Homework - 6

Shreenidhi Hegde

Problem1:

Proof by reduction from Satisfiability:

We reduce each clause in SAT instance to inequality constraint in Integer LP as the following

- If a literal in the clause is in the normal form xi , then we add zi to the inequality constraint
- If a literal in the clause has a negation form ¬ xi , then we add (1 – zi) to the inequality constraint.

For example, consider the clause (x1 ∨ ¬ x2 ∨ x3) , it is reduced as z1 + (1-z2) + z3 ≥ 1
So, for each literal xi in SAT, we write variable zi in Integer LP, where zi takes the values {0, 1} which is equal to corresponding Boolean literal values in SAT {FALSE, TRUE}

This reduction can be clearly achieved in polynomial time

Claim => If SAT is satisfiable, then Integer LP has an integer solution:
If each clause in SAT is true, then at least one literal in each clause must have a truth assignment. This will make it add 1, due to the corresponding variable in the corresponding inequality has value assigned as 1. Thus, the inequality constraint will be satisfied. All clauses need to be True for SAT to be satisfiable, this mean all our inequalities will also be satisfied based on the above.

Conversely, Claim <= If Integer LP has a solution, then SAT is satisfiable:
If all inequality constraints in the Integer LP are satisfied (ILP has solution), then at least one variable in each inequality constraint will make a corresponding literal in the SAT instance clause to be True. Thus, making all the clauses in the SAT to have a truth assignment. Which in turn make the SAT satisfiable.

From the claims, Satisfiablity $\leq p$ ILP . We know that SAT is NP Complete. Hence ,we have ILP ∈ NP-Hard.

Problem 2:

Given F | F is a CNF formula

Prove HALF-SAT ∈ NP

Given the values/assignment of the half literals to be true in the HALF-SAT problem, it can be easily verified in polynomial time, whether the HALF-SAT is satisfiable.

Just substitute the truth assignment values to the literals by traversing the CNF formula, and verify the satisfiability. Thus, HALF-SAT ∈ NP

Prove HALF-SAT ∈ NP-Hard

We show that HALF-SAT is NP-Hard, by reducing SAT to HALF-SAT.

- Let F be a CNF SAT instance formula with n variables, x1, x2, ... , xn.
- We now construct a new CNF formula F' which is an instance of HALF-SAT as follows: F' will have all the n variables of F - x1, x2, ... , xn along with additional new variables y1, y2, ... , yn where yi = ¬ xi
- Now coming to clauses in F, for each clause C in F, we convert it into the form of C ∧ C' in F' Where C' corresponds to the clause C in F, but its variables replaced with the new variables yi.

Ex: C = (x1 ∨ x2) , then F' has (x1 ∨ x2) ∧ (y1 ∨ y2)
Clearly, we can see that this reduction is polynomial. i.e., no. of variables in F' = 2n Thus, even if n is odd or even, in F' the total variables will be even (2n), so we can divide variables exactly in half. And If there are m number of clauses in F, then no. of clauses in F' = 2m

Consider the example : Let F = (x1 ∨ ¬ x2 ∨ x3) , and x1 = True, x2 = True, x3 = False
F = T ∨ F ∨ F = True (Satisfiable Assignment)

Then F' = (x1 ∨ ¬ x2 ∨ x3) ∧ (y1 ∨ ¬ y2 ∨ y3) , here y1 = False, y2 = False, y3 = True (since they are the negation of the corresponding xi variables) F' = (T ∨ F ∨ F) ∧ (F ∨ T ∨ T) = T ∧ T = True (Satisfiable Assignment)

If we observe the variables in F' our HALF-SAT formula, we have 3 True variables, x1 , x2 , y3 out total 6 Variables x1 , x2 , x3 , y1 , y2 , y3. And the other 3 variables are assigned False. Both F and F' are satisfiable, based on our Construction.

Claim => HALF-SAT is satisfiable, if and only if SAT is satisfiable
If SAT is satisfiable , then HALF-SAT is also satisfiable: Clearly, we can observe this based on our construction which is a polynomial reduction.

<= If HALF-SAT is satisfiable, then SAT is also satisfiable:
Given, HALF-SAT is satisfiable, we observe from the above example, that at least one of the literals in each clause of HALF-SAT must have a truth assignment. Since every clause in the original SAT instance F is present in our HALF-SAT instance F', at least one literal in the clause C of F must have a truth assignment, making the whole SAT instance F satisfiable.

Thus, it is concluded that we can reduce SAT to HALF-SAT. SAT ≤p HALF-SAT This concludes HALF-SAT ∈ NP-Complete

Problem 3:

Prove course selection problem is in NP:
We can verify in polynomial time if given at least k courses, are they aligning in disjoint time intervals.

