CS 577 # Assignment 11 – Intractability Spring 2023

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: Matej Popovski Wisc id:

Intractibility

1. *Kleinberg, Jon. Algorithm Design (p. 506, q. 4).* A system has a set of n processes and a set of m resources. At any point in time, each process specifies a set of resources that it requests to use. Each resource might be requested by many processes at once; but it can only be used by a single process at a time. If a process is allocated all the resources it requests, then it is active; otherwise it is blocked.

   Thus we phrase the Resource Reservation Problem as follows: Given a set of processes and resources, the set of requested resources for each process, and a number k, is it possible to allocate resources to processes so that at least k processes will be active?

   For the following problems, either give a polynomial-time algorithm or prove the problem is NP-complete.

   (a) The general Resource Reservation Problem defined above.

Solution: The Resource Reservation problem is stated: For a set of n processes each requested

a subset of m resources decide if there are k >=1 processes whose requested resources are disjoint To prove we use Independent set problem known to be NP complete Independent Set <= p Resource reservation Independent Set problem we stated as equivalent to a graph G where we define nodes as processors and edges incident to one processor v are resources used by this processor. This reduction need polynomial time to complete. If there are k processor whose resources are disjoint than they form an independent set. This is true as any edge between these nodes would be a resource that they both request. If there is an independent set of size k, than k correspond to these nodes from a set of k processors that request disjoint sets of resources

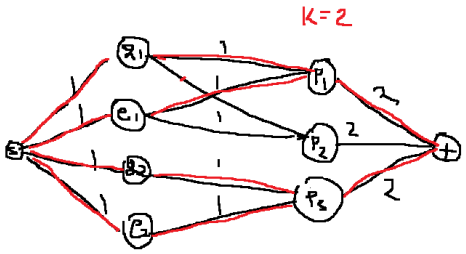
(b) The special case of the problem when $k = 2$.

Solution:

The case k=2 can be solved directly by trying all pairs of processes, if they have common edge. (O(n*n) time).

(c) The special case of the problem when there are two types of resources–say, people and equipment– and each process requires at most one resource of each type (In other words, each process requires one specific person and one specific piece of equipment.)

Solution: This can be seen as Bipartite matching problem. For each person and each equipment we define edge to process which need them. We than checks if there is matching at least of size k.



(d) The special case of the problem when each resource is requested by at most two processes.

Solution:

This is special case of reduction in a) each edge/resource in the graph connects only the two nodes (processes)

2. *Kleinberg, Jon. Algorithm Design (p. 506, q. 7).* The 3-Dimensional Matching Problem is an NP-complete problem defined as follows:

   Given disjoint sets X, Y, and Z, each of size n, and given a set $T \subseteq X \times Y \times Z$ of ordered triples, does there exist a set of n triples in T that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples?

   Since 3-Dimensional Matching is NP-complete, it is natural to expect that the 4-Dimensional Problem is at least as hard.

   Let us define 4-Dimensional Matching as follows. Given sets W, X, Y, and Z, each of size n, and a collection C of ordered 4-tuples of the form (w_i, x_j, y_k, z_ℓ), do there exist n 4-tuples from C such that each element of $W \cup Y \cup X \cup Z$ appears in exactly one of these 4-tuples?

   Prove that 4-Dimensional Matching is NP-complete. Hint: use a reduction from 3-Dimensional Matching.

Solution:

4-Dimensional matching is NP, since we can check in polynomial time whether the set of n, 4- Dimensional tuples have no two element in common.

Next we show reduction 3-Dimensional Matching ≤p 4-Dimensional Matching.

We define collection C, instance of 3-Dimensional Matching, with sets X, Y, and Z of size n each, and of ordered triples. 4-Dimensional Matching can be defined by sets W, X, Y, and Z, each of size n, as a collection C' of 4-Dimension matching so that for every (Xj, Yk, Zl) ∈ C, and every i between 1 and n, there is a 4-tuple (wi, xj, yk, zl). This instance has a size that is polynomial in the size of the initial 3-Dimensional Matching instance. If A=(xj, yk, zl) is triple in C let f(A) be 4-tuple (wi, xj, yk, zl) in C'. If B=(wi, xj, yk, zl) is a 4-tuple in C', define f'(B) to be the triple (xj, yk, zl). It is element of C.

Given a set of n, {Ai} disjoint triples in C, {f(Ai)} is a set of n disjoint 4-tuples in C'. Conversely given a set of n disjoint 4-tuples {Bi} in C' than {f'(Bi)} is a set of n disjoint triples in C. Thus by determining weather there is perfect 4-Dimensional matching in the instance it is constructed, it can be solved initial instance of 3-Dimensional matching

3. *Kleinberg, Jon. Algorithm Design (p. 507, q. 6).* Consider an instance of the Satisfiability Problem, specified by clauses $C_1, ..., C_m$ over a set of Boolean variables $x_1, ..., x_n$. We say that the instance is monotone if each term in each clause consists of a nonnegated variable; that is, each term is equal to x_i, for some i, rather than $\overline{x_i}$. Monotone instances of Satisfiability are very easy to solve: They are always satisfiable, by setting each variable equal to 1.

   For example, suppose we have the three clauses

   $$(x_1 \vee x_2), (x_1 \vee x_3), (x_2 \vee x_3).$$

   This is monotone, and the assignment that sets all three variables to 1 satisfies all the clauses. But we can observe that this is not the only satisfying assignment; we could also have set x_1 and x_2 to 1, and x_3 to 0. Indeed, for any monotone instance, it is natural to ask how few variables we need to set to 1 in order to satisfy it.

