Lab11-NP Reduction

CS214-Algorithm and Complexity, Xiaofeng Gao & Lei Wang, Spring 2021.

∗ If there is any problem, please contact TA Yihao Xie.

∗ Name:Zirui Liu Student ID:519021910343 Email: L.prime@sjtu.edu.cn

1. We are feeling experimental and want to create a new dish. There are various ingredients we can choose from and we'd like to use as many of them as possible, but some ingredients don't go well with others. If there are n possible ingredients (numbered 1 to n), we write down an $n \cdot n$ matrix giving the discord between any pair of ingredients. This discord is a real number between 0.0 and 1.0, where 0.0 means "they go together perfectly" and 1.0 means "they really don't go together." Here's an example matrix when there are five possible ingredients.

	1	2	3	4	5
1	0.0	0.4	0.2	0.9	1.0
2	0.4	0.0	0.1	1.0	0.2
3	0.2	0.1	0.0	0.8	0.5
4	0.9	1.0	0.8	0.0	0.2
5	1.0	0.2	0.5	0.2	0.0

In this case, ingredients 2 and 3 go together pretty well whereas 1 and 5 clash badly. Notice that this matrix is necessarily symmetric; and that the diagonal entries are always 0.0. Any set of ingredients incurs a penalty which is the sum of all discord values between pairs of ingredients. For instance, the set of ingredients $(1, 3, 5)$ incurs a penalty of $0.2+1.0+0.5 = 1.7$. We define the Experimental Cuisine as follows:

Given n ingredients to choose from, the $n \times n$ discord matrix and integer k and a number p, decide whether there exists a collection of at least k ingredients that has a penalty $\leqslant p$

Prove that 3-SAT $\leq_p$ Experimental Cuisine

Solution. We will show that 3-SAT problem can be changed into Experimental Cuisine problem. Assume that for any 3-SAT problem which made of n clauses, for one of the clauses that $A = (x \vee y \vee z)$, we then have 7 ingredients, representing 7 different cases for A to be satisfied. But these 7 cases can not be correct at the same time, so it is safe for us to set all the discord between these 7 ingredients to be 1. For different clauses of ingredients, if they have conflicts, discord is set to be 1. If NO CONFLICTS, DISCORD WILL BE SET 0. When we can choose enough ingredients, whose number of clauses equal to n, then this 3-SAT problem can be satisfied. Thus 3-SAT can be transformed into Experimental Cuisine problem. When solving the EXPERIMENTAL CUISINE problem, for an ingredient, if it is added to the selection and the size of p does not exceed the set value, then it can be selected, otherwise it cannot. Each judgment costs polynomial time, and the result can be obtained after a polynomial number of judgments, then the whole process can be completed in polynomial time. Therefore,in conclusion, if EXPERIMENTAL CUISINE is solvable in polynomial time, so is 3-SAT.

□

2. An induced subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is a graph that satisfies $V' \subseteq V$ and $E' = \{(u, v) \in E | u, v \in V'\}$. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and an integer b, we need to decide whether G_1 and G_2 have a common induced subgraph G_c with at least b nodes. This problem is called Maximum Common Subgraph (MCS). Prove that MCS is NP-complete. (Hint: reduce from INDEPENDENT-SET)

Solution. First note that the maximum common subgraph problem is in the class of NP. To verify a given solution $V_1^{'}$ and $V_2^{'}$ along with a proposed isomorphism mapping $V_1 \setminus V_1^{'}$ to $V_2 \setminus V_2^{'}$, we will simply need to delete these sets of vertices and their incident edges from the graphs G_1 and G_2 and check whether the proposed isomorphism is indeed bijective, which can be done in $O(n^2)$ time. This is an example of a problem with a less trivial step to prove that it is in NP, and you will not be penalized if your solution is left somewhat informal. To prove NP-hardness, we reduce from the clique problem. In the clique problem, we are given a graph $G = (V, E)$ and a goal g, and are asked to find a clique of size g in G. A clique of size g is a set g vertices such that there is an edge between every pair of vertices in the set. Given an instance of the clique problem, we reduce it to an instance of the MCSP as follows: create a complete graph G_0 which has the same vertices as G but has an edge between all pairs of vertices. The two input graphs to our maximum common subgraph problem are G and G_0, Suppose that $V_1^{'} \subseteq G$ and $V_2^{'} \subseteq G_0$ is a solution to the MCSP on G and G_0 . If $|V|-|V_1^{'}| \geq g$, then there is a clique of size g in G. Since G_0 is a complete graph, any subgraph must also be a complete graph,and thus a clique. So if there is a common subgraph of G and G_0 of size at least g, then this must be a clique of size at least g. In particular, $V \setminus V_1^{'}$ is a clique of size at least g and one can trivially throw away vertices to get a clique of size exactly g. Otherwise, there is no clique of size g in G. On the other hand, we suppose that there is a clique V_0 of size g in G. Then clearly it forms a common subgraph of size g with any subgraph of G_0 of size g. The reduction is in polynomial time because it just requires creating a complete graph with a polynomial number of edges and vertices with respect to G. So in conclusion, MCS is NP-complete.

