CSCI 570 Homework 6
Spring 2023

Due Date: Apr. 23, 2023 at 11:59 P.M.

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.

   For each boolean variable x_i, assign an integer variable $y_i \in \{0, 1\}$, i.e $0 \leq y_i \leq 1$.

   For each clause, we introduce a constraint of the form

   $$t_i + t_j + t_k \geq 1$$

   where

   - $t_i = y_i$ if x_i
   - $t_i = 1 - y_i$ if $\bar{x}_i$

   For example, a clause of the form $x_3 + \bar{x_7} + x_9$ will be translated to

   $$y_3 + (1 - y_7) + y_9 \geq 1$$

   Clearly, if there exists a solution for the 3-SAT, then there exists a solution for the ILP by setting y_i to one if x_i is assigned true else zero in the ILP solution. Similarly for the reverse direction, if y_i is set to one, set x_i to true else false in the 3-SAT solution.

   On the hardness of integer linear programs, since 3-SAT reduces to ILP, ILP must be atleast as hard as 3-SAT and 3-SAT is NP-Complete. Hence ILP is NP-Hard. (2pts)

2. There are N cities, and there are some undirected roads connecting them, so they form an undirected graph $G = (V, E)$. You want to know, given K and M, if there exists a subset of cities of size K, and the total number of roads between these cities is larger or equal to M. Prove that the problem is NP-Complete.

(a) Show this problem is in NP.

(b) Show this problem is in NP-hard.

solution: We can formalize the problem as, given an undirected graph $G = (V, E)$ and asks for if there exists a subset S of K vertices such that the number of edges in the induced graph $G[S]$ is at least M.

(a) The solution of the problem can be easily verified in polynomial time, just check the total number of roads between these cities. Thus it is in NP.

(b) Given a decision version of the independent set problem, suppose it asks for whether an independent set exists of size K. Then we ask whether the complement of the graph has a subset S of K vertices such that number of edges in the induced graph $G[S]$ is at least $K(K-1)/2$. If so, we find an independent set of sizes K in the original graph. For the other direction, if there is an independent set of size K in the original graph, we can find a subset S of K vertices such that number of edges in the induced graph $G[S]$ is at least $K(K-1)/2$. Thus we reduced the independent set problem to this problem in polynomial time. Since an independent set problem is NP-Complete, so this problem is in NP-Hard.

Thus this problem is NP-Complete.

3. Consider a modified SAT problem, SAT' in which a CNF formula with m clauses and n variables $x_1, x_2, \ldots, x_n$ outputs YES if exactly $m-2$ clauses are satisfied, and NO otherwise. Prove that SAT' is NP-Complete.

(a) Show SAT' is in NP.

(b) Show SAT' is in NP-hard.

solution: To show that SAT' is NP-Complete,
First we will show that SAT' $\in$ NP: Given the assignment values as certificate, we can evaluate the SAT' instance and verify if it is satisfied. This is same as the SAT-verification. Moreover, we can count the number of satisfied clauses and check if it is equal to m $-$ 2 in linear time.
Next, we show that SAT $\leq$ p SAT': Construction: Add four more clauses

$x_1, x_2, \neg x_1, \neg x_2$ to the original SAT instance.

Claim: CNF formula obtained for SAT', F' has an assignment which satisfies SAT' iff CNF formula of SAT, F has an assignment which satisfies SAT.

$\Rightarrow$) if F has an assignment which satisfies SAT, then F$'$ has an assignment which satisfies SAT'
Proof: If an assignment $x_1 \ldots x_n$ satisfies F, it satisfies exactly two of the four extra clauses, giving exactly $m+2$, which is m ' -2 satisfied clauses for F$'$.
$\Leftarrow$) if F' has an assignment which satisfies SAT', then F has an assignment which satisfies SAT
Proof: By construction, the only unsatisfied clauses for F must be one of x_1 or $\neg x_1$ and one of x_2 or $\neg x_2$, so all the original m clauses are satisfied.

4. Longest Path is the problem of deciding whether a graph $G = (V, E)$ has a simple path of length greater or equal to a given number k. Prove that the Longest path Problem is NP-complete by reduction from the Hamiltonian Path problem.

   solution: The Longest Path Problem is in NP. Given a solution path P, we just check that P consists of at least k edges, and that these edges form a path (where no vertex is used more than once). This verification can be done in polynomial time. The reduction follows directly from Hamiltonian Path. Given an instance of Hamiltonian Path on a graph G = (V, E). We create an instance of the longest path problem G' as follows. We use exactly the same graph, i.e G' = G and we set k = V-1. Then there exists a simple path of length k in G' iff G' contains a Hamiltonian path. The proof is obvious.

