CS 577 # Assignment 11 – Intractability Fall 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: Riya Kore Wisc id: rykore

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:

1. The Resource Reservation Problem (RRP) is generally classified as NP. By considering a set of k processes, it is possible to efficiently verify the absence of resource overlap among them. This verification involves counting, for each resource, how many selected processes have requested that resource, with a time complexity of O(km). If all counts are less than or equal to 1, the solution is deemed valid; otherwise, it is considered invalid. The confirmation or rejection of a proposed solution can be achieved in polynomial time, establishing the problem's NP status.
2. The NP-Hard nature of the general RRP can be demonstrated through a polynomial-time reduction from the Independent Set problem (IS) to RRP. Given an arbitrary instance of IS defined on a graph G and an integer k, the transformation involves representing the nodes of G as the set of processes and the edges as the set of resources. A process requires a resource if and only if the corresponding node is adjacent to the edge in G. This transformation is polynomial, and the correctness of the reduction is proven as follows: an independent set of k nodes in G, having no common edges, implies no overlap in required resources among processes in the transformed RRP instance. Conversely, if k processes exhibit no overlap in required resources, the corresponding nodes in G must have no common edges. Thus, a positive instance in IS corresponds to a positive instance in the general RRP, establishing the NP-Hardness of RRP as Independent Set is NP-Complete.

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

Solution:

Consider the following polynomial-time algorithm which solves the special case of k = 2:
For each pair of processes, check all m resources to see if they have any requirements in common ($O(mn^2)$).
If there is a pair of processes which have no requirements in common, the answer is yes.
If no such pair exists, the answer is no.

(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:

Consider the set of processes denoted as S, where each process only requires one resource, and all resource requests are mutually exclusive. Eliminate any processes that request a resource already required by a process in S, a task achievable in polynomial time based on the number of processes.

Next, establish two sets of nodes, namely T_1 and T_2, corresponding to each type of resource. For each resource r requested by processes in S, remove the corresponding node for r. This removal operation is polynomial in the number of resources.

The problem at hand can now be approached as a bipartite matching problem, a task solvable in polynomial time. The goal is to find the desired number of matches, which is k - |S|.

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

Solution:

The reduction outlined in (a) corresponds to this special case, since the resource/edge of independent set are adjacent to exactly two process/nodes. Thus, this special case is still NP-Complete, by that reduction.

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 such 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:

1. The problem of 4-Dimensional Matching falls within the class of NP. Given n 4-tuples in C, it can be easily verified in polynomial time whether each element in the union of sets W, X, Y, and Z belongs to exactly one of the n tuples. Hence, 4-Dimensional Matching is categorized as an NP problem.

2. The NP-Hardness of 4-Dimensional Matching is established by demonstrating a polynomial-time reduction from 3-Dimensional Matching (3DM). In a given instance of 3DM defined on sets X, Y, and Z, each of size n, along with a set T belonging to X x Y x Z of ordered triples, this can be transformed into a 4D Matching instance by padding the tuples. One approach is to duplicate one of the sets and associated elements, creating a set T' in X x X x Y x Z with the same ordered triplets but replacing <x, y, z> with <x, x, y, z>. This transformation is polynomial-time. The resulting 4-Dimensional Matching instance includes the union of sets X, Y, and Z, and it is observed that X U X U Y U Z is equivalent to X U Y U Z. Additionally, for any element a belonging to X U Y U Z, a corresponds to <x, y, z> if and only if a is in x, y, z and is also in <x, x, y, z>. If there exists a set of n triples satisfying the instance of 3DM, then there exists a set of n 4-tuples satisfying the instance of 4-Dimensional Matching, and vice versa. Consequently, 4-Dimensional Matching is proven to be NP-Hard.

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: 1. Monotone Satisfiability with Few True Variables (MSFTV) is within the class of NP.

When provided with k variables to set to 1, along with a boolean formula ψ, it can be efficiently verified in polynomial time whether ψ is monotone and satisfied by setting the k identified variables to 1, while all other variables are set to 0. Therefore, MSFTV is categorized as an NP problem.

2. The NP-Hardness of MSFTV is demonstrated through a polynomial-time reduction from the Vertex Cover problem. In an arbitrary instance of vertex cover defined on a graph G = (V,E) and a number k, variables x_v are created for every v in V. Clauses $(x_u or x_v)$ are generated for every u,v in E, and the formula ψ is the conjunction of all such clauses. This transformation is clearly polynomial, involving the processing of each vertex and edge in G, and it consistently produces a monotone formula. Thus, an arbitrary instance of vertex cover is transformed into an instance of MSFTV.

3. If there exists a vertex cover for G of size at most k, setting the corresponding variables to 1 in ψ results in every clause evaluating to true, making ψ satisfiable. Conversely, if there is a satisfying assignment for ψ in which at most k variables are set to 1, the monotonicity of ψ implies satisfiability only if at least one variable in every clause is set to 1. Taking the vertices corresponding to these variables forms a vertex cover with at most k vertices.

In conclusion, Monotone Satisfiability with Few True Variables is established as NP-Hard.

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: 1. Strategic Advertising falls within the NP class. Given an instance of the Strategic Advertising Problem on graph G, represented by $P_i : i = 1, ..., t$, and a parameter k, the validity of a solution proposing k or fewer nodes for placing advertisements can be efficiently checked. This verification involves examining all t paths to confirm the presence of any of the k nodes (O(kt|V|)), rejecting the solution if any path lacks all selected nodes and accepting otherwise.

2. The NP-Hardness of Strategic Advertising is demonstrated through a polynomial-time reduction from the Vertex Cover problem. In an arbitrary instance of vertex cover defined on an undirected graph G = (V,E) with a parameter k, direct all edges arbitrarily (O(|E|)) to create the directed graph G'. Each new directed edge e_i corresponds to a path P_i consisting of a single edge, also O(|E|). This transformation converts an arbitrary instance of Vertex Cover into an instance of Strategic Advertising.

If G has a vertex cover with at most k vertices, then every path in G' includes at least one of those vertices, establishing the existence of a strategic advertisement. Conversely, if there is a strategic advertisement for G' with at most k vertices, then every path contains at least one of those vertices, as per the problem definition. If every path in G' contains at least one of the selected vertices, every edge in G is incident on one of the (at most) k vertices, leading to the conclusion that there exists a vertex cover of size at most k.

In summary, Strategic Advertising is affirmed as NP-Hard.

(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:

1. Query set S on the original problem instance:

If S returns no, output that there is no set of at most k nodes that satisfies the condition.
2. If S returns yes, proceed to the next step.
Create a graph G' = (V', E'), along with a copy P' of the paths and a counter k', initially set to k. Maintain a set A for advertising, initially empty.

3. Arbitrarily choose a vertex v from V':

For each occurrence of v in E':
If u,v is a directed edge and no edge v,w exists, remove u,v from E'.
If v,w is a directed edge and no edge u,v exists, remove v,w from E'.
If u,v and v,w are both directed edges, create a new edge u,w and remove u,v and v,w from E'.
Apply the above steps to all paths P_i' containing v, achievable in $O(t|E'|^2)$. Remove v from V'.
4. Query set S on G', P', and k':

If S returns no:

Add v to set A.
Remove all paths from P' containing v.
Decrement k'.
If k' is 0 or the size of A equals k, output the contents of A.
If k' $>$ 0, return to step 3.
If S returns yes, return to step 3.