Richmond Osei, Wasif Khan, and Pooja Doshi

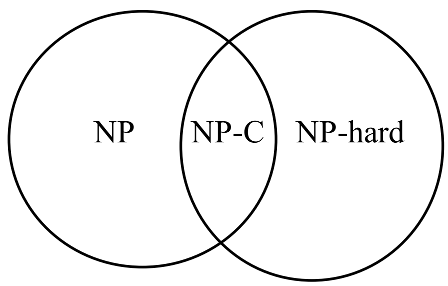
Assignment 1

1. SET COVER: A set covering instance is (X, F) for a finite set X and $F \subseteq 2^X$ such that $X = \cup_{S \in F} S$. We say that S covers its elements. Given a set covering instance (X, F) and an integer k, is there a subset $C \subseteq F$, $|C| = k$, whose members cover X.

Solution

To show a given problem is problem is NP- complete, It has to belong to the class of NP and also belong to the class of NP-hard, that is at least hard as any problem in NP. So if I have a problem X. X is NP- complete if :

$$X \in NPC \Leftrightarrow L \subseteq NP \text{ and } L \subseteq NP\text{ - hard.}$$



a) Proving our problem X belongs to NP problems. That is our problem X is decision problems with a polynomial size certificate and a polynomial time verifiers for Yes instances of X.

b) To show problem X is in NP-hard, that is at least as hard as any other problem in NP. We formulate a related decision problem (reduction) to prove X is NP-Complete. That is taking a proved NP- complete problem and reducing it to our problem X. So having a problem Y which is known to be NP-complete, we can reduce it our problem X. Also, with NP-complete definition, if our problem X is NP-Complete , any problem in NP complete can be reduce to prove X is NP complete.

c) We produce an efficient algorithm to solve for the Y known problem, that is showing an instance of Y can be solved in polynomial number of operations, and a polynomial number of calls to a black box that can solve our problem X.
(Kleinberg, 2008)

A proof that the SET COVER belongs to the class NP:

To prove that SET COVER belongs the class NP. Algorithm that take instances for a SET COVER and a certificate. The algorithm must take polynomial time to run in length and if the answer to the Set-Cover input is "Yes" there should exist a value for cert that makes the algorithm output "Yes".

Algorithm 1: algorithm to show SET-COVER $\in$ NP

Input: n, m, k, set of $C_1, C_2, \ldots C_n \subseteq \{1, \ldots n\}$ and a string certificate.

Output: "Yes" or "No"

- $C \leftarrow$ interpret cert as a subset of $\{1, \ldots, n\}$
- If $|C| > k$ then
- | return "No"
- end
- for $j \leftarrow 1$ to n do
- | if $C_j \cap F = \emptyset$ then
- | | return "No"
- | end
- end
- return "Yes"

Lemma 1: The answer to the SET COVER input n, m, k and $C_1, \ldots, C_n$ is "Yes" if and only if there exists a cert such that Algorithm 1 returns "Yes" on input n, m, k, $C_1, \ldots, C_n$ and cert. (Benabbas, 2012, p. 1)

Reduction of 3 SAT(a known NP-hard problem) to SET COVER

To prove that for every $X \in NP$ $X \leq_p$ SET-COVER (that is SET-COVER is NP-hard) we only need to prove that $X' \leq_p$ SET-COVER for some specific NP-hard problem X' of our own choosing. Formula:

$$X' \leq_P \text{Set-Cover} \wedge \forall X \in NP \quad X \leq_P X' \Rightarrow \forall X \in NP \quad X \leq_P \text{Set-Cover}$$

In reduction of 3-SAT to SET-COVER, instance of 3- SAT is transform it into an instance for SET-COVER. Noticing that the choices taking in making up with a solution to the 3-SAT input is whether each variable is set to true or false and the choices to make for a solution to the SET-COVER input is whether each element of {1,….,n} is selected to be in C.

Each literal in {1, . . . , n}, i.e. we let $n = 2n'$ and for each variable x_i we will name an element of {1, . . . , n} as x_i and another as $-x_i$. The goal is to manipulate the Set-Cover solution to take exactly one of these two elements to be in S while satisfying all the clauses. In doing this, any solution to Set-Cover to select exactly one of these two elements we will have a set

$$Ai = \{xi, -xi\}$$

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 , $-x_i$ because of A_i and it has to take at most one because it cannot take more than $k = n^1$ 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.
(Benabbas, 2012, p. 2)

<u>Algorithm 2: A reduction from 3-SAT to Set-Cover</u>
Input: n', m', variables $x_1, \ldots, x_{n'}$ and m' clauses $C_1, \ldots, Cm'$
Output: n, m, k, sets $A1, A2, \ldots, Am \subseteq \{1, \ldots, n\}$

- $n \leftarrow 2n'$
- $k \leftarrow n'$
- $m \leftarrow n'$
- Give the following names to the elements of $\{1,\ldots,n\}.x_1,-x_1, x_2,-x_2,\ldots x_{n'}, -x_{n'}$
- <u>for</u> $i \leftarrow 1$ <u>to</u> n' <u>do</u>
- | $A_i \leftarrow \{x_i, -x_i\}$
- <u>end</u>
- <u>for</u> $j \leftarrow 1$ <u>to</u> m' <u>do</u>
- | $l, l', l'' \leftarrow$ the literals in C_j
- | $A_{n+j} \leftarrow \{l, l', l''\}$
- end
- <u>return</u> 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.
(Benabbas, 2012)

We then 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 x_i if a_i = False, i.e. the set that corresponds to all satisfied literals. Clearly $|S| = n$ $0 \leq k$ and for all $1 \leq i \leq n^1$, $S \cap A_i = \emptyset$. We need to show that S takes at least one element of $A_{n^1+1}, \ldots, A_m$.

Consider A_{n^1+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 $S \cap A_{n^1+j} = \emptyset$. 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 $A_1, \ldots, A_{n^1}$ it has to take at least one of x_i , $-x_i$. This together with the fact that S takes at most $k = n^1$ elements implies that $|S| = n^1$ and that S takes precisely one of x_i , $-x_i$ 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 = (True if $x_i \in S$
False if $-x_i \in S$

We then prove 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 $S \cap A_{n^1+j} = \emptyset$. But A_{n^1+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 .
(Benabbas, 2012)

Conclusion

From lemma 1 that Set-Cover ∈ NP. On the other hand Lemma 2 implies that 3-SAT $\leq_P$ Set-Cover which together with the theorem shows that 3-SAT is NP-hard implies that Set-Cover is NP-hard. These two proof shows that Set-Cover is NP-complete.

Author(s): Richmond Osei, Pooja Doshi, Wasif Jami Khan

Reference

Benabbas, S. (2012, August 7). University of Toronto. Retrieved from https://web.cs.toronto.edu/: http://www.cs.toronto.edu/~siavosh/csc373h/files/TN8.pdf

Kleinberg, P. B. (2008, Spring). Introduction to Analysis of Algorithms. Retrieved from http://www.cs.cornell.edu/courses/cs482/2007su/NPComplete.pdf