Homework 10

CS570 Spring 2025

Due: Apr 18, 2025

1. (**15 points**)

Suppose you move to a new city. Due to busy work schedule, you have to eat out very often. There are m different food items that you are fond of. You seek recommendations from friends and online, and you get a list of n restaurants, each offering one or more (or none) of your m favorite foods. You realize this list is too big to try them all, so you want to remove as many from the list as possible while making sure that each of your favorite foods is offered in at least one of the remaining restaurants. Thus, the problem can be stated as deciding if at least k restaurants can be removed from the given list so that each of your favorite foods is offered in at least one of the remaining restaurants. Prove that the problem of *shortening restaurant list* (SRL) is

a) in NP.

b) NP-Hard using a poly-time reduction from Vertex Cover.

Solution:

(a) (Showing *shortening restaurant list* (SRL) in NP)

1) Certificate: A subset of restaurants to remove

2) Certifier: The given certificate can be verified in polynomial time: Just check the number of the restaurants in the certificate is k, and check that each food is in at least one **remaining** restaurant after removing the k restaurants. Simply by brute force iteration, this can be done in $O(mn)$ time (where we have n restaurants and m foods).

(b) (Showing SRL is NP-Hard) We show that vertex-cover (VC) $\leq_P$ SRL.

Construction: Given an instance $(G = (V, E), k')$ of VC, we create an SRL instance as described below. We construct set of restaurants R by constructing a restaurant r_u for each vertex $u \in V$, and set of foods F by creating a food f_e for each edge $e \in E$ and let each restaurant r_u have all foods f_e such that e is incident on u. We set the value k for SRL as $k = |V| - k'$. It is easy to see that this construction can be done in poly-time.

Claim: The constructed instances of SRL (R, F, k) outputs YES if and only if the VC instance (G, k') outputs YES. (Can also expand this as "The constructed SRL instance has k redundant restaurants if and only if G has a VC of size k' ").

Proof:

1) Suppose G has a vertex cover S of size k'. Then consider the set of restaurants R' restaurants corresponding to the vertices in S (Formally written as $R' = \{r_u | u \in S\}$. For each edge $e = (u, v)$ to be covered, we must have u or v in S — say $u \in S$. Hence, $r_u \in R'$, which implies that f_e is in at least one restaurant in R' since it is in r_u by construction. Thus, restaurants in R' altogether ensure that each food is in at least one of them, making all of restaurants in $R - R'$ removable ($|V| - k'$ of them).

2) Suppose we have a set of k restaurants that can be removed successfully, that is, every food item is offered in at least one of the remaining restaurants - call it R'. R' has size $|R| - k =$

$|V| - (|V| - k') = k'$. Then, consider the set of vertices S in G corresponding to the restaurants in R' (Formally written as $S = \{u | r_u \in R'\}$. Since each food f_e is in at least on restaurant r_u in R', this means the corresponding edge e has an endpoint u in S by construction. Hence S is a VC of G of size k'.

Thus, we can reduce VC to SRL in polynomial time as required.

Rubrics (15 points):

- 4 pts: Correctly shows in NP
- 5 pts: Correctly constructs and explains the reduction.
- 6 pts: Proof in both directions.

2. (**16 points**)

Given a graph $G = (V, E)$ and a positive integer r, the dominating set problem (DSP) is to decide if there is a subset $B \subseteq V$ of size r such that every vertex that is not in B, has an adjacent vertex in B. Prove that the vertex cover (VC) problem is polynomial time reducible to DSP. For simplicity, you may assume that VC remains NP-complete when restricted to input graphs with no isolated vertices (i.e., each vertex has at least one neighbor), and reduce from this version of VC if required.

Useful terminology: Similar to the context of Vertex cover where an edge is *covered* by an endpoint vertex being included in the VC set, here, we say a vertex u is *dominated* if u itself or one of its neighbors is included in the dominating set (DS); thus, the DS solution is a set of vertices that collectively *dominates* all the vertices in the graph (similar to how a VC solution collectively *covers* all the edges).

