CSOR 4246 Algorithms for Data Science (Fall 2022) Problem Set #4t

Xinhao Li - xl2778@columbia.edu

2022/12/06

Problem 1

For this problem, we are asked to show that decision version of MAXSAT(D) is NP-complete (NPC). So the overall steps to prove that MAXSAT(D) is NPC should follows:

1. Prove that MAXSAT(D) can verify any inputs in polynomial time, which means that MAXSAT(D) belongs to NP class.
2. Show that MAXSAT(D) is at least as hard as any known NPC problems.

Let define the decision version of MAXMAT first.

Decision version of the problem

Suppose that an assignment of literals and the maximum number of satisfied clauses g are given to the certifier. We can quickly check how many satisfied clauses can give and consequently determine whether this value is greater, equal, or less than g. Since the formula ϕ has m clauses over n variables, the total running time is O(nm), which is polynomial. So MAXSAT(D) belongs to the NP class.

The inputs to the two problems

- The inputs to the SAT problem are a CNF formula ϕ with n literals and m clauses.
- The inputs to the MAXSAT(D) are a CNF formula ϕ' with n' literals and m' clauses as well as an integer g.

The Reduction transformation and the polynomial time requirements

For this question, we are going to show the reduction from SAT to MAXSAT(D). To transform input from SAT to MAXSAT(D), we can take the inputs from SAT problem and assign those to MAXSAT(D) problem. And then for the integer that MAXSAT(D) required, we will assign: $g = m$.
This transformation requires only $O(1)$ time since the only thing we did is just to re-assign

different values from SAT to MAXSAT(D).
As for the output, we can say that:

- If MAXSAT(D) returns NO then return NO for SAT. Since we assigned $g = m$, returning No means that the CNF formula has number of satisfied clauses less than g. And to be a SAT, each clause has to be true (requires g clauses).
- If MAXSAT(D) returns an assignment of literals, then return the same assignment for SAT. This means that $g = m$, so any returned assignment of literals for MAXSAT(D) should also be the solution for SAT.

Prove equivalence of the original and the reduced instances.

- Forward direction: $\phi \to \phi'$
  - For any given **valid** assignment of literals for SAT, ϕ is satisfied which means that m clauses are set to be True. Since $g = m$ in MAXSAT(D) problem, so at least m clauses need to be satisfied in ϕ'. Thus, MAXSAT(D) will also be satisfied.
  - For the cases that the given assignment of literals does not satisfy SAT which means that there are less than m clauses, then MAXSAT(D) will not be satisfied either.
- Backward direction: $\phi' \to \phi$
  - For any given **valid** assignment of literals for MAXSAT(D) (**valid** here means that will maximize the number of satisfied clauses), ϕ' is satisfied and we will have at least $g = m$ clauses be True.
  - The condition that m clauses to be True is exactly the requirement to be SAT.

This proves that $\phi \rightleftarrows \phi'$. In conclusion, we proved that MAXSAT(D) is at least as hard as SAT and it is also NP-complete.
Basically, MAXSAT(D) can be seen as a generalization of SAT since if we know one assignment of literals that maximize the number of satisfied clauses correctly, we can simply compare this number with the total number of clauses in the CNF formula. If they are equal, this means that this CNF formula is satisfiable and if they are not equal, this CNF formula is not satisfiable. So simply known the results of MAXSAT(D) will let us know if the formula is SAT or not.

Problem 2

Description

In order to design an algorithm B that takes as input a formula ϕ and returns a satisfying truth assignment for ϕ if one exists, or no otherwise. We can first run algorithm A to first check if ϕ is satisfiable. If it is satisfiable, then we can start to add y_i or $\overline{y_i}$ to the tail of ϕ and the logic connection between ϕ with y_i or $\overline{y_i}$ is AND.

Algorithm B

Input: ϕ with m clauses over n variables $(y_1, y_2, ..., y_n)$

```
1. Run algorithm A to check if $\phi$ is satisfiable
   If False; then return False
   else go to the next step

2. Let's set $\phi' = \phi$ and initialize a list $ans$ with n elements inside and all elements are
   assigned to 1.

3. for $i$ in range(len($ans$)):
   $\phi'_t = \phi'$
   $\phi' = \phi' \wedge y_i$
   Run algorithm A to check if $\phi'$ is satisfiable
      (a) If return True; continue to check the second element in $ans$
      (b) If return False; set V[i] = 0 and $\phi' = \phi'_t \wedge \overline{y_i}$

4. return $ans$
```

The ans should contain a satisfiable assignment of $y_1, y_2, ..., y_n$ for CNF formula ϕ.

Time complexity

Overall, we have checked n times using polynomial time algorithm A, so the total running time for this new designed algorithm B is also polynomial

Proof of correctness

Since ϕ has been checked first that it is satisfiable, so the progress that we add y_i or $\overline{y_i}$ to the tail of ϕ is to make sure that the new added clause is satisfiable and also this assignment of y_i or $\overline{y_i}$ also makes the initial CNF formula ϕ satisfiable.

Problem 3

First, again, to prove that this problem is NPC, we first need to show that this problem is in NP and then show it is at least as hard as any known NPC problems. Let me state the decision version of the problem first.

Decision version of the problem

Suppose a matrix A with size $m \times n$ with $A_{ij} = 1$ iff customer i has bought product j, else 0, and an integer k. Then the certifier can output YES or NO in polynomial time if there is an subset of orthogonal customers with size at least k. Since there are m customers and n products, so there are at most $O(m^2)$ pairs to check. Besides, each check takes $O(n)$ time if there exists a pair of customers have purchased the same thing. So overall, the time complexity will be $O(nm^2)$ which is clear a polynomial time algorithm.

