CSCI 570 - Analysis of Algorithms

Assignment 6

Harshitha Kurra

---

1. **In Linear Programming, variables are allowed to be real numbers. Consider that you are restricting variables to be only integers, keeping everything else the same. This is called Integer Programming. Integer Programming is nothing but a Linear Programming with the added constraint that variables be integers. Prove that integer programming is NP-Hard by reduction from SAT.**

Solution: We are considering Integer Programming as IP and we need to prove this is NP-Hard by reduction from SAT.
Consider SAT has n variables and m clauses. The constraints can be defined as follows:
Constraints for IP:

- $x_i \in [0,1]$ where x_i is the variable
- $-x_i \in [0,1]$
- $x_i + (-x_i) = 1$
- All the clauses like C = $\{x_1, x_2, x_3, \neg x_3, \dots . x_i\}$ we say $x_1 + x_2 + x_3 + \neg x_3, \dots . + x_i \geq 1$

Claim: we claim that SAT has a satisfiable assignment if and only if it has a solution in Integer Programming (IP).

Proof:

I. SAT assignment is satisfiable:
Here, TRUE => 1 and FALSE => 0. Since, SAT is satisfiable we say at least one variable in each clause is TRUE. So our constraint: All the clauses C = $\{x_1, x_2, x_3, \neg x_3, \dots . x_i\}$ we say $x_1 + x_2 + x_3 + \neg x_3, \dots . + x_i \geq 1$ is valid.
So, IP is having a solution.

II. IP has a solution:
In IP, we will be having 0 or 1. Here, 0 implies FALSE and 1 implies TRUE. Since, a variable and its complement can never have the same value, we will say SAT is satisfiable.

Finally, We can say that $SAT \leq_p IP$.
Therefore, Integer Programming is NP-Hard by reduction from SAT.

---

2. **We know that the SAT problem is NP-complete. Consider another variant of the SAT problem: given a CNF formula F, does there exist a satisfying assignment in which exactly half the variables are true? Let us call this problem HALF-SAT. Prove that HALF-SAT is NP-complete.**

Solution: Inorder to prove a problem X is NP-Complete, we need to prove the Problem X ∈ NP and we need to take a problem Y which is NP-Complete and reduce it to X(NP-Hard).

$$Y \leq_p X$$

HALF-SAT ∈ NP:

We can prove this by assigning values to the variables and checking if each clause is satisfiable after substitution. And then checking if half of those values are TRUE.
So, HALF_SAT belongs to NP. ——————————> 1.

HALF-SAT ∈ NP-HARD:

We will be reducing the problem from SAT.

SAT consists of variables $x_1, x_2, x_3, \ldots . x_n$ and let us define this by the function:

$f(x_1, x_2, x_3, x_4, \ldots x_n)$

For HALF-SAT, we introduce n new variables $y_1, y_2, y_3, \ldots y_n$ where $y_i = - x_i$ and let us represent this function as $g(x_1, x_2, x_3, x_4, \ldots x_n, y_1, y_2, y_3, \ldots y_n)$

$$g(x_1, x_2, x_3, x_4, \ldots x_n, y_1, y_2, y_3, \ldots y_n) = f(x_1, x_2, x_3, x_4, \ldots x_n) \wedge f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$$

Claim: SAT has satisfiable assignment if and only if HALF-SAT has a satisfiable assignment.

Proof:

I. SAT has a satisfiable assignment:

We are given SAT is satisfiable. So, $f(x_1, x_2, x_3, x_4, \ldots x_n)$ is TRUE. For HALF-SAT we are assigning n more variables as follows:

$$g(x_1, x_2, x_3, x_4, \ldots x_n, y_1, y_2, y_3, \ldots y_n) = f(x_1, x_2, x_3, x_4, \ldots x_n) \wedge f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$$
$$= \text{TRUE} \wedge f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$$

Since we know that $y_i = \neg x_i$, implies $\neg y_i = \neg(\neg(x_i)) = x_i$

$f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$ will always be true. If SAT has k variables which are TRUE, for each $y_i = \neg(x_i)$, we will have k variables in $f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$ which will be TRUE.

Therefore, $g(x_1, x_2, x_3, x_4, \ldots x_n, y_1, y_2, y_3, \ldots y_n)$ will have n variables (k + (n-k) = n variables) which are TRUE out of 2n variables.
So, HALF-SAT has a valid assignment and is satisfiable.

II. HALF-SAT has a satisfiable assignment:
We know that

$g(x_1, x_2, x_3, x_4, \ldots x_n, y_1, y_2, y_3, \ldots y_n) = f(x_1, x_2, x_3, x_4, \ldots x_n) \wedge f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$

If we say $g(x_1, x_2, x_3, x_4, \ldots x_n, y_1, y_2, y_3, \ldots y_n)$ is TRUE then $f(x_1, x_2, x_3, x_4, \ldots x_n)$ is TRUE and $f(\neg y_1, \neg y_2, \neg y_3, \ldots \neg y_n)$ is TRUE. So, SAT has a valid assignment and is satisfiable.

$$SAT \leq_p HALF-SAT$$

Implies, HALF-SAT is NP-Hard ———————> 2.

Therefore, from 1. and 2., HALF-SAT is NP-complete.