1. We first check the number of courses in those intervals.
2. We check if at most one course is there in each time interval
3. Finally, we verify if all the courses will correspond to their respective time intervals without overlapping with other jobs.

The verification takes O(mn) time, where n is the number of courses and m is the number of time intervals.
Therefore, this problem is in NP.

Prove course selection problem is in NP -Hard:

We prove this by reduction from Independent Set (IS).

Let G(V,E) be a graph having independent set IS. We will take an instance S of the independent set and a number k, and we construct an instance C of our problem of choosing courses.

Construction :

i. For each vertex $v_i \in V$, we will construct a course c_i .
ii. For each edge $v_i\ v_j \in E$, we construct a time interval t_{ij}
iii. In this way, we are copying k courses to construct our graph. This reduction can be done in polynomial time.

Claim => G has an independent set of the size k if and only if there is a valid course selection of size k.

Assume that G contains an independent set $\{v_{i1}, v_{i2}, \ldots v_{ik}\}$. Then, we can include k courses $c_{i1}, c_{i2}, \ldots c_{ik}$, since no two of them require a common interval.

<= There is a valid course selection of size k, if and only if there is a Independent set of size k.

Assume that there is a valid k course selection $c_{i1}, c_{i2}, \ldots c_{ik}$. Then the set $\{v_{i1}, v_{i2}, \ldots v_{ik}\}$ is an independent set of G; for otherwise, there must be some y and z such that $v_{iy}v_{iz} \in E$, which means both c_{iy} and c_{iz} require the interval $s_{iy}i_z$, contradicting the fact that c_{ih} and c_{il} are not conflicting.

Thus, the problem is in NP-Complete.

Problem 4:

We know that planar graph is 4-colorable.

Let G(V,E) be a planar graph. Now let's use 4 different colors to color its vertices such that no two adjacent vertices have same color.
The vertices V of the graph G will now be grouped together based on their color.
Thus, we have 4 color groups: C1, C2, C3, C4 .
We pick the largest group out of these 4 groups, i.e., the group with largest number of vertices. Assume the largest group to be C4 . Its size must at least be V/4 (if it has fewer vertices, then the total vertices would be fewer than V).
So, apart from C4, we must have at most V - V/4 = 3V/4 other vertices. Thus, we will have at most 3V/4 vertices as vertex cover on Planar Graph G(V,E).

Now, given the assumption that optimal vertex cover (i.e., min. vertex cover) is at least V/2
We find the approximation factor by taking ratio of our Algorithm Output to Optimal Output
$\propto$ = 3V/4 : V/2 = 3/2 = 1.5

Therefore ALG ≤ 1.5 OPT, hence our algorithm approximation ratio is 1.5

Problem5:

a) Let T be DFS tree obtained by running the DFS traversal algorithm on an arbitrary vertex as root.

Let L be the set of leaf nodes in the DFS tree, and N be the set of non-leaf nodes.
In the graph G, let V be the total number of vertices. (V – L = N)

In a DFS tree, there as no edges between leaf nodes, as all edges are either forward edges that go from parent to child or backward edges which go from child to parent.

If suppose, there is an edge (u, v) that is not covered by our DFS traversal, which implies both u and v are leaves. Thus, we have contradiction, it impossible for leaves to have edges between them, then they won't be leaves anymore.

Hence, we can say that V – L is the vertex cover, i.e., N which is the set of non-leaf nodes.

b) Since tree is a part of a graph G, then vertex cover of graph must also be a vertex cover of its tree T. For a tree T, we can assume that none of its leaves are in the optimal vertex cover set. It is better to take the parent as a part of vertex cover, instead of its leaf.

For any vertex cover, VC $\sum$ deg (v) $v \in VC \geq |E|$, since the nodes in the vertex cover must cover all the edges, and sometimes two different nodes might cover the same edge twice or more.
Also, $\sum$ deg (v) $v \in VC$ + $\sum$ deg (v) $v \notin VC$ = 2|E| since, the sum of degree of vertices in graph is twice the number of edges (undirected edges, each edge is counted twice, as it connects two vertices)

From above, we can say $\sum \deg(v)\ v \notin VC \leq 2\ |E|$.

Since none of the leaves are in vertex cover, and each non-leaf node has a degree of at least 2. We get $2|E| \geq L + 2(n - L - VC)$, which further can be reduced to : $VC \geq n - L + 1\ 2$

This concludes that, for any tree, the vertex cover must have these many number of vertices, which is more than half of vertices returned by the algorithm. Thus, concluding that the α value to be 2. We have 2 - α approximation algorithm.