5. Assume that you are given a polynomial time algorithm that decides if a directed graph contains a Hamiltonian cycle. Describe a polynomial time algorithm that given a directed graph that contains a Hamiltonian cycle, lists a sequence of vertices (in order) that form a Hamiltonian cycle.

solution: Let $G = (V, E)$ be the input graph. Let A be an algorithm that decides if a given directed graph has a Hamiltonian cycle. Hence $A(G) = 1$. Pick an edge $e \in E$ and remove it from G to get a new graph G'. If $A(G') = 1$, then there exists a Hamiltonian cycle in G' which is a subgraph of G, set $G = G'$. If $A(G') = 0$, then every Hamiltonian cycle in G contains e. Put e back into$G$. Iterate the above three lines until we are left with exactly V edges. Since after each step we are left with a subgraph that contains a Hamiltonian cycle, at termination we are left with the set of edges that forms a Hamiltonian cycle. Starting from an edge, do a BFS to enumerate the edges of the Hamiltonian cycle in order.

6. Consider the maximum cut problem in an undirected network $G = (V, E)$. In this problem, we are trying to find a set of vertices $C \subset V$ that maximizes the number of edges with one endpoint in C and the other endpoint in $V \backslash C$ (note that there is no constraint on any special nodes s and t unlike in the minimum cut problem). A simple greedy algorithm would be to start with an arbitrary set C and then ask if there is a vertex $v \in V$ such that we will increase the number of edges in the cut by moving it from one side of the cut to the other. We then iterate this process until there is no such vertex. a) Show that the algorithm must terminate, and b) show that when the algorithm stops, the size of the cut is at least half of the optimum.

   a) At each iteration the number of edges in the cut goes up. Since there are only a $|E|$ edges in the graph, the algorithm will terminate in at most $|E|$ steps. (3pts)

   b) Let $D = V \backslash C$. Let d_i be the vertices in D and c_i be the vertices in C.

   Let $I(c_i) = \{e | e = (c_i, v), v \in C\}$. Similarly, let $I(d_i) = \{e | e = (d_i, v), v \in D\}$. This is the internal degree of the nodes.

   Let $O(c_i) = \{e | e = (c_i, v), v \in D\}$. Similarly $O(d_i) = \{e | e = (d_i, v, v \in C)\}$. This is the external degree of the nodes, i.e edges crossing over from one partition to other. Note that the total number of edges crossing one partition to other is given by $Y := \sum_{c_i} O(c_i) = \sum_{d_i} O(d_i)$

   When the algorithm terminates, we must have $I(x) \leq O(x), x \in V$ as

otherwise we can move x to the other partition to increase the edges going across the partition.

If $D(x)$ be the degree of the node x, then

$$D(x) = I(x) + O(x) \leq 2O(x)$$

$$\sum_x D(x) \leq 2\sum_x O(x) = 2\left(\sum_{c_i} O(c_i) + \sum_{d_i} O(d_i)\right) = 2(Y + Y) = 4Y$$

$$2|E| \leq 4Y \quad or \quad \frac{|E|}{2} \leq Y$$

where we have used the fact that sum of degrees equals twice the number of edges in the graph.

7. 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: Take a planar graph G = (V, E), and color it in 4 colors. This will split all vertices in four color groups C1, C2, C3 and C4. Find the largest group. Let it be C4. Its size must be at least V/4. So there must be at most V V/4 = 3V/4 other vertices. We will use these vertices as a vertex cover.

8. Consider a well-known 0-1 Knapsack problem. Given collection of n items with integral sizes $w_1, \ldots, w_n > 0$ and values $v1, \ldots, v_n > 0$, and an integer knapsack capacity $W > 0$. The problem to find integers $w_1, \ldots, w_n > 0$ such that the total value is maximized is known to be NP-Hard. As a possible heuristic, let's try the following greedy algorithm: sort items in non-increasing order of v_i/w_i and greedily pick items in that order. Show that this approximation algorithm is pretty bad, namely it does not provide a constant approximation.

   solution: Let $n = 2, w_1 = 1, v_1 = 2, and w_2 = W, v_2 = W$. After sorting we have $v_1/w_1 = 2, v_2/w_2 = 1$. So our approximation algorithm will choose the first item with the total value 2. However, the OPT

value is W. Algorithm chooses the first object, because it has a higher v_i/w_i ratio, but choosing it leaves much of the knapsack empty. The ratio$ALG/OPT = 1/W$ which is arbitrarily small as W grows to infinity.