COT 6405 Analysis of Algorithms
Homework 5

Michael Novo

December 2, 2015

Note: Discussed with Nestor Hernandez, Gregory Reis, Georges Kamhoua

1. 2-CNF $\in \mathbf{P}$

   We prove that 2-CNF is $\in \mathbf{P}$ by showing a poly-time reduction from 2-CNF to PATH- SEARCH. PATH- SEARCH is the decision problem where the input is $< G, a, b >$ and we are trying to determine if there is a pth from a to b in G. We know that PATH- SEARCH $\in \mathbf{P}$ and we would like to use it to show that 2-CNF $\in \mathbf{P}$.

   We first construct a graph, $G = (V, E)$ with $2n$ vertices where n represents the number of variables in the 2-CNF formula Φ. Then, for each clause of the form $(p \vee q) \in \Phi$, we create a directed edge from $\bar{p}$ to q and from $\bar{q}$ to p. Doing so signifies that if p is false, then q must be true. Therefore, there exists a directed edge $(p, q) \in G$ if there exists a clause $(\bar{p} \vee q)$.

   We then see that if G contains a path from p to q, then it also contains a path from $\bar{q}$ to $\bar{p}$. If the path from p to q be $p \rightsquigarrow p_1 \rightsquigarrow p_2 \cdots \rightsquigarrow p_k \rightsquigarrow q$, then by construction of G, if there is an edge (p_i, p_j), then there is also an edge$(\bar{p_j}, \bar{p_i})$. Therefore, we have the edges $(\bar{q}, \bar{p_k}), (\bar{p_k}, \bar{p_{k-1}}), \ldots, (\bar{p_2}, \bar{p_1}), (\bar{p_1}, \bar{p})$. This means that we have a path from $\bar{q}$ to $\bar{p}$.

   The next step is to prove that 2-CNF is unsatisfiable iff there exists a variable p_i such that: there is a path from p_i to $\bar{p_i}$ and also a path from $\bar{p_i}$ to p_i in G. For the sake of contradiction, suppose that we have both paths $p_i \rightsquigarrow \bar{p_i}$ and $\bar{p_i} \rightsquigarrow p_i$ for some variable $p_i \in G$. Also, suppose that there also exists a satisfying assignment for Φ. Let the path $p_i \rightsquigarrow \bar{p_i}$ be $p_i \rightsquigarrow ... \rightsquigarrow a \rightsquigarrow b \rightsquigarrow ... \rightsquigarrow \bar{p_i}$. By construction, there is an edge $(\alpha, \beta) \in G$ if and only if there is a clause $(\bar{\alpha} \vee \beta) \in \Phi$. In particular, the edge (α, β) means that if α is true then β must be true.

   Suppose p_i is true. Then, all literals in the path from p_i to a must be true. Similarly, the literals in the path from b to $\bar{p_i}$ must be false since $\bar{x}$ is false. This implies that we have an edge (a, b) with a set to true and b set to false. As a result, the clause $(\bar{a} \vee b)$ becomes false which contradicts the assumption that there exists a satisfying assignment for the formula Φ. This reasoning holds when p_i is false.

   Hence, the problem reduces to checking the existence of the paths in the graph G and deciding whether or not the related 2-CNF formula is satisfiable. The graph search problem can be solved by algorithms like BFS and DFS which take polynomial time $O(|V| + |E|)$. Thus, 2-CNF can be solved in polynomial time.

   Reference: Slides presented by Dana Moshkovitz at a colloquium on this topic at the University of Maryland were used in the construction of this solution.

2. BALANCED-SAT is NP-complete

   We first show that we can construct a polynomial time verifier such that when given a 3-CNF formula and an assignment, we can check if the assignment satisfies exactly half the clauses. The input into the algorithm is a boolean formula in CNF and a certificate representing boolean variables and their corresponding truth assignments. The algorithm then iterates through the formula checking if exactly half of the clauses are true. If this is the case, true is returned. Otherwise, false is returned. Because the boolean formula is checked only once, the verifier runs in polynomial time.