---

3. There is a set of courses, , each of them is represented by a set of disjoint time intervals with the starting and finishing times. For example, a course could require the time from 9am to 11am and 2pm to 3pm and 4pm to 5pm. You want to know, given a number K, if it's possible to take at least K courses. You cannot choose any two overlapping courses. Prove that the problem is NP- complete, which means that choosing courses is indeed a difficult thing in our life. Use a reduction from the Independent set problem.

Solution: We need to prove taking K courses is NP-Complete. Implies, we need to prove that it ∈ NP and ∈ NP-Hard.

Assigning K course ∈ NP:
We need to show that the problem can be verified in polynomial time. For this we need to take set of courses and we need to check if they have overlapping intervals. Let us assume there are n courses and t time intervals. verifying this solution is polynomial. Since, we are able to assign the courses in polynomial time, we say that the given problem ∈ NP.

Assigning K courses ∈ NP-Hard:
We will be reducing the problem from the Independent set problem since we know that independent set(IS) problem is NP-Hard, if $IS \leq_p K-COURSE-ASSIGNMENT$
For this, we consider a graph G which is an instance of independent set where the vertices are courses and an edge between them implies that they have an overlapping time interval.

Claim: An Independent set of size k exists if and only if there are k courses with no overlapping time intervals.

Proof:

I. Independent set of k exists:
In an independent set there are no edges between two vertices (courses). No edges implies no overlapping courses with the same time interval.
So, there are k courses with no overlapping time intervals.

II. There are k courses with no overlapping time:
This implies that there are k nodes which do not share any edges between them. So, these k nodes can be a part of independent set.
So, there exists independent set of size k.

Therefore, Assigning k courses with no overlapping time interval is NP-Hard.
=> $IS \leq_p K-COURSE-ASSIGNMENT$
=> The given problem is NP-Complete.

---

4. It is well known that planar graphs are 4-colorable. However finding a vertex cover on planar graphs is NP-hard. Design an approximation algorithm to solve the vertex cover problem on planar graph. Prove your algorithm approximation ratio.

Solution:
Assumptions: the vertex cover is at least of size $|V|/2$.

Algorithm: The given problem can be solved using Greedy Algorithm.
- Take a coloring function which assigns color to the vertices where adjacent vertices are not given same color.
- Pass the Graph to the above defined function so that we get 4-colored graph.
- Let these 4 color classes be C_1, C_2, C_3, C_4
- Ignore the color class with the largest number of vertices, get all the vertices from the remaining color classes.
- Add these to Vertex Cover.

In the above defined Algorithm, we add almost $(3/4)^{th}$ of the vertices.
Our algorithm is ALG and the optimal solution is OPT.
So, the approximation algorithm will be as follows:

$$ALG \leq 1.5OPT$$

Proof for the approximation ratio:
According to our assumption: $OPT \geq |V|/2$ and $ALG \leq 3|V|/4$
So, Approximation ratio is ALG/OPT
= $3|V|/4$ / $|V|/2$
$\leq$ 1.5
$\implies ALG \leq 1.5OPT$
So, our algorithm is a 1.5 approximation at the worse.

---

5. **Consider the following heuristic to compute a minimum vertex cover of a connected graph G. Pick an arbitrary vertex as the root and perform depth first search. Output the set of non-leaf vertices in the resulting depth first search tree.**
   A. **Show that the output is indeed a vertex cover for G.**
   B. **How good is this approximation? That is, upper bound the ratio of the number of vertices in the output to the number of vertices in a minimum vertex.**

Solution: We are considering a graph with V vertices. According to the heuristic mentioned in the question, lets say the Tree we get when we run DFS on the graph G is M.
The degree of the non-leaf nodes is > 1 and the degree of the leaf nodes is = 1.

A. We will be proving using Proof by contradiction:

Let us consider there is an edge $e = (u, v)$ and let's say both u,v are not non-leaf nodes. So, they are leaf nodes. Let's say out DFS explore u first. Then there will be a path for DFS to go from u to v. Which means u is not a leaf node. Therefore, our assumption is wrong. The non-leaf nodes should definitely cover the edge $e = (u, v)$. The set of non-leaf vertices does cover every edge of *G,* making it a vertex cover for *G*.

B. Approximation Ratio:
From the above defined algorithm, we say that the vertex cover is of the size of number of non-leaf nodes (————-> Fact 1). Let us consider our Tree M. Let M_{ODD} be the set which contains all the non-leaf nodes at odd level in the tree and M_{EVEN} be the set which contains all the non-leaf nodes at the even level in the tree.
Consider a variable x
if $M_{ODD} > M_{EVEN}$:
    $x = M_{ODD}$
else:
    $x = M_{EVEN}$
Therefore, if the number of non-leaf nodes is N then x will be of size $|N|/2$.
$K = \{alltheedges \in x\}$ where K is the edge matching in Graph G. Here
$u \in Nonleafnodes$
Now, inorder to cover all the edges in K, the optimal vertex cover should be at least $|K|$ vertices.
$\implies OPT \geq |K|$
$\implies OPT \geq |N|/2$ where $|N|$ is the number of non-leaf nodes.
We know that $ALG = |N|$ from Fact 1.

So, the above given algorithm is 2-approximation algorithm at worst case.