Question #1: Set-Cover is NP-complete

In the (decision version of) Set-Cover problem you are given n, m, k and m sets $A_1, A_2, \ldots, A_m \subseteq \{1, \ldots, n\}$ and asked "Is there a subset $S \subseteq \{1, \ldots, n\}$ that has at least one element from each A_j and is of size at most k?" I.e. does there exist $S \subseteq \{1, \ldots, n\}$ such that $|S| \leq k$ and $S \cap A_j \neq \emptyset$ for all $1 \leq j \leq m$. Show that this problem is **NP**-complete.

Solution: To show that Set-Cover is **NP**-complete we have to show two things: a) Set-Cover $\in$ **NP**; b) for every $L \in$ **NP**, $L \leq_P$ Set-Cover.
To show that Set-Cover $\in$ **NP** we have to provide an algorithm that takes an input for Set-Cover and a certificate *cert*. The algorithm has to be efficient (run in time polynomial in length of the first input) and if the answer to the Set-Cover input is "Yes" there should exist a value for *cert* that makes the algorithm output "Yes"; otherwise the algorithm should output "No" for every cert. Lemma 1 states that Algorithm 1 seen below has this property.

Algorithm 1: A tester showing Set-Cover $\in$ **NP**.

```
Input: $n$, $m$, $k$, sets $A_1, A_2, \ldots, A_m \subseteq \{1, \ldots, n\}$ and a string cert
Output: "Yes" or "No".
$S \leftarrow$ Interpret cert as a subset of $\{1, \ldots, n\}$.
if $|S| > k$ then
    return "No"
end
for $j \leftarrow 1$ to $m$ do
    if $A_j \cap S = \emptyset$ then
        return "No"
    end
end
return "Yes"
```

Lemma 1. *The answer to the* Set-Cover *input n, m, k and $A_1, \ldots, A_m$ is "Yes" if and only if there exists a cert such that Algorithm 1 returns "Yes" on input n, m, k, $A_1, \ldots, A_m$ and cert.*

Proof. Is left as an exercise. □

To show that for every $L \in$ **NP** $L \leq_P$ Set-Cover (i.e. Set-Cover is **NP**-hard) we only need to show that $L' \leq_P$ Set-Cover for some specific **NP**-hard problem L' of our own choosing. That way we will have,

$$L' \leq_P \text{Set-Cover} \wedge \forall L \in \mathbf{NP}\ L \leq_P L' \Rightarrow \forall L \in \mathbf{NP}\ L \leq_P \text{Set-Cover}.$$

In class we saw that 3-SAT is **NP**-hard (in fact we showed that 3-SAT is **NP**-complete) so it is enough to show that 3-SAT $\leq_P$ Set-Cover.
Remember that in the 3-SAT we are given n', m', the name of n' variables $x_1, \ldots, x_{n'}$ and m' clauses $C_1, \ldots, C_{m'}$ and are asked if there is an assignment of True/False values to the variables that makes all the clauses "satisfied". Each clause is an OR of three "literals" and is said to be satisfied if at least one of the literals are true. A literal is either a variable or the negation of a variable. For example if $n' = 5$, $x_1, x_2, \ldots, x_5, \neg x_1, \neg x_2, \ldots, \neg x_5$ are the possible literals and $x_1 \vee \neg x_2 \vee \neg x_5$, $x_2 \vee x_4 \vee \neg x_5$ and

$\neg x_1 \vee \neg x_3 \vee \neg x_5$ are some of the possible clauses[1]; the assignment $x_1 = False$, $x_2 = True$, …, $x_5 = True$ does not satisfy the first clause but satisfies the other two and the assignment $x_1 = False$, …, $x_5 = False$ satisfies all the clauses. So if the input is $n' = 5$, $m' = 3$, $x_1, \ldots, x_5$ and $x_1 \vee \neg x_2 \vee \neg x_5$, $x_2 \vee x_4 \vee \neg x_5$ and $\neg x_1 \vee \neg x_3 \vee \neg x_5$ the correct output is "Yes".

To reduce 3-SAT to Set-Cover we have to take an input to 3-SAT and transform it into an input for Set-Cover. Given that the choices one has to make in coming up with a solution to the 3-SAT input is whether each variable is set to true or false and the choices one needs to make for a solution to the Set-Cover input is whether each element of $\{1, \ldots, n\}$ is selected to be in S or not (at first) it seems natural to have one element in $\{1, \ldots, n\}$ corresponding to each of the variables of the 3-SAT input. After playing with this idea for a bit it become clear that it does not work. The reason is not very hard: Choosing an element of $\{1, \ldots, n\}$ to be in S only helps a solution but assigning a variable to be True or False has both positive and negative consequences for a solution: Assigning x_i to be True for example helps all the clauses that have x_i as a literal but hurts all those that have $\neg x_i$ as a literal.