   Next, we perform a reduction from 3-SAT to BALANCED-SAT. In order to do so, we create a formula Φ' that works as follows: first the original m clauses in Φ are considered, then create

m more always true clauses with the form $(x \vee \bar{x} \vee y)$, lastly create $2m$ additional always false clauses of the form $(x \vee y \vee z)$

The new formula Φ' contains $4m$ clauses, where m are always true and $2m$ are either all true or all false. This conversion takes polynomial time because the addition of $3m$ new clauses and 3 new variables is polynomial.

Next, we prove that Φ is in 3-SAT if and only if Φ' is in BALANCED-SAT. Suppose there is an assignment such that Φ is satisfiable. Then, the m clauses which correspond to Φ in Φ' are true and the m clauses that are always true. Then, by setting x, y and z to false, we obtain a truth assignment satisfying BALANCED-SAT because the last $2m$ clauses will be false.

Lastly, assume there exists a truth assignment such that half the clauses in Φ' are true while the other half of the clauses are false. We know that m clauses are always true in Φ', which means that the last $2m$ clauses cannot be true if BALANCED-SAT is satisfied (because we would have $3m$ clauses set to true which is greater than $2m$). Therefore, the last $2m$ clauses must be false, which means that Φ is true proving that 3-SAT is satisfiable. Hence, we have shown BALANCED-SAT to be NP-hard and NP-complete.

Reference: Nina Amenta's slides from UC-Davis were used to construct this solution.

3. UNIQUE-COVER is NP-complete

Proving that UNIQUE-COVER is NP-Complete involves two steps. The first is to show that a polynomial time verifier can be built for UNIQUE-COVER. The second step is to show a reduction from 3-Coloring (seen in class) to UNIQUE-COVER.

Let's first show that we can design a polynomial time verifier, V, for UNIQUE-COVER. We design V as follows. V takes in input of the form $\langle X, S, S' \rangle$ and verifies that each set of S is a subset of X. Then, the verifier checks that each set in S' is an element of S. Verify that each element in X can be found in a set in S'. Lastly, verify that each element of X belongs to at most one element in S'. This shows that UNIQUE-COVER $\in$ NP.

Next, we show that a 3-Coloring problem can be made into a corresponding UNIQUE-COVER problem. For each node u in the 3-Coloring graph, G, we construct four elements that are found in X. Include the original element, u, itself, then include R_u, G_u and G_u. Then, for each edge $\{u, v\} \in G$, construct three elements in X, R_{uv}, G_{uv} and G_{uv}. The size of X will then be $\mathrm{O}(|V| + |E|)$.

Then, to replicate the family of subsets S, construct each set that is in S by including the going through each node u in G and constructing a set with the element u and each B_{uv} for each edge $\{u, v\}$ found in the graph. Then, repeat the procedure for the remaining colors G and R. Then, add singleton sets containing individual elements except those that are nodes. The size of X will then be $\mathrm{O}(|V| + |E|)$.

By this construction, we now show that we have a "yes" instance of a UNIQUE-COVER problem if and only if we have a "yes" instance of the 3-COLORING problem. Suppose that we have a graph G that has a 3-Coloring. Then, we choose $S' \subseteq S$ as follows. For each vertex u in G, include in S' the set containing u itself along with the color that was picked for u along with the edges that originate from it and are incident upon another vertex. Also, include in the S' the singleton sets that contain individual elements and have not been chosen in the sets mentioned previously. That is, choose G_v to be in S' if v was chosen as a color that is not

green and is not connected to any vertices that are green. Then, for each edge that is in G, chose the sets that contain the vertex itself along with the color that was chosen for it along with the vertices that is connected to. For example, vertex v is green and is connected to y, choose vertex v and G_{vy} to be in S'. Similarly, include the singleton sets of this form like the singleton sets were chosen for vertices. This corresponds to choosing singleton sets of edges of colors that the its adjoining vertices are not colored by. This covers all the points that are in X, and each point is covered exactly one in S' .

