CS 599: Algorithms **(Due: December 4th, 2020)**

Submission Homework 9

Instructor: Prof. Prasad Tadepalli *Name:* Rashmi Jadhav, *Student ID:* 934-069-574

1. 0-1 Integer Programming

Inequalities: $a_{1,1}X_1 + ... + a_{1,n}X_n \geq b_1$; $a_{m,1}X_1 + ... + a_{m,n}X_n \geq b_m$
Decision problem: Is there a set of values for variables $X_1, X_2, ..., X_n$ such that all the inequalities hold?
To prove that 0-1 integer programming problem is NP-complete, we first show that the problem is in NP and then we move on to the reduction step to prove its NP-completeness.

Part 1: 0-1 integer programming problem $\in$ NP.
For the given problem we can come up with the following non-deterministic polynomial time algorithm:

Guess $X_1 = T/F$
Guess $X_2 = T/F$
...
...
...
Guess $X_n = T/F$

We call this guess a certificate. Once we have a certificate, we need to verify whether the inequalities hold with this certificate.

For the verifier, computation of:

$a_{1,1}X_1 + a_{1,2}X_2 + ... + a_{1,n}X_n$
$a_{2,1}X_1 + a_{2,2}X_2 + ... + a_{2,n}X_n$
...
...
$a_{m,1}X_1 + a_{m,2}X_2 + ... + a_{m,n}X_n$

would take $O(nm)$ time and for each of the m inequality comparisons we would take $O(m)$ time.

This shows verifier takes polynomial time to verify the certificate.
Thus, 0-1 integer programming problem $\in$ NP.

Part 2: 0-1 integer programming problem is NP hard.
We now reduce 3-CNF SAT to 0-1 Integer Programming.
For 3-CNF SAT formula ϕ with v variables and c clauses, we can construct $a_{i,j}$'s as:

$a_{i,j} = 1$ if variable j occurs without negation in clause i
$a_{i,j} = -1$ if variable j occurs with negation in clause i and
$a_{i,j} = 0$ otherwise

Further, we can construct b_i's as:

$b_i = 1 - number\ of\ negated\ literals\ in\ clause\ i$
$b_i = 1 - \sum_{j=1}^{v} min(0, a_{i,j})$

The $a_{i,j}$'s and b_i's can be constructed in $O(cl)$ where c is number of clauses and l is number of literals. This shows that the reduction can be done in polynomial time.
We now need to prove that the existence of certificate x that satisfies the inequalities given the formula ϕ:
For example: $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4)$
Representing the inequalities in a matrix form from the constructed $a_{i,j}$ and b_i, the reduced instance looks like:

$$\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

We can see that $x_1 = T, x_2 = T, x_3 = F, x_4 = T$ satisfies ϕ and that

$$\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

Thus, ϕ is satisfiable and since 3-SAT is a known NP-complete problem which can be reduced to 0-1 integer programming in polynomial time, we say 0-1 integer programming is NP-complete.

2. Clique Decision Problem

Undirected graph G with V vertices
Decision Problem: Does G have a clique of size p?(subgraph of p vertices in which there's an edge between each pair of vertices)
To prove that clique decision problem is NP-complete, we first show that the problem is in NP and then we move on to the reduction step to prove its NP-completeness.

Part 1: clique decision problem $\in$ NP.
For the given problem we can come up with the following non-deterministic polynomial time algorithm:

Guess $V' \subseteq V$ of size p which would be our certificate.

For the verifier we need to verify the certificate by checking if each pair of vertices in V' has an edge between them.

For p vertices, we need to check for an edge between a total of $p(p-1)/2$ pairs. This can be checked with DFS/BFS graph traversal techniques taking $O(p^2)$ time complexity.
This shows that the verifier takes polynomial time to verify the certificate.
Thus, clique decision problem $\in$ NP.

Part 2: clique decision problem is NP hard.
We now reduce independent set problem to clique decision problem.
The independent set problem requires us to find out if there's a subset of G of size p for which there's no edge between any of these vertices in the subgraph. Comparing the two problems, independent set wants vertices where **none** are connected whereas clique wants vertices where **all** are connected. We need to convert a **none** problem to **all** problem.

Construct a complement graph G' by removing all the edges in G and by adding all the edges between vertices that don't have an edge in G. For an adjacency matrix, this would just be flipping all the $(V \times E)$ bits $\implies$ polynomial time.

Now we should show that a solution to p-clique exists iff solution to p-independent set exists.

By definition, the independent set has no edges between any vertices. These will all be edges in G' and therefore they will form a clique of size p. Hence, given a graph G that has an independent set of size p, G' has a clique of size p.

By definition, the clique will have an edge between every vertex. None of these vertices will therefore be connected in G, so we have an independent set. Thus, given G' that has clique of size p, G has an independent set of size p.
We have reduced independent set problem to clique decision problem in polynomial time and hence clique decision problem is NP-complete.

3. Directed Hamiltonian Path

Directed graph G
Decision Problem: Does G have a directed hamiltonian path from some node **s** to some other node **t** that visits every other node exactly once?
To prove that directed Hamiltonian path is NP-complete, we first show that the problem is in NP and then we move on to the reduction step to prove its NP-completeness.

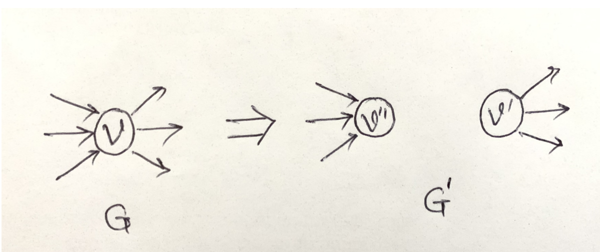
Part 1: directed Hamiltonian path $\in$ NP.
For the given problem we can come up with the following non-deterministic polynomial time algorithm:

Guess $E' \subseteq E$ to be the set of edges in Hamiltonian path. This is our certificate.

