Independent Set is NP-Complete.*

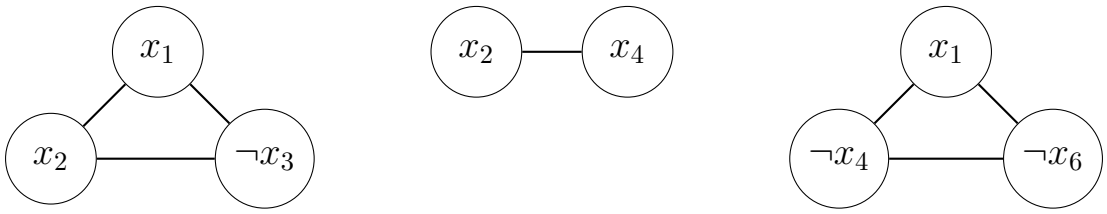
Proof. First, we claim that Independent Set is in NP as we can verify a candidate solution in polynomial time. Simply check the size of the set is $\geq g$ and that there are no edges in E where both endpoints are in the candidate solution set.

Now, we want to reduce from some problem to independent set. Let's pick 3-SAT as 3-SAT has a lot of structure. This means we take some input (F) of clauses and literals ($x_1, \ldots, x_n$ and $c_1, \ldots, c_m$) and we are going to output some graph G with $g = m$ (g being the parameter for independent set). We are going to fix the fact that F is satisfiable if and only if G has an independent set of size m.

To start, for each clause c_i, we make a subgraph with $|c_i|$ vertices labeled by literals. For example, if we have:

$$(x_1 \vee x_2 \vee \neg x_3) \qquad (x_2 \vee x_4) \qquad (x_1 \vee \neg x_4 \vee x_6)$$

then we will create the following graphs:



We want the independent set of this graph to correspond to a satisfying assignments. In the above graph, clause edges are also included (edges between vertices in the same clause). However, we can make an independent set in the above graph that does not make a satisfying assignment, so we want to add additional edges.

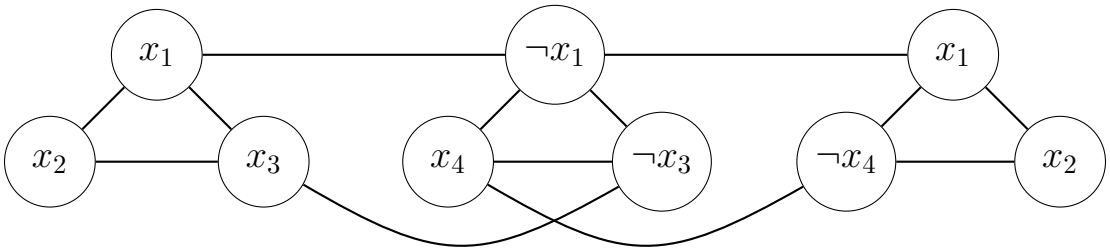
Within this graph, we make two types of edges.

1. Clause edges: connects each vertex to each other within a clause subgraph
2. Contradicting edges: connects vertices x_i and $\neg x_i$

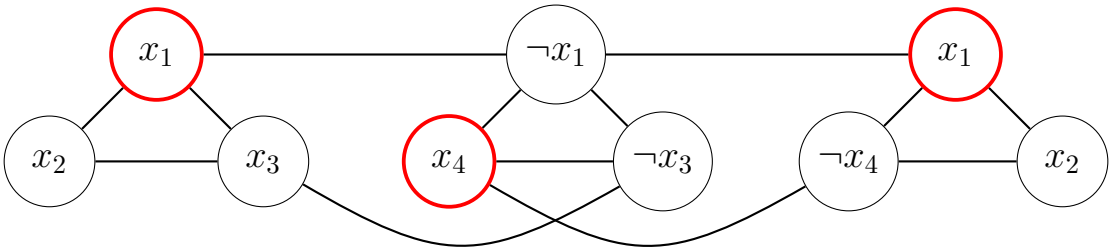
We add *contradicting edges* between vertices that are the same literal but only negated as these values rely on each other. Hence, they both cannot be included in the independent set and would not cause an illegal assignment. Another example:

$$(x_1 \vee x_2 \vee x_3) \qquad (\neg x_1 \vee x_4 \vee \neg x_3) \qquad (x_1 \vee \neg x_4 \vee x_2)$$

This boolean expression corresponds with the following graph:



Hence, on this graph, we can make the following independent set of size m:



Hence, we have the assignment where $x_1 = x_4 = true$ while x_2, x_3 can be any value. In the graph above, $\neg x_1$ and $\neq x_4$ are also assigned values of $false$ as they are the negation of assigned literals. Now, we must prove the reduction.

