Lab11-NP Reduction

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

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

∗ Name: Log Creative Student ID: Email: logcreative-lzl@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 $\leq p$

Prove that 3-SAT $\leq_p$ Experimental Cuisine

Proof. It is required to prove Independent-Set $\leq_p$ Experimental Cuisine, since 3-SAT $\leq_p$ Independent-Set. $\leq_p$ satisfies transitivity so that

$$\text{3-SAT} \leq_p \text{Independent-Set} \leq_p \text{Experimental Cuisine}$$

Given an instance Φ of Independent-Set with a graph $G = (V, E)$ and an integer k, construct an instance of Experimental Cuisine:

Label vertices in V as $v_1, v_2, \cdots, v_n$. Define $n \times n$ discord matrix S where the element at (i, j) is

$$S(i,j) = \begin{cases} 0, & i = j \text{ or } (v_i, v_j) \text{ in } G \text{ is not adjacent,} \\ 1, & (v_i, v_j) \text{ in } G \text{ is adjacent.} \end{cases}$$

Then, it is required to prove G has an independent set whose size $\geq k$ iff there exists a collection of at least k ingredients that has a penalty ≤ 0 with the definition of S.

$\Leftarrow$: If there exists a collection of at least k ingredients that has a penalty ≤ 0 with the definition of S, then all pairs of ingredients fit well, where the penalty is 0 (Otherwise, the total penalty must be larger than 0). the definition of S, the corresponding subset of vertices with the same label has the property that no two vertices in this subset is adjacent. Thus a independent set.

$\Rightarrow$: If G has an independent set whose size $\geq k$, then corresponding cuisine can go well (=0) for every pair, since they are not adjacent. Thus, the collection of such cuisine satisfies penalty ≤ 0.

□

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)

   Proof. Firstly, show that Maximum Common Subgraph is in NP. Algorithm **??** that the procedure to find the maximum common subgraph for G_1 and G_2 in a certification way.

Algorithm 1: Maximum Common Subgraph

```
Input: Graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$
Output: Maximum Common Subgraph $G_c = (V_c, E_c)$
1 $V \leftarrow V_1 \cap V_2$;
2 $V_c \leftarrow \varnothing, E_c \leftarrow \varnothing$;
3 foreach subset $V' \subseteq V$ do
4     induced subgraph $G'_1 \leftarrow (V', E'_1)$ where $E'_1 \subseteq E_1$;
5     induced subgraph $G'_2 \leftarrow (V', E'_2)$ where $E'_2 \subseteq E_2$;
6     if $G'_1 = G'_2$ and $|V'| > |V_c|$ then
7         $G_c \leftarrow G'_1 = G'_2$;
8 return $G_c = (V_c, E_c)$;
```

Assumming that V has n vertices, then 2^n subsets are required to search. The certifier from Line **4** to **7** is $O(|E_1| + |E_2|)$, which is of poly-time. So Maximum Common Subgraph is in NP.

Then, it is required to prove that Independent-Set $\leq_p$ Maximum Common Subgraph, since Independent-Set is NP-complete.

Given an instance Φ of Independent-Set with a graph $G = (V, E)$ and an integer k. It is required to prove that $G = (V, E)$ has an independent set of size k iff $G_1 = (V, E)$ and $G_2 = (V, \varnothing)$ have a common subgraph of vertex size k.

$\Leftarrow$: If $G = (V, E)$ and $G' = (V, \varnothing)$ have a maximum common subgraph of vertex size k, assume that such a maximum common subgraph is $G'' = (V'', \varnothing)$, then no two vertices in G'' are adjacent since there are no edges in the graph. Because $G'' \subseteq G \cap G' \subseteq G = (V, E)$, G has an independent subgraph G' with vertex size k.

$\Rightarrow$: If $G = (V, E)$ has an independent set of size k, which is G'', then G'' is the common subgraph of $G = (V, E)$ and $G' = (V, \varnothing)$ for a similar reason, which has at least k nodes.

□

3. Let us define the k-spanning tree as a spanning tree in which each node has a degree $\leq k$. 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)

Algorithm 2: k-Spanning Tree

```
Input: Graph $G = (V, E)$ and a positive integer $k$
Output: $k$-Spanning Tree $T$
1  if $|E| < |V| - 1$ then return No spanning tree;
2  foreach edge set $E' \subseteq E$ with $|E'| = |V| - 1$ do
3      if no loop in $G' = (V, E')$// By DFS
4       then
5          flag $\leftarrow$ false;
6          foreach $v \in V$ do
7              if $\deg v > k$ then
8                  flag $\leftarrow$ true;
9                  break;
10         if flag =false then return $G'$;
11 return No k-spanning tree;
```

Proof. Firstly, show that K-SPANNING TREE is in NP. Algorithm **??** shows K-SPANNING TREE how to certificate whether the tree is is a k-spanning tree.

The certifier from Line **5** to **10** is of $O(|V| + |E|)$, which is of poly-time. So K-SPANNING TREE is in NP.

Then, it is required to prove that HAMILTONIAN-CYCLE $\leq_p$ K-SPANNING TREE, since HAMILTONIAN-CYCLE is NP-complete.

Given an instance Φ of HAMILTONIAN-CYCLE with an undirected graph $G = (V, E)$. It is required to prove that there exists a simple cycle Γ in $G = (V, E)$ that contains every node in V iff $G' = (V, E - E')$ has a 2-spanning tree where $E' \neq \varnothing$ only contains all edges from connected vertices v_i and v_j in G.

$\Leftarrow$: if $G' = (V, E - E')$ has a 2-spanning tree T, then T is in fact a hamitonian path. Then add one edge in E' will give a hamitonian cycle of $G = (V, E)$.

$\Rightarrow$: if there exists a simple cycle Γ in $G = (V, E)$ that contains every node in V, then remove the edge in E' will give Γ'. If Γ didn't use any edge in E', then remove any one edge in Γ' will give a 2-spanning tree of G'. Otherwise Γ' is a 2-spanning tree of G'.

2-ST $\leq_P$ k-ST required.$(G' = (V', E'))$

□

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 $V \geq k$.

   Prove that KNAPSACK PROBLEM is NP-complete.

Proof. Firstly, show that KNAPSACK PROBLEM is NP. Algorithm shows a searching way of finding a solution to KNAPSACK PROBLEM.

The certifier of Line **2** is of poly-time. So KNAPSACK PROBLEM is in NP.

Then, it is required to prove that SUBSET SUM $\leq_p$ KNAPSACK PROBLEM, since SUBSET SUM is NP-complete.

Algorithm 3: Knapsack Problem

Input: n indivisible objects, each with weight of $w_i > 0$ kilograms and a value $v_i > 0$. Knapsack capacity W and a value target k

Output: a collection of objects that can be put into the knapsack with a total vale $V \geq k$

```
1 foreach subset $S$ of $n$ objects do
2     if $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq k$ then return $S$;
3 return No solution;
```

Given an instance of Subset Sum with natural numbers $w_0, w_1, \cdots, w_n$ and an integer W. The corresponding Knapsack Problem is n divisible objects, each with weight of w_i and the same number of value w_i, with knapsack capacity W and a value target W. To choose a subset S such that

$$\begin{cases} \sum_{i \in S} w_i & \leq W \\ \sum_{i \in S} w_i & \geq W \end{cases}$$

In other word, $\sum_{i \in S} w_i = W$. They are equivilent. As a result, Subset Sum $\leq_p$ Knapsack Problem. $\square$

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