Junhao Zhang “Freddie”

USC ID: XXXX-XXXX-XXXX

CSCI 570 – HW6

01/25/2016

* **[Question 1]**

We can use a reduction from 3-SAT (which is a well-known NP-Complete question) to prove that Integer Programming is NP-hard.

Let’s have integer variables:

And then we can convert any 3-SAT instances by doing the following:

* For each literal *i*, if negation then we have , otherwise we have solely .
* After having done that above, then for each clause (which should have 3 literals), we examinate is the sum of the clause greater than 0.

For example, this 3-SAT instance:

It would be converted to:

This way, we can convert all instances of 3-SAT to Integer Programming problems. Since 3-SAT is NP-Complete, that means all NP (includes 3-SAT itself) can be reduced to Integer Programming, which further proves that Integer Programming is NP-Hard.

* **[Question 2]**

We can use

* **[Question 3]**

We can convert any instances of such kind of problem into independent set problems by doing the following:

* Let each course be a vertex, and we make overlapped courses adjacent by connecting them by edges. This way we will construct a graph *G*.
* Then, we can find independent sets in the *G*. If there exist an independent set that has size more or equal to *K*, then students can choose *K* courses and they don’t overlap.

Because of above, we showed independent set problem can be reduced to such kind of problem, which further proved this kind of problem is NP-hard.

Moreover, the solution of the problem can be verified in polynomial time, by checking the size of the independent set is larger or equal to K, and the courses don’t overlap. Therefore, this kind of problem itself is NP. Thus, this kind of problem is NP-Complete.

* **[Question 4]**

We can