   Given a monotone instance of Satisfiability, together with a number k, the problem of *Monotone Satisfiability with Few True Variables* asks: Is there a satisfying assignment for the instance in which at most k variables are set to 1? Prove this problem is NP-complete.

Solution:

Monotone Satisfiability with Few True Variables is in NP. Given instance of Satisfiability and number k, we can verify in polynomial time whether a given assignment satisfy the instance and has at most k variables set.

Problem can be defined as type of covering problem. A small number of variables has to be chosen in such a way that each clause contains one of chosen variables. Vertex Cover is chosen to show

Vertex Cover $<=_p$ Monotone Satisfiability with Few true Values

Given Graph G(V,E) and number k, we want to decide whether there is a vertex cover in G of size at most k. Let vertex v_i in G, represent variable x_i. Each edge $e_j=(v_a,v_b)$ represents clause $C_j=(x_a \vee x_b)$.The full instance are clauses C1,C2,....Cm one for each edge in G. We have to check if they can all be satisfied by setting at most k variables to 1. The answer to the Vertex Cover instance is "yes" iff the answer to the monotone Satisfiability is "yes". Suppose there is a vertex cover S in G of size at most k, and setting the corresponding variables to 1 (all other 0). Since each edge is covered by a member of S, each clause contains at least one variable set to1, so all clauses are satisfied. Conversely suppose there is a way to satisfy all clauses by setting a subset X of at moat k variables to 1. Then if we consider corresponding variables in G, each edge must have at least one end equal to one of these vertices since the clause corresponding to this edge contains a variable X. Thus the nodes corresponding to the variables in X form a vertex cover of size at most k.

4. *Kleinberg, Jon. Algorithm Design (p. 509, q. 10).* Your friends at WebExodus have recently been doing some consulting work for companies that maintain large, publicly accessible Web sites and they've come across the following Strategic Advertising Problem.

   A company comes to them with the map of a Web site, which we'll model as a directed graph $G = (V, E)$. The company also provides a set of t trails typically followed by users of the site; we'll model these trails as directed paths $P_1, P_2, ..., P_t$ in the graph G (i.e., each P_i is a path in G).

   The company wants WebExodus to answer the following question for them: Given G, the paths $\{P_i\}$, and a number k, is it possible to place advertisements on at most k of the nodes in G, so that each path P_i includes at least one node containing an advertisement? We'll call this the Strategic Advertising Problem, with input $G, \{P_i : i = 1, ..., t\}$, and k. Your friends figure that a good algorithm for this will make them all rich; unfortunately, things are never quite this simple.

   (a) Prove that Strategic Advertising is NP-Complete.

Solution:

Set of advertisement is "valid" if it covers all paths {P}. Strategic Advertising (SA) is NP: Given a set of k nodes, we can check in polynomial time whether at list one of them lies on path P. To show that it is NP-complete Vertex Cover ≤p SA is used.

Let from the undirected graph G=(V,E) and number k, produce Graph G'=(V,E') by arbitrary direction of each edge.

We'll say a set of advertisements is "valid" if it covers all paths in {P}. First, Strategic Advertising (SA) is in NP: Given a set of k nodes, we can check in O(kn) time (or better) whether at least one of them lies on a path P, and so we can check whether it is a valid set of advertisements in time (knt).

We have to show that Vertex Cover ≤p SA.

Given an undirected graph G = (V, E) and a number k, we produce a directed graph G' (V, E') by arbitrarily directing each edge of G. For each edge in E' define a path Pi. This involves one pass over edges, taking the polynomial time. Claim is that G has a valid set of at most k advertisements iff G has a vertex cover T of size at most k. Suppose G' have such a valid set U. Since it meets at least one end of each edge, it is vertex cover for G. Conversely, suppose G has a vertex cover T of size at most k, then set T meets each path in {Pi} so it is a valid set of advertisements

(b) Your friends at WebExodus forge ahead and write a pretty fast algorithm $\mathcal{S}$ that produces yes/no answers to arbitrary instances of the Strategic Advertising Problem. You may assume that the algorithm $\mathcal{S}$ is always correct.

Using the algorithm $\mathcal{S}$ as a black box, design an algorithm that takes input $G, \{P_i : i = 1, ..., t\}$, and k as in part (a), and does one of the following two things:

- Outputs a set of at most k nodes in G so that each path P_i includes at least one of these nodes.
- Outputs (correctly) that no such set of at most k nodes exists.

Your algorithm should use at most polynomial number of steps, together with at most polynomial number of calls to the algorithm $\mathcal{S}$.

Solution:

Construct the algorithm by induction on k. If k =1, simply check whether there is any node that lies on all paths. Otherwise, ask the fast algorithm S whether there is a valid set of advertisements of size at most k. If "no," simply report this. If "yes", perform the following test for each node v: delete v and all paths through it, and ask S whether, on this new input, there is a valid set of advertisements of size at most k= 1. Claim is that there is at least one node v where this test will succeed. For consider any valid set U of at most k advertisements (one exists since S said "yes"). The test will succeed on any v EU, since U - {v} is a valid set of at most k=1 advertisements on the new input.

Once such a node is identified, add it to a set T that is maintained. Now we are dealing with an input that has a valid set of at most k – 1 advertisements, and so algorithm will finish the construction of T correctly by induction. The running time of the algorithm involves O(n+t) operations and calls to S for each fixed value of k, for a total of O(n2 + nt) operations.