Then the next step is to prove it is at least as hard as any known NPC problems. Here, we will reduce from the Independent Set Problem (IS), which is already shown in the lecture that it is NPC.

The inputs to the two problems

- The inputs to IS problem: a graph $G = (V, E)$ with n vertices and m edges, and an integer k
- The inputs to X problem: A binary $m \times n$ matrix A with $A_{ij} = 1$ iff customer i has bought product j, else 0

The Reduction transformation and the polynomial time requirements

As mentioned before, we will convert the inputs from IS problem to X problem. IS problem takes (G, k) with vertices $\{v_1, v_2, ..., v_k\}$ and edges $\{e_1, v_e, ..., e_l\}$ as the inputs. We can define a matrix A with size $k \times (k + l)$ where $\{c_1, c_2, ..., c_k\}$ are the customers and $\{p_1, p_2, ..., p_{k+l}\}$ are the products. Then we can set that:

- A[i, i] = 1 which means that customer c_i purchased product p_i. This is used to make sure that each customer has at least purchased one product since if any customer dose not buy anything in the store, then it will definitely not be orthogonal with any other customers. So this customer c_i should either be directly added into the independent set or just remove from the dataset since it does not contribute anything to this problem.
- A[i, k+j] = 1 which means that customer c_i purchased product p_{k+j} iff the edge e_j is incident on the vertex v_i. This step is used to make sure that any pair of customers that have purchased the same product will have an edge between them in the IS problem and then if there is an edge between two vertices, these two vertices can not be in the independent set at the same time.

This reduction outputs A matrix and the number k as the the inputs to X problem. This transformation is obviously in polynomial time since $O(n \times (n+m))$ time is required to fill in matrix A.

Prove equivalence of the original and the reduced instances.

- Forward direction: $IS \to X$
  - Suppose the IS has an independent set of size k, which is $\{v_1, v_2, ..., v_k\}$. Then we will also have a set a customers $\{c_1, c_2, ..., c_k\}$ that does not have any pair of customers are orthogonal. This is because if any two customers in $\{c_1, c_2, ..., c_k\}$ have purchased the same product, then there must be an edge between these two customers. We also know that $\{v_1, v_2, ..., v_k\}$ is an independent set so, no such edge is allowed to exist.
- Backward direction: $X \to IS$
  - Suppose we have a set of customers $\{c_1, c_2, ..., c_k\}$ that does not have a pair of customers purchase the same product. Then $\{v_1, v_2, ..., v_k\}$ should be an independent set. This is because if $\{v_1, v_2, ..., v_k\}$ is not an independent set which means that there is an edge between any two vertices in set $\{v_1, v_2, ..., v_k\}$, then according to our reduction transformation, there should also be a pair of customers that have purchased the same product. This is definitely can not be the case. So from $X \to IS$ is also been proved.

Therefore, we have proved that the decision version of X problem can be certified in polynomial time and X problem is at least as hard as IS problem (NPC). So X problem is NPC.

Problem 4

Input: A set F of m fire stations, a set H of n homes, and a cost a_{ij} for each fire station i and home j, a cost f_i for operating each fire station i, and a cost K.
Output: YES if there exists a subset F' $\subseteq$ F of fire stations to operate and an assignment of homes to fire stations in F' such that the sum of the following two costs is less than or equal to K: (i) the total cost of assigning homes to operating fire stations in F'; and (ii) the total cost of operating the fire stations in F'. Otherwise, the output is NO.
It is in NP because given a proposed solution (subset of fire stations F' and an assignment of homes to fire stations), we can verify in polynomial time whether the solution satisfies the constraints and whether the sum of the costs is less than or equal to K.

It is NP-hard because we can reduce an instance of the NP-complete problem Set Cover to an instance of the decision version of Y as follows:

Given a set X with n elements and a collection S = $\{S_1, S_2, ..., S_m\}$ of subsets of X, we construct a corresponding instance of the decision version of Y as follows:

- For each element x in X, we create a corresponding home h.
- For each subset S_i in S, we create a corresponding fire station f.
- For each element x in X and each subset S_i in S that contains x, we set the cost a_{ij} to be 1 if the home corresponding to x is assigned to the fire station corresponding to S_i, and 0 otherwise.
- For each fire station f, we set the cost f_i to be the number of elements in the corresponding subset S_i.
- We set the cost K to be the size of the smallest subset in S.

It is easy to see that this reduction can be done in polynomial time.

We claim that the instance of Set Cover has a solution if and only if the corresponding instance of the decision version of Y has a solution. First, suppose that the instance of Set Cover has a solution, i.e. a subset of subsets S' $\subseteq$ S such that the union of subsets in S' covers all elements in X. We construct the corresponding solution to the instance of the decision version of Y as follows: We choose the fire stations corresponding to the subsets in S' as the subset F' of fire stations to operate. We assign each home corresponding to an element in X to the fire station corresponding to the subset in S' that contains the element. It is easy to see that this solution satisfies the constraints and the sum of the costs is less than or equal to K because the union of subsets in S' covers all elements in X, and the cost of operating each fire station is equal to the number of elements in the corresponding subset in S'. Next, suppose that the instance of the decision version of Y has a solution, i.e. a subset F' of fire stations to operate and an assignment of homes to fire stations in F' such that the sum of the costs is less than or equal to K. We construct the corresponding solution to the instance of Set Cover as follows: We choose the subsets corresponding to the fire stations in F' as the subset S' of subsets.

Appendix

References

- Q-1

Useful discussions with

- Hongyi Hue (hh2955)