Now, suppose S' has a UNIQUE-COVER. Then, each vertex u in G is covered by exactly one set in S' and must be of the form $\{u, R_u, R_v\}$ for the color that u is colored and the vertices that u is connected to by an edge. We claim that using this procedure will lead to a valid 3-coloring. If a vertex u has been colored by a color c, we include the set as mentioned above, and therefore cannot include the sets that include u but are associated with some other color that is not c, since u is already covered. Therefore, the other sets that contain u with some other color will be chosen from the remaining sets that have not been chosen. Choosing sets in this fashion will cover all sets except those elements that correspond to edges not affected by a color since the edge only connects two vertices. We can choose these sets from the singleton sets, which by the construction of S correspond to edges that are between vertices with different colors.

Reference: Slides from a CS 103 lecture from Stanford were used to construct this solution along with some ideas from a University of Maryland lecture by William Gasarch.

4. INTEGER LINEAR PROGRAMMING is NP-complete

In order to prove the ILP is NP-Complete, we first note that ILP is in NP because if we are given a certificate we can check in polynomial time if it is solution to ILP. Next, we reduce 3-SAT to ILP.

A reduction of 3-SAT to ILP will show that ILP is NP-hard. Let $\Phi \in$ 3-SAT with variables $p_1, p_2, \ldots, p_n$. We then create the corresponding variables to the integer linear programming problem $x_1, x_2, \ldots, x_n$ such that $x_i \in \{0, 1\}$. The assignment of $x_i = 1$ in the ILP problem corresponds to setting p_i to true in Φ.

For each clause of the form $(p_1 \vee \bar{p_2} \vee p_3)$, we convert to the ILP problem as follows:

$$x_1 + (1 - x_2) + x_3 > 0$$

Satisfying this inequality corresponds to setting $x_1 = 1$ or $x_2 = 0$ or $x_3 = 1$, which translates to p_1 being true p_2 being false or p_3 being true in the truth assignment. This assignment can be done in polynomial time since the algorithm involves just creating as inequalities from clauses found in Φ. We have reduced 3-SAT to ILP in polynomial time and ILP is in NP, we may now conclude that ILP is NP-complete.

5. EQUAL-PARTITION is NP-complete

In order to prove that EQUAL-PARTITION is NP-complete we first show that EQUAL-PARTITION is in NP. This is true since if we are given a certificate that is a subset S' and the complement $S - S'$, we may check in polynomial time if the sum of weights in S' is equal to the sum of weights of the elements in $S - S'$.

Next, we reduce from SUBSET-SUM to EQUAL-PARTITION. Suppose we have an instance of the EQUAL-PARTITION problem $S_{EP} = \{e'_1, e'_2, \ldots, e'_n\}$ and we know $\sum_{x \in S'_{EP}} w(x) = \sum_{x \in S - S'_{EP}} w(x)$. We can then say that there is a correspondence between e_k and e'_k. We claim that SUBSET-SUM has a solution $S'_S S$ if and only if EQUAL-PARTITION has a solution S'_{EP} and $S - S'_{EP}$. The corresponding value of t is:

$$t = \frac{\sum_{i=1}^{n} w(e'_i)}{2}$$

because there does not exist another way to sum a subset and its complement to the same values unless the value is half the sum of all the values in the original set.

We can then see that the reduction of SUBSET-SUM to EQUAL-PARTITION can be done in polynomial time be constructing a one-to-one mapping of $\{e_1, e_2, \ldots, e_n\}$ to $\{e'_1, e'_2, \ldots, e'_n\}$ and allowing $t = \frac{e'_1 + e'_2 + \cdots + e'_n}{2}$. Thus, the EQUAL-PARTITION problem is NP-complete.