Solution:

We show that vertex-cover $\leq_P$ DSP as is given in the problem statement.
Construction: Given an instance $(G' = (V', E'), k)$ of vertex-cover, we create a DSP instance (G, r) as described below. First, we explain the solution by assuming no isolated vertices in G' (i.e., each vertex has at least one neighbor) and then explain the tiny change needed to handle the case with isolated vertices (**explicitly** assuming no isolated vertices will be accepted for full credit here).

For every edge $e = (u, v) \in E'$, create a new vertex x_e and two new edges (u, x_e) and (v, x_e). (It will help to imagine a bipartite picture with original vertices on one side and the the new "edge-vertices" on the other side).

Claim: G' has a VC of size at most k if and only if G has a dom. set of size at most r $(= k)$.

Proof:

- Suppose we have A as a vertex cover of G' of size $\leq k$. Then, we show A is a dominating set for G. For any arbitrary vertex $u \in V'$, since it has at least one neighbor, consider any of them, say v (i.e., $(u, v) \in E$). Since the edge (u, v) must be covered by the VC A, one of u or v is in A, which ensures u is dominated by A. Thus, all the original vertices are dominated by A in G. To argue about the new edge-vertices, again, consider any edge $e = (u, v) \in E'$): either u or v must be in the VC A to ensure e is covered, this vertex (included in A) dominates the edge-vertex x_e in G. Thus, all the new vertices are dominated by A in G as well, making it a DS of size $\leq k$.
- Conversely, suppose we have A as a dominating set of G of size $\leq k$. Now, A might contain ANY vertices in G — suppose $A = B \cup C$ where B are the vertices that were originally in V' and C are the new edge-vertices. We obtain C' from C by replacing each edge-vertex x_e by **one** of the endpoints of e. Thus C' only contains the original vertices in V' and is at most as big as C. Now, we can show that $A' = A \cup C'$ is a vertex cover of G (of size at most $|A| \leq k$, note!). For any edge e in E': 1) if x_e was an edge-vertex in C (i.e., part of the DS A), then e is covered by C' (thus, A') as we added one of e's endpoints to C'. 2) If x_e was not an edge-vertex in C, the only way for it to be dominated, would be due one of the vertices in B, say u; but x_e is only connected to e's endpoints as per the construction. Thus, this endpoint u (included in B, thus in A') covers the edge e. Thus, all the edges in E' are covered by A', making it a VC of size $\leq k$.

To handle the general case, suppose the graph G' has a set of isolated vertices V_0. Then, simply set $r = k + |V_0|$ in the construction. In the first part of the proof, given VC A of size $\leq k$, first add all of V_0 to A. This will ensure all the isolated vertices are dominated by A and keeps the new size of A within $k + |V_0|$. For the other direction, given the DS A of size at most $k + |V_0|$, first note that A must include all of V_0 since the isolated vertices would not be dominated otherwise. Therefore, we can first simply remove V_0 from A reducing its size to k, and then proceed with the rest of the argument above to to get a VC of size $\leq k$.

Rubrics (16 points):

- 6 pts: Correct construction of DSP instance from Vertex-Cover.
- 5 pts: Correctly explains/proves forward direction.
- 5 pts: Correctly explains/proves the backward direction.

3. (**15 points**)

Suppose you are a movie director. For your next movie, you are considering n actors that you like. You want the audience to witness all the new combinations of acting talent working together - i.e., you want to pick a set of actors from the n candidates, such that no two have worked together in a movie before. For this, you make a combined list of all the movies in which any of these actors have acted, giving you a total of m movies (thus, each of the movies features one or more of these n actors). You want to find at least k actors to cast for your movie such that no two have acted in the same movie before. Your input consists of n actors, m movies, along with the information of which actors have acted in which movies, and the number k. Prove that this problem of *Novel Combination Casting* (NCC) is.

a) in NP.

b) NP-hard (by choosing a suitable problem studied in class to show a poly-time reduction).

