Problem 2

I would like to select the best possible times for my office hours. I ask all n students in the class what times they can attend, and I obtain a list of times from each (assume that the times are discretized into hours). My goal is to select the minimum number of hours such that all students can attend at least one. Prove that this problem is NP-complete.

Problem is in NP

First of all, This problem is clearly in NP. This is because if given sets, representing hours when students can attend classes
we can verify in polynomial time if all the students have been covered by these sets.

Now we want to find a known NP problem and reduce it to this problem. This problem is a covering problem. We can try reducing set cover to it.

In set cover we need to choose mininum number of sets such that all features of a given collection are covered. Sets represent
gadgets that contain some features n'. We need to choose minimum number of sets such that all the featurs of the collection are covered.

size of collection set = N
sets s1,s2 .. sk such that |s1| = n1 , |s2| = n2, .. |sk| = nk minimum sets s(i) such that their sum is equal to N.

In this problem we need to cover all the students.We should construct an instance of given problem in which the students represent the collection of features, and the discrete hour preference units by different students are represented by sets.

Suppose there is a collection of N elements and sets s1,s2..sk where each set is s(i) and i varies from 1 to k contains
some subset of elements of N. Each set s contains repectively s[i] elements.
for each element of collection N we have a student N[i] from the list of students. Also for each set s we have a discrete
hour unit. So we have a total of N students and 'k' discrete hour units. For each k discrete unit we have some number of students that prefer attending the lecture in that time unit.

NP-Complete (Set Cover <p Given problem)

The following claim will establish that Set Cover <p given problem and hence will conclude the proof that the given is NP-complete. Given how closely our construction of the instance shadows the original Set Cover instance, the proof is completely straightforward.

In the instance constructed, the number of hours chosen by professor would be at most k if and only if there is a set cover of size at most k.

Proof

First, suppose that the instance contains a set cover C of size at most k. Then consider the corresponding collection of sets S representing hours in the instance of given problem. Each student will be present in atleast one of the sets corresponding to the hours.The hour corresponding to the set is preferred by students belonging to the set. Since set cover covers all the elments in the given collection, here the elements of the collection are students , since set cover also chooses the smallest number of sets that cover all the elements, by the reduction from set cover to the given problem we will have the mininum number of hours that covers all the students.

Conversely,
Let the minimum number of hours that all students can attend atleast 1 class be k. Consider the sets corresponding to each discete time unit.For each discrete hour unit atleast in the preference list of studetns at least one student prefers attending the class in that hour. Given the prefernce list containing containing k hours we cover all students since we have more than 1 or more student having a preference for that hour.

Hence C is a set cover

Problem 3

Scrabble: We’re given a dictionary of n strings, each with at most L characters. Each string in the dictionary has an associated number of points. We’re also given a set of available characters S (which may include repeats). We’d like to determine if we can select at most k strings from the dictionary which can be simultaneously constructed from S and are worth at least P points total. For example, we may have characters S = A, A, C, C, R, T . We can form CAT and CAR simultaneously, but we cannot form CAT and RAT simultaneously (because there is only one T). Prove that this problem is NP-Complete.

We are given a dictionary of N strings each with at most length L. Each string has an associated number of points. We also
have a set of available characters S.

Problem in NP

Now given the set of selected k strings we can verify in polynomial time if we can split each word into characters simulataneously without any conflict. Since this is a polynomial operation the problem is in NP.

We now show that Independent Set <p given problem.

Given a graph G and a number k , we create an equivalent formulation that depicts the given problem. The words correspond to the
nodes of the graph and characters common between two words correspond to the edges. We need to show two things that the
given problem and independent set problem are indeed equivalent.

NP Complete

If there are k words that do not conflict among themselves then k words corresponding to the nodes form an independent set.
This is true and the character between is a character that both of them want to use.

We can also show this in the reverse direction.

If there is an independent set of size k then the k words corresponding to these nodes form a set of k words that do not compete
for characters in set S.

Hence the problem is NP-Complete.

Since independent set given us maximum number of elments that are non conflicting in nature. The sum of the weigths associated
with these words would be the maximum. We can assign a weight of 1 to each word node or any aribitray value P(i) for the i th node
in either case the value of P would be maximized.

In the Bipartite Directed Hamiltonian Cycle (BDHC) problem, we are given a bipartite directed graph G = (V, E). We want to determine if there is a cycle which visits every node exactly once.

Part A

Show that BDHC ∈ NP.

Given a bipartite directed graph it is easy to verify in polynomial time if BDHC contain a hamiltonion cycle. So the problem is in NP

Part B

Give a polynomial-time reduction from BDHC to Directed Hamiltonian Cycle (DHC). Explain why this reduction does not help you prove that BDHC is NP-complete.

A bipartitie directed hamilonian cycle is a graph which already has a hamiltonian cycle. Infact BDHC is a restricted version of DHC having only even number of nodes. We can reduce BDHC to a DHC by removing half of the nodes. or merging some of the nodes together to get the original nodes in DHC.