The second try would then be as follows: We have an element corresponding to each *literal* in $\{1, \ldots, n\}$, i.e. we let $n = 2n'$ and for each variable x_i we will name an element of $\{1, \ldots, n\}$ as x_i and another as $\overline{x_i}$. The goal is to force the Set-Cover solution to take *exactly one* of these two elements to be in S while satisfying all the clauses. This turns out to be not too hard. To force any solution to Set-Cover to choose exactly one of these two elements we will have a set $A_i = \{x_i, \overline{x_i}\}$ among the sets in the Set-Cover input, we will also set $k = n' = n/2$. This way any solution to the Set-Cover has to take at least one of $x_i, \overline{x_i}$ because of A_i and it has to take at most one because it can not take more than $k = n'$ elements overall. Given what we have so far it is easy to make sure that the Set-Cover solutions also "satisfy" the clauses of the original 3-SAT input. For every clause of the original 3-SAT input we will add a set A_j that has all the literals in the clause. This way at least one of the literals has to be selected. The end result is a reduction that is shown as Algorithm 2 below. Lemma 2 states that Algorithm 2 is a correct reduction from 3-SAT to Set-Cover.

Algorithm 2: A reduction from 3-SAT to Set-Cover.

```
Input: $n'$, $m'$, variables $x_1, \ldots, x_{n'}$ and $m'$ clauses $C_1, \ldots, C_{m'}$
Output: $n$, $m$, $k$, sets $A_1, A_2, \ldots, A_m \subseteq \{1, \ldots, n\}$
1  $n \leftarrow 2n'$
2  $k \leftarrow n'$
3  $m \leftarrow n' + m'$
4  Give the following names to the elements of $\{1, \ldots, n\}$. $x_1, \overline{x_1}$, $x_2, \overline{x_2}$, …, $x_{n'}, \overline{x_{n'}}$
5  for $i \leftarrow 1$ to $n'$ do
6      $A_i \leftarrow \{x_i, \overline{x_i}\}$
7  end
8  for $j \leftarrow 1$ to $m'$ do
9      $l, l', l'' \leftarrow$ the literals in $C_j$
10     $A_{n+j} \leftarrow \{l, l', l''\}$
11 end
12 return $n$, $m$, $k$, sets $A_1, A_2, \ldots, A_m$
```

Lemma 2. *For every value input of* 3-SAT *Algorithm 2 produces a valid input of* Set-Cover *such that their answer is exactly the same.*

[1] $\neg$ is the symbol for NOT and $\vee$ is the symbol for OR

Proof. First assume that the answer to the 3-SAT input is "Yes". Then there exists an assignment $x_1 = a_1, \ldots, x_n = a_n$ where a_i's are True/False values that satisfies all the clauses. We have to show that the answer to the produced Set-Cover instance is also "Yes". Consider the set S that for each i contains x_i if $a_i = True$ and $\overline{x_i}$ if $a_i = False$, i.e. the set that corresponds to all satisfied literals. Clearly $|S| = n' \leq k$ and for all $1 \leq i \leq n'$, $S \cap A_i \neq \emptyset$. We need to show that S takes at least one element of $A_{n'+1}, \ldots, A_m$. Consider $A_{n'+j}$; this set corresponds to the clause C_j of the 3-SAT input and contains all its literals. Given $x_1 = a_1, \ldots, x_n = a_n$ satisfies the clause C_j it must set one of its literals to true and by definition (of S) that literal is in S so $S \cap A_{n'+j} \neq \emptyset$. This completes the proof that if the answer to the 3-SAT input is "Yes" then the answer to the Set-Cover input is "Yes".
We now show that if the answer to the 3-SAT input is "No" then the answer to the Set-Cover input is also "No". Assume that the answer to the Set-Cover input is not "No", i.e. it is "Yes", we will show that this implies that the answer to the 3-SAT input is also "Yes". If the answer to the Set-Cover input is "Yes" then there exists a set S of size at most k that intersects $A_1, \ldots, A_m$. Given that S intersects $A_1, \ldots, A_{n'}$ it has to take at least one of $x_i, \overline{x_i}$. This together with the fact that S takes at most $k = n'$ elements implies that $|S| = n'$ and that S takes precisely one of $x_i, \overline{x_i}$ for every i. Consider the assignment $x_1 = a_1, \ldots, x_n = a_n$ where a_i's are True/False values defined as

$$a_i = \begin{cases} True & \text{if } x_i \in S \\ False & \text{if } \overline{x_i} \in S \end{cases}.$$

We will show that this assignment satisfies all the clauses of the 3-SAT input so the answer to the 3-SAT input must be "Yes". Consider a clause C_j: S is a valid answer to the Set-Cover input so $S \cap A_{n'+j} \neq \emptyset$. But $A_{n'+j}$ is the set of literal of C_j so it follows from the definition of the assignment $x_1 = a_1, \ldots, x_n = a_n$ that it sets at least one of the literal of C_j to true hence satisfying C_j. This completes the proof that if the answer to the 3-SAT input is "No" then the answer to the Set-Cover input is also "No". □

Finally putting everything together we have the following Theorem.

Theorem 3. *The (Yes/No) problem* Set-Cover *is* ***NP****-complete.*

Proof. It follows from Lemma 1 that Set-Cover $\in$ **NP**. On the other hand Lemma 2 implies that 3-SAT $\leq_P$ Set-Cover which together with the theorem from the lecture that 3-SAT is **NP**-hard implies that Set-Cover is **NP**-hard. These two taken together mean that Set-Cover is **NP**-complete. □