Proof of reduction. ($\implies$) Suppose F is satisfiable, then we form an independent set by picking one satisfied literal per clause and we claim that none of these vertices have an edge between them (these vertices are independent). Since we pick literals *per* clause, there are no clause edges. Further, we have no contradicting

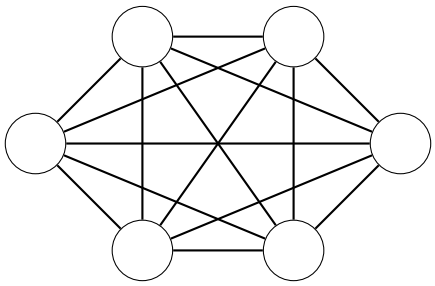
edges because it is not possible that x_i and $\neg x_i$ are both true. Therefore, there is an independent set in G of size m.

($\Longleftarrow$) If there is an independent set of size m, then make a satisfying assignment by satisfying literals corresponding to vertices within the picked vertices. We know this satisfies each clause since there is a single vertex picked per clause (m clauses and m sized independent set). There are no contradictions to this as we prevented this by creating contradicting edges so the independent set would not contain contradicting literals (x_i and $\neg x_i$).

Hence, we have reduced 3-SAT $\rightarrow$ IS and since $IS \in NP$, then the Independent Set problem is NP-Complete. $\square$

4 Clique

A clique in a graph is a set of vertices from V such that all vertices in S are pairwise connected (an edge between every two vertces in S). In other words, $\forall u, v \in S \implies (u, v) \in E$. The subset of vertices forms a complete graph.



Problem Given a graph G and goal g, does G contain a clique of size $\geq g$?

Theorem. *Clique is NP-Complete.*

Proof. First, we check that Clique $\in$ NP (we can verify that a candidate Clique solution is in fact correct). We can verify a candidate solution by looping through $u, v \in C$ that $(u, v) \in E$. Now, we want to find a reduction to Clique.

Let us reduce from Independent Set $\rightarrow$ Clique. Notice that the independent set problem is essentially the opposite of Clique. Hence, if we have some polynomial

time algorithm for Clique, we can try to solve the Independent Set problem.

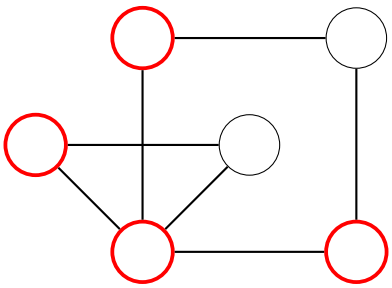
Let us define $\overline{G} = (V, \overline{E})$ where if $(u, v) \in E$ then $(u, v) \notin E$. We claim that S is an independent set in G then $\overline{S}$ is a clique in $\overline{G}$.

Proof. If S is an independent set in G, then in $\overline{S}$ we can pick any two vertices $u, v \in \overline{S}$ we know that $u, v \notin G$. Therefore, $u, v \in \overline{G} \implies S$ is a clique in $\overline{G}$. Hence, we have reduced Independent Set to this problem.

Since we have made a reduction and Clique $\in$ NP, Clique is NP-Complete. □

5 Vertex Cover

A set $S \subseteq V$ is a vertex cover of G if every edge has at least one endpoint in S.



In the above graph, every edge has at least one endpoint that is marked red.

Now, we will turn the generic Vertex Cover into a decision problem. Given a graph G and budget b, does G have a vertex cover of size $\leq b$?

Theorem 5.1. *Vertex Cover is NP-Complete.*

Proof. We can show that Vertex Cover is in NP as we can simply iterate through the edges and check whether or not one of the endpoints is in the candidate solution set. This works in polynomial time. Now, we formulate a reduction. Let us reduce from the Independent Set problem.

We claim that if S is an independent set in G, then $\overline{S}$ is a vertex cover in G. If S is an independent set, we can pick any edge in G and at most only one endpoint is

in S which implies at least one endpoint is in $\overline{S}$, so $\overline{S}$ is a vertex cover.

For the other direction, if $\overline{S}$ is a vertex cover then we can pick two vertices in S which we know cannot share an edge because then that edge has no endpoint in $\overline{S}$ but $\overline{S}$ is a vertex cover.

Hence, Vertex Cover is NP-Complete. $\square$