CS 270 Homework 10

Neel Gupta

April 12, 2023

Problem 1. Consider the partial satisfiability problem, denoted as 3-Sat(α). We are given a collection of k clauses, each of which contains exactly three literals, and we are asked to determine whether there is an assignment of true/false values to the literals such that at least αk clauses will be true. Note that 3-Sat(1) is exactly the 3-SAT problem from lecture.

Prove that 3-Sat(15/16) is NP-complete.

Answer:

We need to show that 3-Sat(15/16) is in NP and that it is NP-hard to show that it is NP-complete.

1. Given a collection of n clauses and assignments of true or false to the literals, we can verify in polynomial time whether at least $(15/16)k$ clauses are satisfied. Therefore, 3-Sat(15/16) is in NP.

2. We want to show that 3-Sat(15/16) is in NP-hard, and we can do that by reducing the 3-SAT problem into an instance of 3-Sat(15/16).

Given a 3-SAT instance with a collection of k clauses, we will construct an instance of 3-Sat(15/16) by creating a new set of clauses with $15k$ clauses, where the first k clauses are the original clauses from the 3-SAT problems, and the remaining sets of clauses are duplicates of the first k clauses.

Now, we want to show that the original 3-SAT problem is satisfiable if and only if the new 3-Sat(15/16) is satisfiable with at least $(15/16)(15k) = 225k/16$ clauses satisfied.

Forward: If 3-SAT is satisfiable with k clauses, then the new 3-Sat(15/16) has at least $225k/16$ satisfied clauses.

Backward: if the new 3-Sat(15/16) instance has at least $225k/16$ satisfied clauses, then the original 3-SAT instance is satisfiable.

Since the reduction from 3-SAT to 3-Sat(15/16) can be computed within polynomial time and 3-SAT is NP-complete, 3-Sat(15/16) is NP-hard. As it is also in NP, 3-Sat(15/16) is NP-complete.

Problem 2. Consider a graph $G = (V, E)$ and two integers k, m.

(a) A k-clique is a subset of nodes $u_i \in G, i = 1, ..., k$ such that there is an edge connecting e3very pair of distinct vertices u_i, u_j. In other words, the k-clique is a complete sub-graph of G. Prove that finding a clique of size k is NP-complete.

(b) The Dense Subgrapgh problem is to find a subset V' of V, whose size is at most k and are connected by at least m edges. Prove that the Dense Subgrapgh problem is NP-complete.

Answer:

(a) We need to show that finding a clique of size k (k-clique) is NP-complete, we need to show that it is in NP and that it is NP-hard to then conclude that it is NP-complete.

1. Given a graph $G = (V, E)$ and a subset of nodes $V' \subseteq V$, we can verify in polynomial time whether V' forms a k-clique. We simply need to check that there are $\binom{k}{2}$ edges among the vertices in V', which can be done in polynomial time with respect to the input size. Therefore, the k-clique problem is in NP.

2. We will now show that k-clique is NP-hard by reducing the 3-SAT problem to the k-clique problem. Given a 3-SAT instance with n variables and m clauses, we will construct a graph $G = (V, E)$ as follows:

- For each literal in each clause, create a vertex in V.
- Connect two vertices with an edge if they correspond to literals from different clauses and they are not negations of each other.

Now, we will show that the 3-SAT instance is satisfiable if and only if the graph G has a k-clique, where $k = m$ (the number of clauses).

Forward: If the 3-SAT instance is satisfiable, there exists a truth assignment such that each clause has at least one true literal. Choose one true literal from each clause, and the corresponding vertices in G will form a k-clique since they are not negations of each other and come from different clauses.

Backward: If the graph G has a k-clique, each vertex in the clique corresponds to a literal from a different clause. The truth assignment that makes these literals true will satisfy the 3-SAT instance.

Since the reduction from 3-SAT to k-clique can be computed within polynomial time and 3-SAT is NP-complete, k-clique is NP-hard. As it is also in NP, k-clique is NP-complete.

(b) To prove that the Dense Subgraph problem is NP-complete, we need to show that it is in NP and that it is NP-hard. We will do this by first showing it is in NP, and then by reducing the k-Clique problem, which we have shown to be NP-complete, to the Dense Subgraph problem.

