CS 577 # Assignment 12 – More Intractability Spring 2024

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: Mahir Khan Wisc id: mhkhan9@wisc.edu

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) only 1 outgoing edge per node, cuz when a job w/ intervals is selected, those intervals can no longer be selected

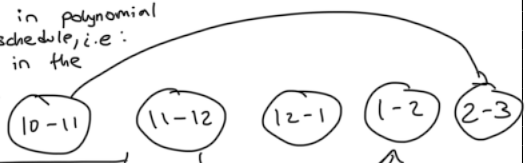
2) only 1 incoming edge per node for same reason as above.

Independent set reduction?

(MIS)

Multiple Interval Scheduling is in NP:

* Certificate is a schedule of n jobs. Certifier checks in polynomial time that a certificate has no conflicts in time slots in the schedule, i.e: no overlaps in time, and also checks that there are k jobs in the schedule. If these conditions are satisfied, then a yes instance has been found. Otherwise, certifier will return a no instance.

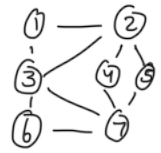


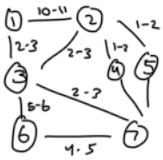
MIS is NP Hard :

Independent Set ≤p Multiple Interval Scheduling

Assume a graph G which is an instance of Independent set. G* = MIS graph instance.

* For each vertex in G create a job j in G*
* For each edge in G, create a time interval i in G*
* Time intervals between jobs in G* represent conflicts, i.e: if a job v_j has an edge with job$_k$, this means that both jobs require the same time interval in order to be completed, thus a conflict.
* If there are no edges between a job v_j and job v_k, this means there are no scheduling conflicts between the 2 jobs





Problem to Reduce:

Independent set, both try to maximise k nodes in set w/ independent nodes (jobs)

Proof:

=> A yes instance of size k on graph G on the independent set instance shows that we can select k nodes in a set such that none of the nodes are adjacent to one another in the set. This is reflected in our MIS reduction as each node in G represents a job in G*, and each edge in G represents a time interval in G*, thus there are only conflicts with jobs that share a time slot. In the independent set, there are no edges with adjacent nodes, thus in the reduction there is no conflict in time intervals between jobs due to how we set up the problem. Thus, G* will also contain a set of k jobs with no conflict in time intervals, thus showing =>

<= If there are k jobs that can be run without any conflicts, then by the reduction done above, it must mean there is an independent set of k nodes in Graph g based on the fact that jobs with no conflicts w/ other jobs means that based on our reduction mapping, there must be no nodes with adjacent nodes in the set.

↳ Thus a polynomial time reduction from IS to MIS is proven, showing that MIS is NP-Hard, and thus NP-Complete due to transivity.

We have also shown that a yes instance in IS is a yes instance in MIS, and vice versa.

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.

Strongly Independ Set (SIS) is NP

A certificate contains a set of nodes from graph G, and the certifier has to check that: 1) none of the nodes share an edge 2) none of the nodes have an edge to an arbitrary node w, where w is adjacent to a different node in the set and 3) that the # of strongly independent sets = = k. ↳ This can be done by using a searching algorithm such as BFS, thus we can polynomially certify certificates of SIS.

Strongly Independent set is NP-Hard

Independent Set $\leq_p$ SIS : Assume a graph G which has an instance of an Independent set. G* will be the SIS reduction instance.

* For every node v_i in G, create a node v_i in G*
* For every edge e_i between (u, v) in G, create a node w_i, and edges between (u, w_i) and (w_i, v) i.e: (u)—(v) ⟹ (u)(w)(v) → Meets the strongly connected limit now, thus SIS will not discard any additional nodes that would have otherwise been in an independent set.
* Add an edge between all added w nodes so that they aren't considered for the Strongly independent set.

Proof

⇒ If there is an independent set of size k, then the same set is strongly independent in our reduction graph G*. This is true because an extra node w_i was added for every edge pair (u, v) such that there is now a path of two edges between u and v, meaning an artifical path was created so that any nodes in the IS are not removed in the SIS due to having a path of size 2 to another node, thus a yes instance in IS is a yes instance in our SIS reduction

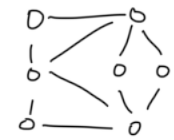
⇐ If there is a size k of a strongly independent set in G', then based on our construction of the reduction, there must be an independent set in G. There is no chance of any w node to be a part of the SIS since they are all connected to one another, thus not independent. And due to how we structured the reduction, a SIS in G (yes instance) correlates directly to a yes instance of IS in G.