The reason this reduction does not prove BDHC is NP complete because we need to show that BDHC is atleast hard as some NP-complete hard problem. By saying BDHC is reducible to DHC we are saying that we can convert BDHC to DHC in polynomial number of steps.
If we assume BDHC is polynomial time solvable then this polynomial reduction implies that even DHC is polynomial time solvable
which is not the case as we clearly know that DHC is NP-Complete.

Part C

Give a polynomial-time reduction from DHC to BDHC.

We now show that DHC <p BDHC.

Given a graph G , we create an equivalent formulation that depicts the given problem. The nodes in the graph is equal to the nodes in the original graph, but if there are odd number of nodes in the original graph we split any node in the odd length cycle into 2 nodes N(in) and N(out) we connect N (in) to N(out). This new formulation has the same properties as the original DHC. It infact is a DHC. BDHC and DHC are indeed equivalent.

NP Complete

If there is a BDHC in other words a directed hamiltonian cycle in a bipartite graph then cycle in the bipartite graph corresponds to a directed hamiltonian cycle.

We can also show this in the reverse direction.

If there is an direct hamiltonian cycle in a graph then if the graph contains even number of nodes then the graph represents a directed hamiltonian cycle. If the number of nodes are not even then the graph cab be transformed in a bipartite graph containing even number of nodes by splitting any odd number node and splitting it into 2 nodes and coloring them separately. This new transportation is always a bipartite graph.

Hence this way we can show DHC <p BDHC

Part D

Explain how to convert your BDHC answer into an answer for DHC.

A bipartitie directed hamilonian cycle is a graph which already has a hamiltonian cycle. Infact BDHC is a restricted version of DHC having only even number of nodes. We can reduce BDHC to a DHC by removing half of the nodes. or merging some of the nodes together to get the original nodes in DHC.

Part E

Briefly justify why your reduction is correct. That is, explain why there is always an answer to your BDHC instance when the DHC instance has a solution, and vice versa.

If there is a BDHC in other words a directed hamiltonian cycle in a bipartite graph then cycle in the bipartite graph corresponds to a directed hamiltonian cycle.

We can also show this in the reverse direction.

If there is an direct hamiltonian cycle in a graph then if the graph contains even number of nodes then the graph represents a directed hamiltonian cycle. If the number of nodes are not even then the graph cab be transformed in a bipartite graph containing even number of nodes by splitting any odd number node and splitting it into 2 nodes and coloring them separately. This new transportation is always a bipartite graph.

Problem 4

We define the following problem, which we will call 3-Sat(α). We are given a collection of m clauses, each of which contains exactly three literals (variables), and asked to determine whether there is an assignment of true/false values to the literals such that at least αm clauses will be true. Note that 3-Sat(1) is exactly the 3-SAT problem.

Part A

(a) Give an O(mn)-time algorithm that outputs a satisfying assignment for 3-Sat(1/2).

(b) (20) Prove that 3-Sat(15/16) is NP-Complete.

Hint: Consider the collection of all possible 3-Sat clauses on three literals. Any truth assignment to the literals will make exactly seven of these clauses true (and one false).

Part A

For 3-Sat(1/2) to be true we need a satifying assignment such that half of the clauses are true.

```
for every literal j in the set of literals:
 for every clause i in the set contaiing M clause:
   if literal i is positive in clause j we might set it to true
   if literal i is negative in clause j we might set it to false
```
if we have contradicting values for literal j do not set it else set to the value that is consistent

This algo will make sure that at least half of the clauses are true

This will give us an O(MN) algorithm for assigning values to the clauses.

Part B

The problem is in NP, given an assignment for literals as we can verify in polynomial time if there is a solution that satifies
all the clauses.

NP-Complete

We need to show thta 3-SAT <p 3-SAT(15/16)

Consider the collection of all possible 3-Sat clauses on three literals. Any truth assignment to the literals will make exactly seven of these clauses true (and one false).

Now given a 3-SAT assignment for each assignment in 3-SAT we can create a corresponding assignment, but this time we include
another variable, we call this variable the oscillating variable, we have full control over it and we can set it true any time
this variable is like a dummy variable. It also has another property since we can assign any value to it we can exchange
its value with any of the assignment value belonging to any of the vairables in any of the clauses.

In other words we have a new parallel assignment which instead of 3 variables has 4 variables, out which we control what values are assigned to the 4 th variable. Now we assign a value of 0 to the

dummy variable in clause number 1. We assign a value of 1 ot the next 7 clauses, effectively making the next 7 out of 8 clauses true.

Since we have 4 variables now we effectively have another 24 clauses for which we need to perform an assignment. Since our dummy varibale has a property that we can replace with any assigned varibles, it can effectively take the value of any variable it wants
and replace its value with its own value. The property allows us to choose values such that only 1 caluse in the remaining 24 clauses is false, yielding a total of 30 true clauses out of 32 clauses which is effectively 15/16 of the total number of clauses.

This transformation in the reverse direction involves getting rid of the extra varible. When get rid of the extra variable we are
left with m clauses and each clause is a combination of 3 literals. Hence we get the 3 SAT problem back.

Hence by this reduction we can deduce that 3-SAT(15/16) is NP-Complete.