1. Given a graph $G = (V, E)$ and a subset of nodes $V' \subseteq V$, we can verify in polynomial time whether V' forms a Dense Subgraph with at most k vertices and at least m edges. We simply need to check that there are at least m edges among the vertices in V', which can be done in polynomial time with respect to the input size. Therefore, the Dense Subgraph problem is in NP.

2. Now, we will show that the graph G has a k-Clique if and only if it has a Dense Subgraph with at most k' vertices and at least m' edges.

Forward: If the graph G has a k-Clique, then by definition, it has a complete subgraph with k vertices and $\binom{k}{2}$ edges. This subgraph is a Dense Subgraph with at most k' vertices and at least m' edges since $k' = k$ and $m' = \binom{k}{2}$.

Backward: If the graph G has a Dense Subgraph with at most k' vertices and at least m' edges, the Dense Subgraph must have exactly k' vertices since any subgraph with fewer vertices cannot have m' edges. The Dense Subgraph is a complete subgraph, as it has $\binom{k}{2}$ edges, so it is a k-Clique.

Since the reduction from the k-Clique problem to the Dense Subgraph problem can be computed within polynomial time and k-Clique is NP-complete, the Dense Subgraph problem is NP-hard. As it is also in NP, the Dense Subgraph problem is NP-complete.

Problem 3. Consider a modified SAT problem, SAT' in which given a CNF formula having m clauses and n variables $x_1, x_2, ..., x_n$, the output is YES if there an assignment to the variables such that exactly $m - 2$ clauses are satisfied, and NO otherwise. Prove that SAT' is NP-complete.

Answer:

We want to first show that SAT' is in NP, then we want to show that it is NP-hard.

1. Given a CNF formula with m clauses and n variables, if there exists an assignment to the variables such that exactly $m - 2$ clauses are satisfied, we can easily verify this in polynomial time. We just need to check each clause and count the number of satisfied clauses. If the number of satisfied clauses is equal to $m - 2$, the output is YES; otherwise, it is NO. Therefore, SAT' is in NP.

2. To show that SAT' is NP-hard, we will reduce the 3-SAT problem, a known NP-hard problem, to SAT'. The reduction should be able to be computed within polynomial time.

Given an instance of 3-SAT with CNF formula F having m clauses and n variables, we will construct a SAT' instance with a CNF formala F' having $m + 2$ clauses and n variables from 3-SAT as follows:

- Create a new variable z.
- Add two new clauses to F: (z) and $(\neg z)$
- Let F' be the resulting formula.

Now, we can show that F is satisfiable if and only if F' is satisfiable with exactly m clauses satisfied (meaning $m - 2$ clauses in F are satisfied).

Forward: If F is satisfiable, then there exists an assignment A for every variable in F such that all m clauses are satisfied. We can extend A to an assignment A' for F' by setting z to true, which will satisfy m clauses in F' as well as the new clause (z). Therefore, F' has an assignment that satisfies exactly m clauses, so the output for the SAT' problem is YES.

Backward: If F' is satisfiable with exactly m clauses satisfied, then $\exists$ an assignment A' for F' that satisfies $m - 2$ of the original clauses in F and one of the two new clauses. Without loss of generality, assume that A' satisifies (z). Now, if we restrict A' to the original n variables, we obtain an assignment A for F that satisfies $m - 2$ clauses exactly. Since F is a 3-SAT instance, it must have at least 1 assignment that satisfies all m clauses. Thus, if we modify A to satisfy the remaining 2 unsatisfied clauses in F, resulting in assignment that satisfies F. Thus the output of the 3-SAT problem is YES.

Therefore, SAT' is NP-complete since it is in NP and at least as hard as another NP-hard problem.

Problem 4. Show that Vertex Cover is still NP-complete even when all vertices in the graph are restricted to have even degree.

Answer:

We want to first show that even-degree Vertex Cover is in NP, then we want to show that it is NP-hard.

1. Given a graph $G = (V, E)$ and a positive integer k, we can verify in polynomial time whether a given subset V' is a valid vertex cover of at most size k. Next, we need to check that every edge $e \in E$ is incident to at least one vertex $v \in V'$ and that the size of V' is at most k. Thus, the modified Vertex Cover problem is in NP.

