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: Zinnia Nie Wisc id: 908 319 4044

Reductions

1. *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. Show Multiple Interval Scheduling is NP
   - Certificate is a set of jobs
   - If set size < k: return no
   - loop through set of jobs:
     - check that they fall in valid time intervals
     - make sure their set of intervals have not been used by another job
   - The loop is polynomial, so this can be done in polynomial time
2. Independent Set $\leq_p$ Multiple Interval Scheduling
   - change a graph G into MIS instance
     - m is the number of edges in G, each correspond with an equal size time slice s_i.
     - each node is a job that require the time slices corresponding to the edges connected to the node
   - Independent Set is yes → MIS is yes
     - an independent node set shares no edges, since each edge is coded as a time slice, a set with no shared edges means no shared time slices, so MIS is a yes
   - MIS is yes → Independent set is yes
     - MIS gives a set of jobs with no overlapping / shared time slices. This means that a set of nodes has no 2 nodes that are end points of the same edge, which is an independent set

2. *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. Prove Strongly Independent Set is NP
   - Certificate is a set of nodes
   - If size of set < k : return no
   - loop through nodes and check if they fulfill the strongly independent requirements.
   - This can be done in polynomial time
2. Independent Set $\leq_p$ Strongly Independent Set
   - For a graph G, add a node in between each pair of nodes connected by an edge, so edge (u,v) becomes (u,w) and (w,v)
   - If IS is yes → SIS is a yes
     - IS gives a set of nodes that do not share edges. The reduction changes each edge into a path of 2. Therefore, for an edge (u,v) only one of the end points is in solution set. (u,v) is now a 2 edge path, so they are not in the same set for SIS.
   - If SIS is a yes → IS must have been a yes
     - SIS gives a set of k nodes. The new nodes cannot be in the set because those nodes are all within 2 edges of all nodes in the graph
     - The added node makes any neighbors in the IS graph have a path of 2 between them, so they would not be in the set of nodes.
     - Since the k nodes are all in the IS graph and are not neighbors, they form an Independent Set.

3. *Kleinberg, Jon. Algorithm Design (p. 527, q. 39)* 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. Prove Directed Disjoint Path is NP
   - certificate is a set of paths
   - If num of paths $\neq k$: return no
   - loop through paths and make sure there are no duplicate nodes
   - This can be done in $O(nk)$ time
2. 3SAT $\leq_p$ Directed Disjoint Path
   - set of k clauses and n variables
   - $2n$ disjoint paths: each length k, one path P_i corresponding with x_i and another path corresponding with $\overline{x_i}$
   - have n (s_i, t_i) pairs and connect paths P_i and P_i' to it
   - select $P_i \rightarrow$ set x_i to false
   - Select $P_i' \rightarrow$ set x_i to true
   - add k more (s_j, t_j) nodes that correspond to each clause C_j
     - add edges to the jth node of each of the 3 paths corresponding to the literals used in Clause j, one from s_j, one to t_j.
   - If 3SAT is yes $\rightarrow$ DDP is yes
     - Select the path corresponding to the assignment of x_i
     - Since x_i can't be both true and false, each (s_i, t_i) pair will only have one disjoint path connecting them
   - If DDP is yes $\rightarrow$ 3SAT is a yes
     - DDP gives a set of disjoint paths connecting all k (s_i, t_i) pairs
     - the (s_i, t_i) pairs have 2 disjoint paths that only share nodes s_i and t_i which are not connected to anything else. Paths are not connected to anything else.
     - Therefore, each (s_i, t_i) pair only has one path chosen an each (s_j, t_j) pair has 3 paths chosen. This creates a satisfying assignment for the 3 variables in the clause.

4. *Kleinberg, Jon. Algorithm Design (p. 508, q. 9)* The *Path Selection Problem* may look initially similar to the problem from the question 3. Pay attention to the differences between them! 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. Show Path Selection is NP.
   - The certificate is a set of paths and a set of requested paths
   - If num of paths $\neq k$: return no
   - Loop through path set :
     - determine if the path is in the requested path set
     - See if there are any duplicate nodes in the path
   - This is polynomial in $O(n)$ time
2. Independent Set $\leq_p$ Path Selection
   - every edge in IS becomes a node in PS
   - every node in IS becomes a path of all edges that are connected to that node in PS
   - these paths are connected to a source s and sink t to account for isolated nodes.
   - If IS is yes → PS is yes
     - IS gives a set of k nodes that share no edges. Since every edge in IS is a node in PS, no shared edges means no shared nodes and PS has k disjoint paths
   - If PS is yes → IS had to have been yes
     - PS being yes means k paths do not share any nodes. Each path in PS is a node in IS and all nodes in the path are edges connected to the node in IS. Since these paths share no nodes, this means the edges connected to the corresponding node in IS are also not shared and IS is an independent set of k nodes.