CSCI 570 : HW-06 (Spring 2021)

April 28, 2021

Question 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.

Answer

Suppose we are given a SAT expression with variables $x_1, x_2, ..., x_n$.

$$(x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee x_2 \vee \bar{x}_4)$$

We can use variables $z_1, z_2, ..., z_n$ to reduce the above expression to ILP instance. We will convert each clause into an inequation as follows:

$z_1 + z_2 + (1 - z_4) \geq 1$

$(1 - z_1) + z_3 + z_2 + (1 - z_4) \geq 1$

$z_i \in \{0, 1\}$ for all $i = 1, 2, ...,$

Claim : SAT has a satisfying assignment if only if ILP instance has a solution.

Proof :

1. For every valid assignment in SAT, there exists a feasible solution to ILP instance.

For every variable x_i, True value for variable x_i denotes $z_i = 1$ and similarly False value for x_i denotes $z_i = 0$. For any clause in SAT to be true, its reduction to inequality using variables z_i must be greater than 0. Since we are replacing x_i by z_i and $\bar{x}_i$ by $(1 - z_i)$, clause will have value greater than 0 if any of the literal in clause is True, which indicates satisfiability of that clause. If all clauses are satisfiable (valid assignment) that implies all inequalities are satisfied too means there exists a feasible solution to ILP instance.

2. For every feasible solution to ILP, there exists a valid assignment in SAT.

If an inequality in ILP is satisfied, that means atleast one of the z_i or $1 - z_i$ in that inequality is 1 means x_i or $\bar{x}_i$ is True for a clause from which that inequality has been reduced. This is will

satisfy that particular clause. If all inequalities in ILP are satisfied that means there exists a valid assignment for that SAT expression.

Here, we have proved that ILP can be reduced from SAT problem ($SAT \leq_p ILP$), so we can conclude that ILP is indeed a NP-hard problem.

Question 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.

Answer:

We can proof HALF-SAT is NP-Complete by proving its NP and can be reduced from another NP-Complete problem i.e. SAT.

1. The HALF-SAT is NP since we can verify given assignment in polynomial time if its satisfy the HALF-SAT property (exactly half of variables are True) or not.
2. Now lets reduce a SAT problem to HALF-SAT. Let F be the CNF expression with $n = 3$ variables with valid assignment.

   $$F = (\bar{x}_1 \vee x_2)$$

   .

   We will convert F to into another expression F' with new variables $y_1, y_2, ..., y_n$ such that $y_i = \bar{x}_i$.

   $$F' = (\bar{x}_1 \vee x_2) \wedge (y_1 \vee \bar{y}_2)$$

   Claim : There exists a valid solution to SAT if and only if there is valid solution for the HALF-SAT.

   - We will first prove that a valid assignment exists for HALF-SAT if there is some assignment for SAT.
     Take the valid assignments for the given example as $x_1 = true, x_2 = true$ or $x_1 = false, x_2 = true/false$. The HALF-SAT problem evaluates to true for all these value of x as y is the invert of x.

     Case 1: $x_1 = true, x_2 = true, y_1 = false, y_2 = false$. Thus $F' = (F \vee T) \wedge (F \vee T) = T$.
     Case 2: $x_1 = false, x_2 = true/false, y_1 = true, y_2 = true/false$. Thus, $F' = (T \vee T) \wedge (T \vee T) = T$ and $F' = (T \vee F) \wedge (T \vee F) = T$. Thus, HALF-SAT has a valid assignment for all valid assignment of SAT.
   - We will now prove that if there is an assignment to HALF-SAT, then there is a valid assignment to SAT. Since we are always inverting every x in out new clause and that the HALF-SAT is in CNF, all individual clauses of HALF-SAT will be true. Meaning the clause corresponding SAT is also true.

Thus we can say that the HALF-SAT problem is NP-complete from steps 1 and 2.

Question 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.

Answer

Suppose we are given a graph G(V,E), Independent set problem (decision problem) is the problem to tell if there exists a independent set which consists of K vertices. Independent Set problem is NP complete.

We will prove that course selection problem (CSP) described in question is also NP complete using following steps.