2. To show that Vertex Cover is NP-hard even when all vertices have even degree, we will reduce Vertex Cover, a known NP-hard problem, to the even-degree Vertex Cover problem. The reduction should be able to be computed within polynomial time.

Given an instance of the Vertex Cover problem with a graph $G = (V, E)$ and an integer k, we will construct a graph G' with even-degree vertices as follows:

- For every vertex $v \in V$ with odd degree, add a new vertex w_v and connect it to v via a new edge to G'.
- Add a new vertex w, and connect it to all new vertices w_v created to edit odd-degree vertices.

Now, we can show that G has a vertex cover of size k if and only if G' has a vertex cover of size $k + 1$ to account for the new vertex w.

Forward: Suppose G has a vertex cover V' of size k. Create a new vertex cover V'' for G' by adding the newly created w vertex to to V'. Since V' is a vertex cover of G, all edges in G are covered. Since all new edges are from new vertices are connected to w, V'' is a valid vertex cover of size $k + 1$ of the graph G'.

Backward: Suppose G' has a vertex cover V'' of size $k + 1$. Since vertex w is connected to all new vertices w_v, it must be part of V''. Remove w from V'' to obtain V' of size k. Then, V' is a valid vertex cover of size k for the graph G because all the edges in G are also in G'.

Therefore, the even-degree Vertex Cover problem is NP-Complete since it is in NP and at least as hard as another NP-hard problem.

Problem 5. Consider a set $A = \{a_1, ..., a_n\}$ and a collection $B_1, B_2, ..., B_m$ of subsets of A (i.e. $B_i \subseteq A$ for each i).

We say that set $H \subseteq A$ is a hitting set for the collection $B_1, B_2, ..., B_m$ if H contains at least one element from each B_i. If $H \cap B_i$ is not empty for each i (so H "hits" all the sets B_i).

We now define the Hitting Set Problem as follows. We are given a set $A = \{a_1, ..., a_n\}$, a collection $B_1, B_2, ..., B_m$ of subsets of A, and a number k. We are asked: Is there a hitting set $H \subseteq A$ for $B_1, B_2, ..., B_m$ so that the size of H is at most k?

Prove that Hitting Set is NP-complete.

Answer:

We want to show that the Hitting Set problem is NP-complete first showing that it is in NP and then that it is NP-hard.by reducing the even-degree Vertex Cover problem to the Hitting Set problem. This reduction should be computable within polynomial time.

1. We will demonstrate that given a set $A = a_1, \ldots, a_n$, a collection $B = B_1, B_2, \ldots, B_m$ of subsets of A, and a non-negative integer k, we can verify in polynomial time whether a given subset $H \subseteq A$ is a valid hitting set of size at most k.

To verify if H is a valid hitting set of size at most k, we can perform the following checks:

- Verify that the size of H is at most k.
- For each subset B_i in the collection B, check that $H \cap B_i$ is not empty.

Both checks can be performed in polynomial time with respect to the input size, as the number of elements in A and the number of subsets in B are both polynomially bounded.

2. Given an instance of the Vertex Cover problem with a graph $G = (V, E)$ and an integer k, we will construct an instance of the Hitting Set problem as follows:

- Let the set $A = V$.
- For each edge $e \in E$, create a subset B_e consisting of the two endpoints of e. Let $B = B_1, B_2, \ldots, B_m$ be the collection of these subsets.

Now, we can show that G has a vertex cover of size k if and only if there is a hitting set $H \subseteq A$ of size at most k for the collection B.

Forward: Suppose G has a vertex cover V' of size k. Then, V' covers every edge in G. For each edge $e \in E$, at least one endpoint of e is in V'. This means that V' intersects every subset B_e in the collection B. Therefore, V' is a hitting set for the collection B with size at most k.

Backward: Suppose there is a hitting set $H \subseteq A$ of size at most k for the collection B. Since H intersects every subset B_e, it means that at least one endpoint of each edge $e \in E$ is in H. Therefore, H is a vertex cover for the graph G with size at most k.

Since the reduction from the Vertex Cover problem to the Hitting Set problem can be computed within polynomial time and the Vertex Cover problem is NP-complete and the Hitting Set problem is in NP, the Hitting Set problem is also NP-complete.