□

3. Let us define the k-spanning tree as a spanning tree in which each node has a degree $\leqslant k$. Given a graph $G = (V, E)$ and a positive integer k, we need to decide whether there exists a k-spanning tree in G. Prove that this problem is NP-complete. (Hint: reduce from HAMILTONIAN-CYCLE)

Solution. First, we note that 2-SPANNING-TREE is exactly undirected HAMILTONIAN-PATH: a tree in which each vertex has degrees at most 2 is a path. Thus, 2-SPANNING-TREE is NP-hard. Now, we give a reduction from 2-SPANNING-TREE to k-SPANNING-TREE for any $k \geq 3$, which exactly means reduction from HAMILTONIAN-CYCLE. Given a graph $G(V, E)$ as an instance of 2-SPANNING-TREE, we construct another graph $G^{'}\left(V^{'}, E^{'}\right)$ where $V^{'}$ contains V and some other vertices as follows. For each vertex $v \in V$, $V^{'}$ also contains $k-2$ new vertices $v_1, v_2, ...v_{k-2}$. Similarly, $E^{'}$ contains E and some extra edges connecting v_i and v. Formally,

$$V^{'} = V \bigcup \{v_1, v_2, ...v_{k-2} | v \in V\} \tag{1}$$

$$E^{'} = E \bigcup \{e_1, e_2, ...e_{k-2} | e \in E\} \tag{2}$$

We will claim that G has a 2-SPANNING-TREE if and only if G_0 has a k-SPANNING-TREE. On one hand, we assume that G has a 2-SPANNING-TREE T, then $T^{'} = T \bigcup \{v_i v | v \in V, i = 1, 2...k-2$ is a spanning tree of $G^{'}$. Since each vertex of T has degree at most 2, each vertex of $T^{'}$ has degree at most k. On the other hand, assume that $G^{'}$ has a k-SPANNING-TREE $T^{'}$. One can see that $T^{'}$ must contains all the edges $v_i v$, since those are the only edges incident at v_i . Furthermore, all v_i ' s are leaves of $T^{'}$ since they all have degree 1. Thus we can remove them to obtain a spanning tree T of G. After removing all edges $v_i v$ decrease the degree

of each v by $k-2$, T is a 2-SPANNING-TREE of G. In conclusion, k-SPANNING-TREE is NP-complete for any $k \geq 2$.

□

4. We define the decision problem of KNAPSACK PROBLEM as follows:

   Given n indivisible objects, each with a weight of $w_i > 0$ kilograms and a value $v_i > 0$, a knapsack with capacity of W kilograms and a number k, decide whether there is a collection of objects that can be put into the knapsack with a total value $V \geqslant k$.

   Prove that KNAPSACK PROBLEM is NP-complete.

   Solution. First of all, we will proof that Knapsack is NP. The proof is the set S of items that are chosen and the verification process is to compute $\sum_{i \in S} s_i$ and $\sum_{i \in S} v_i$, which takes polynomial time for the size of input.
   Secondly, we will show that there is a polynomial reduction from Partition problem to Knapsack, thus leading to our problem, determining Knapsack to be NPC. It suffices to show that there exists a polynomial time of reduction $Q(\cdot)$ such that $Q(X)$ is a 'Yes' instance to Knapsack if and only if X is a 'Yes' instance to Partition. Suppose we are given $a_1, a_2, ..., a_n$ for the Partition problem, consider the following Knapsack problem: $s_i = a_i$, $v_i = a_i$ for $i = 1, ..., n$, $B = V = \frac{1}{2}\sum_{i=1}^{n} a_i$. $Q(\cdot)$ here is the process converting the Partition problem to Knapsack problem. It is clear that this process is polynomial in the input size. If X is a 'Yes' instance for the Partition problem, there exists S and T such that $\sum_{i \in S} a_i = \sum_{i \in T} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i$. We then let our Knapsack contain the items in S, and it follows that $\sum_{i \in S} s_i = \sum_{i \in S} a_i = B$ and $\sum_{i \in S} v_i = \sum_{i \in S} a_i = V$. Therefore, $Q(X)$ is a 'Yes' instance for the Knapsack problem.
   Conversely, if $Q(X)$ is a 'Yes' instance for the Knapsack problem, with the chosen set S, let $T = \{1, 2, ...n\}$. We have $\sum_{i \in S} s_i = \sum_{i \in S} a_i \leq B = \frac{1}{2}\sum_{i=1}^{n} a_i$ and $\sum_{i \in S} v_i = \sum_{i \in S} a_i \leq V = \frac{1}{2}\sum_{i=1}^{n} a_i$. This means that $\sum_{i \in S} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i$ and $\sum_{i \in T} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i$ Therefore, $\{S, T\}$ is the desired partition, and X is a 'Yes' instance for the Partition problem. In conclusion, This can proof that Knapsack problem is NPC.

   □

Remark: Please include your .pdf, .tex files for uploading with standard file names.