Solution:

(a) (Showing Problem in NP)

1) Certificate: A subset of actors

2) Certifier: The given certificate can be verified in polynomial time: Just check the number of the actors in the certificate is larger or equal to k, and for each pair of actors, check that there is no movie featuring both (by checking all of their movies with brute force).

(b) (Showing NCC in NP-Hard) Given an Independent Set (IS) instance: Graph $G = (V, E)$ and value k. We construct an actor A_u for each vertex $u \in V$, and a movie M_e for each edge $e \in E$ and let each actor A_u be in all movies M_e such that e is incident on u in G. We use the same k for this constructed instance of NCC. It is easy to see that this construction can be done in poly-time.

Claim: The constructed instance of NCC outputs YES if and only if the IS instance outputs YES. (Can also expand this as "In the constructed NCC instance, we can choose k actors with no two having a movie in common if and only if G has an ind. set of size k").

Proof:

1) Suppose G has an independent set S of size at least k. Then consider the set of actors corresponding to the vertices in S (Formally, consider $\{A_u | u \in S\}$. Since each pair of vertices u and v in S are disjoint, i.e., there is no edge incident on both u and v, hence the corresponding actors A_u and A_v do not have a movie in common. Thus, the set of actors described above is of size k with no two acting in the same movie.

2) Suppose we have a set of actors SA of size k with no two in the same movie. Then consider the set of vertices in G corresponding to the actors in SA (i.e., consider $S = \{u | A_u \in SA\}$. Since each pair of actors A_u and A_v in SA have not acted in any common movie M_e, the corresponding u and v must not constitute any edge e. Thus, S is an independent set of size k in G.

Thus we can reduce the independent set problem to the NCC problem in polynomial time. Since the independent set problem is known to be NP-Complete, NCC is NP-Hard.

Thus, NCC is NP-Complete.

Rubrics (15 points):

- 4 pts: Correctly shows in NP
- 5 pts: Correctly constructs and explains the reduction.
- 6 pts: Proof in both directions.

4. (**14 points**) Given an undirected graph $G = (V, E)$, and integer m, the *CLIQUE* problem is to decide if there is a subset $A \subseteq V$ of vertices (called as a *clique*) of size at least m such that for every pair of vertices $u, v \in A$ with $u \neq v$, (u, v) is an edge in E.

   The *HALF-CLIQUE* problem asks if G has a clique of size at least $\frac{|V|}{2}$ - Note that it does not take integer m as input unlike the *CLIQUE* problem. Show that *HALF-CLIQUE* is

   a) in NP.

   b) NP-hard by reducing from *CLIQUE*, which is known to be NP-complete.

   Solution:

   - To show the problem is in NP:
     (a) Certificate: A set of vertices
     (b) Certifier: Given a set of vertices A as the certificate, it is easy to verify that the two conditions listed in the question are satisfied in polynomial time. Thus *HALF-CLIQUE* is in NP.
   - To show the problem is NP-Hard we reduce from the known NP-Complete problem *CLIQUE*. Given a graph G and an integer m, the *CLIQUE* problem is to decide if the graph has a clique of size m.
     Given a *CLIQUE* instance $\langle G = (V, E), m \rangle$ we next reduce it to a *HALF-CLIQUE* instance. Depending on the values of parameter m and the number of vertices in G, we will have to think of three situations for this problem, which puts a constraint on the size of the *CLIQUE*.
     Construction and Claims:
     (a) If $m = \frac{|V|}{2}$, then we already have a *HALF-CLIQUE* instance, without additional constructions.
     (b) If $m < \frac{|V|}{2}$, then add $|V| - 2m$ new vertices to G. Then, add an edge between every distinct pair of new vertices. Also, an edge should be added between every new vertex and every existing vertex. The new graph G', which is a *HALF-CLIQUE* instance, has $2(|V| - m)$ vertices. The new graph has a clique of size at least $|V| - m$ if and only if G had a clique of size m.
     (c) If $m > \frac{|V|}{2}$, then add $2m - |V|$ new vertices to G and do not introduce any new edges. The new graph has $2m$ vertices. The new graph G' has a clique of size at least m if and only if G had a clique of size m.