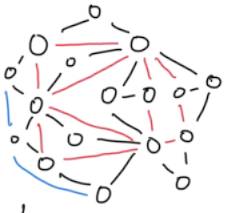
Thus we have a poly-time mapping of IS to SIS, showing that SIS is NP-Hard, and in turn, NP-Complete. (As IS is NP-complete too)

↳ we proved SIS is in NP above

Not strongly connected

Independent set = ✓ SIS = X



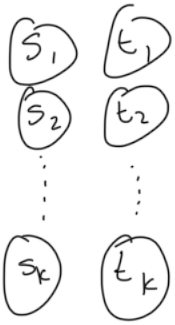


↳ What this does is increase the distance by 1 for EVERY node w/ adjacent nodes, thus basically, It's the same as IS, and the additional path restriction in SIS is circumvented.

↳ Basically add an intermediary node between every node pair, thus creating adjacency paths.

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 $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. 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.



DDP

↳ k paths

DDP is NP

Certificate: For a set of paths $P_1, \ldots P_k$, make a new array that keeps tracks of visited node. Then, for each path P_i, check that all nodes in P_i are not in the array, and if not, then add the node to the array. If all k paths are iterated through with no conflicts, then the certificate is valid, giving a yes instance. If any node is found to already be in the array, then the certificate is not valid.

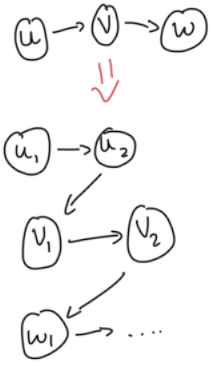
↳ $O(|V|)$

DDP is NP-Hard ⟵ **Hamiltonian Path $\leq_p$ DDP**

Take an instance of the Hamiltonian Path problem on a graph G. Construct a new graph G' with:

* Duplicate each node v in G to v' and v'' in G', where v' is an "in" node, and v'' is an "out" node.
* All v_i' nodes have a directed edge to corresponding v_i'' nodes
* For each edge e_i for (u, v) in G, add a directed edge from u'' to v' (out node to in node) in G'

↳ the number of node-disjoint paths in G' will be equal to the number of nodes in graph G.



Proof: Yes instance in HP ⟺ Yes instance in DDG

⇒ If there is a Hamiltonian path $P = \{P_1, P_2 \ldots P_k\}$ of size k, then our reduction graph G* will also have k node-disjoint paths. This is because each path P_i from HP will go from u' to u'' to v'..., and each distinct node-disjoint path will be found, and thus showing a yes instance in DDG.

⇐ Assume G' has k node-disjoint paths $P = \{P_1, P_2, \ldots, P_k\}$. By construction, all u_i'' nodes have an edge to v', which corresponds to an edge in the original graph G. Therefore, each path visits each node in G exactly once, following the edges specified by the paths in G'

↳ Thus, both directions have been shown, thus DDG has been proven to be in NP-Hard. Since we also proved that it's in NP, thus DDG is an NP-Complete problem.

4. *Kleinberg, Jon. Algorithm Design (p. 508, q. 9)* The *Path Selection Problem* may look initially similar to the *Directed Disjoint Paths Problem*, but 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.

PSP is NP

Certificate: For a set of paths $P_1, \ldots P_k$, make a new array that keeps tracks of visited node. Then, for each path P_i, check that all nodes in P_i are not in the array, and if not, then add the node to the array. If all k paths are iterated through with no conflicts, then the certificate is valid, giving a yes instance. If any node is found to already be in the array, then the certificate is not valid.

↳ O(|V|)

PSP is NP-Hard ← Vertex Cover $\leq_p$ PSP

Take an instance of the Vertex Cover problem on a graph G. Construct a new graph G' with:

* Duplicate each node v in G to v' and v'' in G', where v' is an "in" node, and v'' is an "out" node.
* All v_i' nodes have a directed edge to corresponding v_i'' nodes
* For each edge e_i for (u, v) in G, add a directed edge from u'' to v' (out node to in node) in G'

↳ the number of node-disjoint paths in G' will be equal to the number of nodes in graph G.

Proof: Yes instance in VC <=> Yes instance in PSP

=> If G has a VC of size k, selecting the paths P_i for each node $v \in S$ in G' satisfies the path selection problem. These paths are disjoint because no 2 nodes in S are connected by an edge in the original graph, and the "conflict" edges in G prevent paths using different nodes in S from overlapping.

<= k paths in G' show a VC in G

↳ IDK how to prove this to be honest.

Need to reevaluate!