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: Carson Miller   Wisc id: 9081473028

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 certificate would be a set of processes and their assigned resources. This would result in n*m certificates, which we could verify in polynomial time, checking to see if the pairings resulted in at least k processes being active.

To reduce 3SAT to RRP, we will create two processes and a resource in RRP for each literal in each clause from 3SAT. One process will request its literal's corresponding resource, representing the literal. The other process will request the same resource, representing the negation of the literal. This can be done in polynomial time, as there are only n processes.

For a yes 3SAT, for each literal in each clause we will allocate resources for itself and its negation to the process corresponding to the literal. This will result in each clause having at least k active processes.

For a yes RRP, there will be at least k active processes and thus there will be at least one process in each clause that is active. This corresponds to at least one literal from each clause being a 1, meaning yes for 3SAT.

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

Solution:

We can reduce this problem to the maximum matching bipartite graph problem from network flow. The resources would be the right column, the processes would be the left column. There is a source that connects to all the processes with edges of capacity 1. Then, all processes are connected to the resources that they need with edges of capacity 1. Lastly, all resources are connected to the sink with edges of capacity 1. Calculate the max-flow. If it is greater than 2, the answer to the original question is yes. If not, it 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:

This special case can also be reduced to a network flow graph. Connect the source node to all processes with edges of capacity 2. Then, connect each process to the resources it requires with edges of capacity 1. Then, add edges from each resource to the sink with edges of capacity 1.

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

Solution: This is no different than (a).

The certificate would be a set of processes and their assigned resources. This would result in n*m certificates, which we could verify in polynomial time, checking to see if the pairings resulted in at least k processes being active.

To reduce 3SAT to RRP, we will create two processes and a resource in RRP for each literal in each clause from 3SAT. One process will request its literal's corresponding resource, representing the literal. The other process will request the same resource, representing the negation of the literal. This can be done in polynomial time, as there are only n processes.

For a yes 3SAT, for each literal in each clause we will allocate resources for itself and its negation to the process corresponding to the literal. This will result in each clause having at least k active processes.

For a yes RRP, there will be at least k active processes and thus there will be at least one process in each clause that is active. This corresponds to at least one literal from each clause being a 1, meaning yes for 3SAT.

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:

The certificate would be a set C of n 4-tuples, each with one element from W, X, Y, and Z. There are n^4 certificates, so certifying a solution can be done in polynomial time, thus 4DM is NP.

To reduce 3DM to 4DM, the sets X, Y, and Z remain unchanged, while adding a set of size n, W, that contains integers 1 to n. This can be done in polynomial time. Now, when you solve 4DM, you are also finding a solution to 3DM.

For a yes 3DM, if you were to add a 4th set with n unique elements, you could pair one element from the 4th set with each of n sets from the solution to 3DM. This would yield a yes for 4DM.

For a yes 4DM, you could drop one element from each 4-tuple that is originally from the same set (W, X, Y, or Z). This would yield n triples that would be a yes for 3DM.

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:

The certificate would be the set of k variables that are either 0s or 1s. It takes polynomial time to see if this instance satisfies all the clauses and if there are k or less variables set to 1.

To reduce CSAT to MSAT, we could conserve the set of boolean variables and their corresponding clauses that they are assigned to. They only modification needed to be made would be to set k equal to n.

For a yes CSAT, the output of the boolean circuit is a 1. This means for the corresponding MSAT, the output is 1. There can't possibly be more than k variables set to 1 because there are only n variables and k=n.

For a yes MSAT, the output of the boolean circuit is a 1. This means the variables and clauses are in an arrangement that produces a 1 as the output, which is all that is needed for a CSAT yes.

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:

The certificate would be the set of vertices one which advertisements are placed. In polynomial time, we could certify that each path P includes at least one node and that there are at most k nodes in the set.

To reduce Set Cover to SAP, create a node in G for each element in the universe U from SC. For each subset S from SC, create a path P in G that includes a node for each element in S. Also, k = m.

For a yes SC, there exists a set of subsets where the union of those subsets is equal to U. This means we can place an ad on any node in P corresponding to its subset S from SC. Since the SC covers all elements in U, each path in G will have at least one ad.

For a yes SAP, there exists a set of nodes where every P has at least one ad on it. For each ad placed on a node in P, including the corresponding subset in the SC. Because each path P includes at least one node containing an ad, all n elements in U will be covered.

(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:

For the second thing we could do, S gives us that answer. Given a graph G, paths P, and a constant k, S can tell us whether or not we can place at most k advertisements and get a graph where all paths P have at least one ad on them. If S says no, we can output that no such set of at most k nodes exists. If S says yes, we can output that a set of at most k nodes does exist.