     Proof:

     (a) Suppose G has a clique (of size m).
       – When $m = \frac{|V|}{2}$ it is trivial to see that G' also has a $HALF - CLIQUE$.
       – When $m < \frac{|V|}{2}$, from our construction we see that G' has a clique of size at least $|V| - m$ which is more than half the number of vertices. Thus, it satisfies the condition of $HALF - CLIQUE$.
       – Similarly with the remaining case, we have a clique of the same size in G' as per our construction. The size, i.e., m in this case, is already more than half the number of vertices present and thus has a $HALF - CLIQUE$.

     (b) Suppose G' has a half-clique, i.e., the clique of G' has at least half the number of vertices present in G'.
       – By construction, for the first case, G also has a clique present.
       – It follows trivially for the remaining two cases that if G' has a clique – upon removing the additional vertices and edges present through the construction on G', G ends up with a clique present.

Thus we can conclude that:

$$CLIQUE \leq_p HALF\text{-}CLIQUE$$

Thus *HALF-CLIQUE* is NP-Complete.

Rubrics (14 points):

- 4 pts: Showing the problem is in NP.
- 2 pt: Correct Reduction construction and explanation for $m = V/2$.
- 4 pts: Correct Reduction construction and explanation/proof for $m < V/2$.
- 4 pts: Correct Reduction construction and explanation/proof for $m > V/2$.

Ungraded problems

5. Consider the vertex cover problem that restricts the input graphs to be those where all vertices have even degree. Call this problem VC-EDG. Show that VC-EDG is NP-complete. (15pts)

   Hint: To show NP-hardness, reduce from VC, i.e., think of a construction where given the input of VC, i.e., a graph G and a number K, you can construct G', K', (where G' has even degrees for all its vertices), such that G has a vertex cover of size K if and only if G' has a vertex cover of size K'.

   Solution:

   First we need to prove even degree vertex cover problem is in NP.

   The certifier takes a subset of vertices as its certificate. It verifies that the subset is of size at most K and each of the original graph's edges has one of its endpoints in the given subset. Thus problem is in NP (the same certifier to the original vertex cover problem).

   Then we need to prove VC-EDG is NP-Hard.

   We claim that it is polynomial time reducible from the original vertex cover problem. Let $(G = (V, E), k)$ to be an input instance of Vertex Cover. Because each edge in E contributes a count of 1 to the degree of each of the vertices with which it connects, the sum of the degrees of the vertices is exactly $2|E|$, an even number. Hence, there is an even number of vertices in G that have odd degrees.

   Let U be the subset of vertices with odd degrees in G.

   Construct a new instance $(\bar{G} = (V_0, E_0), k + 2)$ of Vertex Cover, where $V_0 = V \cup \{x, y, z\}$ and $E_0 = E \cup \{(x, y), (y, z), (z, x)\} \cup \{(x, v) | v \in U\}$. That is, we make a triangle with three new vertices, and then connect one of them (say x) to all the vertices in U. The degree of every vertex in V_0 is now even. Since a vertex cover for a triangle is of (minimum) size 2, it can be shown that $\bar{G}$ has a vertex cover of size $k + 2$ if and only if G has a vertex cover of size k.

   Hence, vertex cover with only even degree vertices is NP Complete.

   Rubric (15 pts)

   - 6 pt: state the problem is in NP and mention how to verify it in polynomial time by checking if it is a valid vertex cover with even degree.
   - 3 pt: Choose an NP-complete problem and say that we need to prove that chosen problem $\leq_p$ even degree vertex cover. Other reduction can also get 3 points. Note: wrong direction will get 0.
   - 6 pt: Showing the polynomial reduction with details and its proof.

