Lab11-NP Reduction

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

∗ If there is any problem, please contact TA Shuodian Yu.
∗ Name:Zehao Wang Student ID:518021910976 Email: davidwang200099@sjtu.edu.cn

1. What is the "certificate" and "certifier" for the following problems?

   (a) *ZERO-ONE INTEGER PROGRAMMING*: Given an integer $m \times n$ matrix A and an integer m-vector b, is there an integer n-vector x with elements in the set $\{0, 1\}$ such that $Ax \leq b$.

   (b) *SET PACKING*: Given a finite set U, a positive integer k and several subsets $U_1, U_2, \ldots, U_m$ of U, is there k or more subsets which are disjoint with each other?

   (c) *STEINER TREE IN GRAPHS*: Given a graph $G = (V, E)$, a weight $w(e) \in Z_0^+$ for each $e \in E$, a subset $R \subset V$, and a positive integer bound B, is there a subtree of G that includes all the vertices of R and such that the sum of the weights of the edges in the subtree is no more than B.

Solution.

(a) Certificate is any n-vector x_0 with elements in the set $\{0, 1\}$ such that $Ax_0 \leq b$. Certifier is as follows.

Algorithm 1: certifier for the first subquestion

Input: An $m \times n$ array A
Output: A bool variable $flag$

```
for i = 1 to m do
    if A[i]x > b[i] then
        return False;
return True
```

(b) Certificate is a collection C of the given subsets which satisfies that it has k or more members and the members are disjoint. Certifier is as follows.

Algorithm 2: certifier for the second subquestion

Input: A finite set U, an integer number k, and a collection C of U_i's
Output: A bool variable $flag$

```
if k = 1 then
    return True;
num ← number of members in C;
if num < k then
    return False;
for i = 1 to |C| do
    for j = 2 to |C| do
        if C[i] has the same element as C[j] then
            return False;
return True;
```

(c) Certificate is a subtree T_0 that satisfies the requirements above. Certifier is as follows.

Algorithm 3: certifier for the third subquestion

Input: A graph G, a subtree T of G, a subset R of V, an integer B.
Output: A bool variable $flag$

```
for node v ∈ R do
    if v ∉ V then
        return False;
weight ← 0;
for edge e ∈ T do
    weight ← weight + w(e);
if weight > B then
    return False;
return True;
```

2. Algorithm class is a democratic class. Denote class as a finite set S containing every students. Now students decided to raise a student union $S' \subseteq S$ with $|S'| \leq K$.

   As for the members of the union, there are many different opinions. An opinion is a set $S_o \subseteq S$. Note that number of opinions has nothing to do with number of students.

   The question is whether there exists such student union $S' \subseteq S$ with $|S'| \leq K$, that S' contains at least one element from each opinion. We call this problem *ELECTION* problem, prove that it is NP-complete.

   Solution.

   - First we need to prove that it is NP.

     To prove that it is NP, we can find a polynomial-time certifier $C(s, t)$ for any given certificate S.

     $C(S, t)$ can simply check each option S_o to find out whether S contains at least one element in S_o's.

     Assume that there are p options. Each option has q elements. We can know that this algorith has $O(Kpq)$ complexity, which satisfies the time complexity requirement.

   - Then we need to prove that it is NP-Complete by proving *3-SAT* $\leq_p$ *ELECTION*.

     We assume that there are n boolean variables $x_1, x_2, \cdots, x_n$. There are m clause to satisfy. We want to find out if there is a truth assignment for each variable x_i. We can transform this problem into *ELECTION* in this way:

     There are n students in a class. We want to choose a student union from these n students. There are m options each of which has 3 elements. We want to choose not more than n students.

     If we can solve the transformed question above, we can let $x_i = 1$ if the i-th student is chosen and $x_i = 0$ if he or she is not. Therefore, a typical NP-complete problem: *3-SAT* can be reduced to a *ELECTION* problem. *ELECTION* is an NP-complete problem.

   □

3. Not-All-Equal Satisfiability (NAE-SAT) is an extension of SAT where every clause has at least one true literal and at least one false one. NAE-3-SAT is the special case where each clause has exactly 3 literals. Prove that NAE-3-SAT is NP-complete. (Hint : reduce 3-SAT to NAE-k-SAT for some $k > 3$ at first)

   Proof.

   - Any truth assignment of a clause for NAE-3-SAT is a certificate. And certifier can check this assignment within polynomial time. Therefore NAE-3-SAT is an NP problem.
   - Then we try to reduce a 3-SAT problem to a NAE-4-SAT problem.

     3-SAT requires that in each sub-clause at least one variable should be assigned true. NAE-4-SAT requires that in each sub-clause at least one variable assigned true and another assigned false. This means that they are assigned different truth value. Therefore we can make use of these properties.

     For any sub-caluse $(x \vee y \vee z)$, we can transform it into $(x' \vee y' \vee z' \vee w)$. Let $x = False$ or $x = True$ when $x' = w$ or $x' \neq w$. So it is with y and z. Here we successfully reduce 3-SAT to NAE-4-SAT. NAE-4-SAT proves to be NP-Complete.

     Then we try to reduce NAE-4-SAT to NAE-3-SAT. For any sub-clause for NAE-4-SAT, e.g. $(x \vee y \vee z \vee w)$, we can transform it into $(x \vee y \vee t) \wedge (z \vee w \vee \neg t)$. If x, y, z, w are all true, we can see the sub-clause for NAE-3-SAT is not satisfiable. Therefore we can use NAE-3-SAT to solve NAE-4-SAT.

     Therefore we prove that 3-SAT$<_p$ NAE-4-SAT $<_p$ NAE-3-SAT.

     Considering 3-SAT is NP-complete, we can know NAE-3-SAT is also NP-complete.

   □

4. In the Lab10, we have introduced Minimum Constraint Data Retrieval Problem (MCDR). Prove that MCDR (Version 1 or 2) is NP-complete. (Hint : reduce from VERTEX-COVER or 3-SAT)

   Proof.
   - We first prove that it is NP. A certificate for this problem is just a downloading schedule. A certifier is just to check whether with the schedule the desired data can be downloaded within h switches or within a duration of t.
   - Then we try to prove that it is NP-complete by reducing VERTEX-COVER to MCDR. Given a graph G, we want to know whether there is a VERTEX-COVER of k vertexes. Then in MCDR, we can build $|V|$ channels for the $|V|$ vertexes, and then build k extra channels for the limitation that the VERTEX-COVER should contain k vertexes.

     Let d_0 be the maximal degree of the vertexes. And the broad cast cycle time for each channel is $d_0 + 3$.

     For each edge e_{ij}, add a data item to the i-th and j-th channel from the third time unit.

     As for the extra k channels, add one piece of data to each of them at the first time unit.

     All the data added form a data set D_q to be downloaded.

     For any schedule ,if the data in the i-th channel ($i < |V|$) are downloaded, it means cover all the edges connecting the i-th vertex. If the data in the $n + i$-th ($i < k$) channel is downloaded, it means the VERTEX-COVER assumes a vertex.

     Therefore we have reduced VERTEX-COVER to MCDR. Therefore MCDR is an NP-complete problem.

   □

Remark: Please include your .pdf, .tex files for uploading with standard file names.