For the verifier we need to verify the certificate by checking if E' visits every vertex exactly once from s to t. This can be done using a graph traversal technique with time complexity $O(V + E)$
This shows that the verifier takes polynomial time to verify the certificate.
Thus, directed Hamiltonian path problem $\in$ NP.

Part 2: directed Hamiltonian path is NP hard.
We now reduce directed Hamiltonian cycle problem to directed Hamiltonian path problem.
The directed Hamiltonian cycle requires us to find out if there's a cycle in G that visits each vertex of G exactly once.

Given an instance of Hamiltonian cycle G, choose an arbitrary vertex v and split it into two vertices to get G' as follows:



Now, any Hamiltonian path must start at v' and end at v''.
We need to show that G' has a Hamiltonian Path iff G has a Hamiltonian cycle.

If G' has a Hamiltonian Path, then the same ordering of vertices (after we glue v' and v'' back together) is a Hamiltonian cycle in G.

If G has a Hamiltonian Cycle, then the same ordering of vertices is a Hamiltonian path of G' if we split up v into v' and v''.

We have reduced dHamCycle to dHamPath in polynomial time and hence dHamPath problem is NP-complete.

4. 0-1 Knapsack Decision Problem

Given: a set of n items of values $V_1, V_2, ..., V_n$ and weights $W_1, W_2, ..., W_n$ and a capacity W.
Decision Problem: Is there a selection of items which fit into the knapsack and has a value at least T? To prove that 0-1 Knapsack Decision Problem is NP-complete, we first show that the problem is in NP and then we move on to the reduction step to prove its NP-completeness.

Part 1: 0-1 Knapsack Decision Problem $\in$ NP.
For the given problem we can come up with the following non-deterministic polynomial time algorithm:
Guess subset $S \subseteq n\ items$. This subset is the certificate.
To verify the certificate, we calculate the total weight and total value to meet the constraints total weight $\leq W$ and total value $\geq T$. The calculation would take $O(n)$ time.
This shows that the verifier takes polynomial time to verify the certificate.
Thus, 0-1 Knapsack Decision Problem $\in$ NP.

Part 2: 0-1 Knapsack Decision Problem is NP-hard.
We now reduce the subset sum problem to 0-1 Knapsack decision problem.
The subset sum problem wants us to find out if there is a subset of input numbers $s_1, s_2, ..., s_n$ with total sum T.
To reduce, we create such a Knapsack problem that:
$V_i = W_i = s_i$ and $W = T$
The Yes/No answer to the new problem corresponds to the same answer to the original problem. Now we prove that the two problems are equivalent:
$i.\ e.,\ \sum_{i \in S} s_i = T\ \ iff$
$\sum_{i \in S} W_i \leq W \iff \sum_{i \in S} s_i \leq T$
$\sum_{i \in S} V_i \geq T \iff \sum_{i \in S} s_i \geq T$
Suppose we have a Yes answer to the new problem, it means we can find such a subset $S \subseteq [1, 2, ..., n]$ that satisfies the iff constraints above. Then this subset S is also a solution to $\sum_{i \in S} s_i = T$. So we must also have a Yes answer to the original problem.
Conversely, suppose we have a No answer, it means there is no subset S that satisfies the iff constraints above. So, of course, the answer to the original problem must also be No.
We have reduced subset sum problem to 0-1 Knapsack decision problem in polynomial time and hence 0-1 Knapsack decision problem is NP-complete.

5. Set Partition Problem

Given: n tasks with integer processing times $t_1, t_2, ..., t_n$ to be scheduled on two machines without overlaps.
Decision Problem: Can the processing times be equally divided among the two machines?
To prove that the Set Partition Problem is NP-complete, we first show that the problem is in NP and then we move on to the reduction step to prove its NP-completeness.

Part 1: Set Partition Problem $\in$ NP.
For the given problem we can come up with the following non-deterministic polynomial time algorithm:

Guess the partitions P_1 and P_2 as certificates.

Run them through the verifier to check the constraints of whether the sum of all times in P_1 and and the sum of all timesP_2 are equal. This would take O(n) time.
This shows that the verifier takes polynomial time to verify the certificate.
Thus, Set Partition Problem $\in$ NP.

Part 2: 0-1 Set Partition Problem is NP-hard.
We now reduce the subset sum problem to set partition problem.

The subset sum problem wants us to find out if there is a subset S of input numbers $X = s_1, s_2, ..., s_n$ with total sum T.

Let sum be the sum of members of S.

Feed $X' = X \cup (sum - 2T)$ into set partition and accept if and only if set partition accepts.

We now show that $\langle X, T\rangle \in$ subset sum iff $\langle X'\rangle \in$ set partition

The sum of members of X' is $(2sum - 2T)$

If there exists a set of numbers in X that sum to T, then the remaining numbers in X sum to $(sum - T)$. Therefore, there exists a partition of X' into two such that each partition sums to $(sum - T)$.

Let's say that there exists a partition of X' into two sets such that the sum over each set is $sum - T$. One of these sets contains the number $sum - 2T$. Removing this number, we get a set of numbers whose sum is T, and all of these numbers are in X.

Thus, set partition problem has a solution iff subset sum problem has a solution.

We have reduced subset sum problem to set partition problem in polynomial time and hence set partition problem is NP-complete.