6. Consider the ***partial*** satisfiability problem, denoted as 3-Sat(α) defined with a fixed parameter α where $0 \leq \alpha \leq 1$. As input, we are given a collection of k clauses, each of which contains exactly three literals (i.e. the same input as the 3-SAT problem from lecture). The goal is 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 for $\alpha = 1$, we require all k clauses true, thus 3-Sat(1) is exactly the regular 3-SAT problem.

   Prove that 3-Sat($\frac{15}{16}$)is NP-complete. (20 points)

   Hint: If x, y, and z are variables, note that there are eight possible clauses containing them: $(x \vee y \vee z), (!x \vee y \vee z), (x \vee !y \vee z), (x \vee y \vee !z), (!x \vee !y \vee z), (!x \vee y \vee !z), (x \vee !y \vee !z), (!x \vee !y \vee !z)$ Think about how many of these are true for a given assignment of x, y, and z.

   Solution:

   To prove it's in NP: given a truth value assignment, we can count how many clauses are satisfied and compare it to $15k/16$.

   To prove it's NP-hard:
   We will show that 3-SAT $\leq_p$ 3-SAT(15/16). For each set of 8 original clauses, we add 3 NEW VARIABLES, and create all the 8 possible clauses on these 3 new variables. If the number of clauses is a multiple of 8, then we have created m new clauses for the m existing ones. Now, any assignment will satisfy only 7/8 of the new clauses by construction, so we can say, all of the original clauses in a valid solution can be satisfied if and only if 15/16 of all the new clauses can be satisfied (convince yourself by proving the previous statement both ways). Next, if the number of clauses is not a multiple of 8, say it is of the form $8a + b$ with $b < 8$. Then we follow the procedure above for the first a groups of 8 clauses and end, leading to a total of 16a + b clauses in the new instance. Remember that regardless of any assignment, exactly 7a of the new clauses can be satisfied. Then, it can be shown that one can achieve the factor of 15/16 satisfiability in the new instance if and only if all 8a + b of the original ones can be satisfied (using $b < 8$ and a bit of algebra).

   So, 3-Sat(15/16) is in NP and is NP-hard which makes it NP-Complete.

   Example: If original given formula is
   $(a \vee b \vee c) \wedge (!a \vee b \vee c) \wedge (a \vee !b \vee c) \wedge (a \vee b \vee !c) \wedge (!a \vee !b \vee c) \wedge (d \vee e \vee f) \wedge (g \vee !b \vee !c) \wedge (!a \vee !h \vee !c)$
   So we add our 8 new clauses so that formula now contains total 16 clauses.i.e.:
   $(a \vee b \vee c) \wedge (!a \vee b \vee c) \wedge (a \vee !b \vee c) \wedge (a \vee b \vee !c) \wedge (!a \vee !b \vee c) \wedge (d \vee e \vee f) \wedge (g \vee !b \vee !c) \wedge (!a \vee !h \vee !c) \wedge (x \vee y \vee z) \wedge (!x \vee y \vee z) \wedge (x \vee !y \vee z) \wedge (x \vee y \vee !z) \wedge (!x \vee !y \vee z) \wedge (!x \vee y \vee !z) \wedge (x \vee !y \vee !z) \wedge (!x \vee !y \vee !z)$
   If the original 8 can be satisfied with some assignment, then additionally setting any assignment to the new variables satisfies 7 of the new clauses, thus 15/16 of them in the new instance. If the original number of clauses in a formula is not a multiple of 8, say it was 13, then we still only add the above 8 new clauses making a total of 21. 15/16 fraction of that is $19\frac{11}{16}$, so satisfying 19 clauses doesn't get us above the required ratio, and we must satisfy 20. Since exactly 7 of the newly added 8 can be satisfied, the total of 20 can be achieved if and only if all the original 13 can be satisfied.

   Rubric (18 pts)

   - 4 pt: state the problem is in NP and mention how to verify it in polynomial time by checking how many clauses are satisfied.
   - 14 pt: Showing the polynomial reduction with details and its proof including:
     8 pt: instance construction 6 pt: using the construction proof the polynomial reduction.