Assignment 12 – More Intractability

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: Ramapriya Wisc id: rranganath

Reductions

1. *Kleinberg, Jon. Algorithm Design (p. 508, q. 9)* Consider the following situation: You are managing a communications network, modeled by a directed graph G. There are c users who are interested in making use of this network. User i issues a "request" to reserve a specific path P_i in G on which to transmit data.

   You are interested in accepting as many path requests as possible, but if you accept both P_i and P_j, the two paths cannot share any nodes.

   Thus, the Path Selection Problem asks, given a graph G and a set of requested paths $P_1, \ldots, P_c$ (each of which must be a path in G), and given a number k, is it possible to select at least k of the paths such that no two paths selected share any nodes?

   Show that Path Selection is also NP-Complete.

① The problem is NP.

Given set of paths $\{p_i\}$, iterate and check each node.

If encounter some node twice, return false.

② Independent set $\leq_p$ Path selection

For independent set each node corresponds to path in PS.

For all egdes $\{e_i\}$ adjacent to node u_j.

let path p_j in PS go through node $\{n_i\}$ in $\{e_i\}$.

2. *Kleinberg, Jon. Algorithm Design (p. 512, q. 14)* We've seen the Interval Scheduling Problem several times now, in different variations. Here we'll consider a computationally much harder version we'll call *Multiple Interval Scheduling*. As before, you have a processor that is available to run jobs over some period of time.

   People submit jobs to run on the processor. The processor can only work on one job at any single point in time. Jobs in this model, however, are more complicated than we've seen before. Each job requires a *set* of intervals of time during which it needs to use the processor. For example, a single job could require the processor from 10am to 11am and again from 2pm to 3pm. If you accept the job, it ties up your processor during those two hours, but you could still accept jobs that need time between 11am and 2pm.

   You are given a set of n jobs, each specified by a set of time intervals. For a given number k, is it possible to accept at least k of the jobs so that no two accepted jobs overlap in time?

   Show that Multiple Interval Scheduling is NP-Complete.

① The MIS is NP

Given a set of jobs $\{J_i\}$. Iterate all jobs and check the time intervals. If encounter some time intervals twice, return false.

② Independent set $\le_p$ MIS

For each node in IS corresponds to job in MIS.

each edge in IS corresponds to time interval in MIS

For all egdes $\{e_i\}$ adjacent to node u_j.

let job J_i in MIS require time interval $\{t_i\}$ in $\{e_i\}$.

3. *Kleinberg, Jon. Algorithm Design (p. 519, q. 28)* Consider this version of the Independent Set Problem. You are given an undirected graph G and an integer k. We will call a set of nodes I "strongly independent" if, for any two nodes $v, u \in I$, the edge (v, u) is not present in G, and neither is there a path of two edges from u to v, that is, there is no node w such that both (v, w) and (u, w) are present in G. The Strongly Independent Set problem is to decide whether G has a strongly independent set of size at least k.

   Show that the Strongly Independent Set Problem is NP-Complete.

① SIS in NP

Given a set of nodes I, use BFS check whether there is any other node $\in I$ are close to this node, if there is, return false.

② Independent Set. $\leq_p$ SIS

Consider a graph G in IS,

For each $e_i = (u, v)$ in G, add w_i

So it becomes (u, w_i), (w_i, v)

then add edge between each w_i nodes.

4. *Kleinberg, Jon. Algorithm Design (p. 527, q. 39)* The *Directed Disjoint Paths Problem* may look initially similar to the problem from the question 1. Pay attention to the differences between them!

   The *Directed Disjoint Paths Problem* is defined as follows: We are given a directed graph G and k pairs of nodes $(s_1, t_1), \ldots, (s_k, t_k)$. The problem is to decide whether there exist node-disjoint paths $P_1, \ldots, P_k$ so that P_i goes from s_i to t_i.

   Show that Directed Disjoint Paths is NP-Complete.

① First prove it's NP

Given set $\{P_i\}$, iterate through each nodes, if encounter some node twice, return false.

② 3-SAT $\le_p$ DDP.

For each x_i, we have k start nodes $s_1^i \ldots s_k^i$

$n_1^i \ldots n_k^i$. $t_1^i \ldots t_k^i$

We have point nodes $p_1^i \ldots p_{2k}^i$

Draw $s_a^i \to p_{2a-1}^i \to v_a^i \to t_a^i$ and $s_a^i \to p_{2a-2}^i \to v_{a-1}^i \to t_a^i$

For each clause C_j, create a node s_j and t_j

if $x_i \in C_j$. draw $s_j \to p_{2j-1}^i \to t_j$.

if $\bar{x}_i \in C_j$. draw $s_j \to p_{2j}^i \to t_j$.