1. We will prove that CSP is NP.

2. We will prove that CSP is polynomial reduction of another NP-Complete Problem (Independent-Set Problem).

NP-Completeness of Course Selection Problem

1. Given an set of K courses we can easily verify in polynomial time that if any of the given k courses has a overlapping interval. Thus, CSP is NP problem.

2. We will reduce Independent-Set problem to Course Selection Problem. For the graph G(V,E), each vertex denotes a course and edge between 2 vertices denotes that there is overlap in their timings.

   Claim : There exists a Independent set of atleast K vertices in G iff it is possible to select atleast K courses.

   Proof

   (a) If there exists an independent set of K vertices in G, then we can select K courses with non-overlapping time interval.

   Since there exists a independent set of K vertices, we are able to find K vertices (courses) which do not share any edges means there is no overlap in timings in any of the courses. Thus we can select K courses with non-overlapping time intervals.

   (b) If we can select K courses with non-overlapping time intervals, then there exists an independent set of K vertices in G.

   Since we are able to select K courses with non-overlapping time intervals, then their corresponding vertices will also not share any edges. Thus there must exist a Independent set of K vertices.

Thus, we can say Course Selection Problem is indeed a NP-Complete Problem.

Question 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.

Answer

Algorithm:

1. Lets us propose an algorithm ALG to find the vertex cover of graph G using the idea that is 4 colored. Let VC be the vertex cover of our algorithm ALG anc VC* be the vertex covert generated by the optimal algorithm OPT.
2. Assumption |VC*| >= |V|/2
3. In the 4 colored graph |V| vertices, there will be at least one color (let's say color 1) which is used for more than |V|/4 vertices i.e. $V_1 \geq V/4$
4. So the remaining three colors will cover less than 3/4 |V| i.e. $V_2 + V_3 + V_4 \leq 3/4V$
5. We can discard V_1 from the solution vertex cover since each of the edges covered by these vertices will also be covered by the another vertex which belongs to $\{V_2, V_3, V_4\}$ which we can include in the vertex cover VC. So we have $|VC| \leq 3/4|V|$.

Proof of Correctness:

Vertex cover problem is the min. subset of vertices that include at least one endpoint of every edge in the graph. Since we are already given a 4 colored graph, the algorithm will traverse the graph using BFS/DFS and maintain a hashmap to store the count of vertices that are colored by each of the 4 colors. At the end to traversal, all the vertices except for those with maximum color count will be included in the output vertex cover and algorithm terminates. Hence ALG will always terminate and return the vertex cover of the graph.

Time Complexity:

AlG is a polynomial time algorithm as we just need to find the no. of vertices colored with each color and discard the vertices with the maximum color count. Vertex cover will be total no of vertices- vertices colored in max color count. This can be done by traversing the graph using BFS/DFS and storing the color count values in a hashmap like data structure.

Approximation Ratio:

From the assumption we have $|VC \cdot| \geq |V|/2$ and we have shown that $|VC \leq 3/4|V|$. Hence, taking ratio of both.

|VC*|/|VC| = (|V|/2)/(3/4 |V|), i.e. $|VC| \leq 3/2|VC \cdot |4$.

Hence the approximation ratio = 3/2.

Question 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.

1. Show that the output is indeed a vertex cover for G.

2. 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.

Answer

1. Let L be the set of leaves from DFS tree T. According to the heuristic V-L is the vertex cover, where V is set of all vertices. Since in DFS tree all edges go from parent to child or child to parent. Thus there is no edge between leaves otherwise one of them would not be a leaf and hence V-L is a valid vertex cover.

2. Approximation ratio is within 2 times the optimal. For a tree we can assume that none of the leaves are in the optimal vertex cover. For any vertex cover VC we have

$$\sum_{v \in VC} deg(v) \geq |E|$$

since it must cover every edge. Also we have

$$\sum_{v \in VC} deg(v) + \sum_{v \notin VC} deg(v) = 2 \cdot |E| \implies \sum_{v \notin VC} deg(v) \leq 2 \cdot |E|$$

Since all the leaves are out of vertex cover and for any nonleaf the degree is atleast 2, we get.

$$2 \cdot |E| \geq |L| + 2 \cdot (V - |L| - VC) \implies VC \geq \frac{V - |L| + 1}{2}$$

Thus, for every tree any vertex cover must have these many vertices, which is more than half of the vertices returned by the algorithm, proving that the algorithm is 2-approximation.