Assignment 2

ID: 120037910002 Name: Xingguo Jia Email: jiaxg1998@sjtu.edu.cn

1. STINGY SAT is the following problem: given a set of clauses(each a disjunction of literals) and an integer k, find a satisfying assignment in which at most k variables are true, if such an assignment exists. Prove that STINGY SAT is NP-complete.

Proof.

- First we prove STINGY SAT is NP. Given an assignment, we can count the number of true variables. If the number is at most k, then such an assignment exist. This runs in $O(l)$, where l is the number of clauses. Thus, we can verify STINGY SAT in polynomial time, and it is NP.
- Then we reduce SAT to STINGY SAT. Given a set of l clauses, once we find a satisfying assignment in which at most $k = l$ variables are true, then we solve the SAT problem. If we solve STINGY SAT, then we could solve SAT by setting k to l. Since SAT is **NP** and the reduction takes polynomial time, STINGY SAT is **NP-complete**.

□

2. You are given a directed graph $G = (V, E)$ with weights we on its edges $e \in E$. The weights can be overlineative or positive. The ZERO-WEIGHT-CYCLE PROBLEM is to decide if there is a simple cycle in G so that the sum of the edge weights on this cycle is exactly 0. Prove that this problem is NP-complete.

Proof.

- The problem is NP since given a simple cycle in $G = (V, E)$, we can verify whether the sum of the edge weights on this cycle is exactly 0 by adding all $|E|$ edge weights together, which costs $O(|E|)$ (polynomial) time.
- We reduce **subset-sum** problem to **zero-weight-cycle** problem, and since subset-sum problem is **NP**, zero-weight-cycle problem is **NP-complete**. For a set of number $a_i(i = 1, \ldots, n)$, decide whether there exists a subset whose sum is 0.
- See Figure 1, each vertex $u_i(i = 1, \ldots, n)$ has only one outgoing edge $u_i \to v_i$ with weight a_i, and each vertex $v_i(i = 1, \ldots, n)$ has n outgoing edges to each $u_k(k = 1, \ldots, n)$ with weight 0. Any simple cycle in this graph has the form
- $u_{i_1} \to v_{j_1} \to u_{i_2} \to v_{j_2} \to \cdots \to u_{i_k} \to v_{j_k} \to u_{i_1} (1 \leq k \leq n)$
- which has a weight sum of $\sum_{j=1}^{k} a_{i_j} (1 \leq k \leq n)$. This is a subset sum of set $\{a_i \mid i = 1, \ldots, n\}$. Thus, we could reduce the subset sum problem to zero-weight-cycle problem in $O(n)$ (polynomial) time.

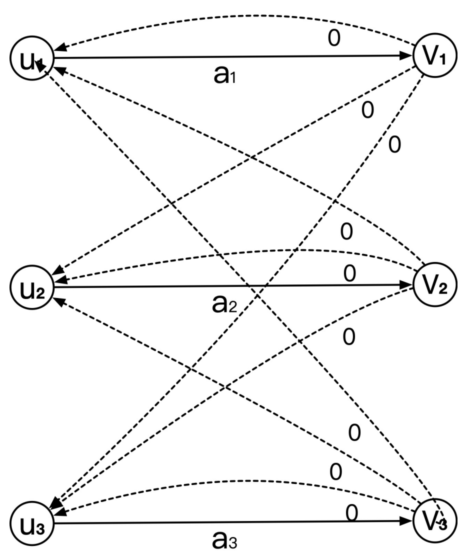

Figure 1: Reduction from subset-sum problem

□

3. Show that INDEPENDENT SET PROBLEM is NP-hard even graphs of maximum degree 3.

Proof.

- We reduce the **3-SAT** problem to **independent set** problem for a graph G.
- For each clause in the formula, we use a vertex in G to represent each literal of the clause, and add an edge between each of them. For each pair of opposite vertices between different clauses, add an edge between them. This reduction runs in $O(|V| + |E|)$ (polynomial) time.
- For example, Figure 2 shows G equivalent to formula ($-y$ stands for $\overline{y}$)

$$(x \vee y \vee z)(x \vee \overline{y})(y \vee \overline{z})(z \vee \overline{x})(\overline{x} \vee \overline{y} \vee \overline{z})$$

- For each clause $C_i (i = 1, \ldots, n)$, we choose a v_i in G. If all v_i forms an independent set, then we set the literal to $true$ corresponding to v_i and no opposite literals are set to $true$ at the same time. Thus we reduce **3-SAT** problem to **independent set** problem.

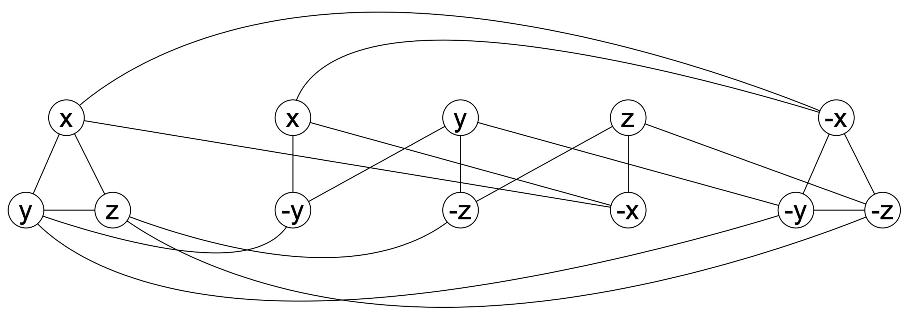

Figure 2: Reduction from 3—SAT problem

- But in this graph, the maximum degree can be larger than 3. We modify the formula to a **equivalent** one in the following way: if variable x appears in $k \geq 3$ clauses, then in G, vertices corresponding to x may have a degree ≥ 4. For variable x that appears in $k \geq 3$ clauses, modify the k x variables to $x_1, x_2, \ldots, x_k$, and

$$F' = F \wedge U, U = (x_1 \vee \overline{x_2})(x_2 \vee \overline{x_3}) \ldots (x_k \vee \overline{x_1}), k \geq 3$$

- To satisfy U, we have $x_1 = x_2 = \ldots x_k$. Thus, if we could find a satisfying assignment for F', then we could also find a satisfying assignment for F by modifying all x_k to x. Then we could translate F' into graph G' with maximum degree of 3.

$\square$

4. For your new startup company, Uber for algorithms, you are trying to assign projects to employees. You have a set P of n projects and a set of E of m employees. Each employee e can only work on one project, and each project $p \in P$ has s subset $E_p \subseteq E$ of employees that must be assigned to p to complete p. The decision problem we want to solve is whether we can assign the employees to projects such that we can complete(at least) k projects.

- Give a straightforward algorithm that checks whether any subset of k projects can be completed to solve the decisional problem. Analyze its time complexity in terms of m, n and k.
- Show that the problem in NP-hard via a reduction from 3D-matching.

Proof.

- Given one of $\binom{n}{k}$ subsets of P, we mark all $employee \in E_p \subseteq E$ to be **employed**. If any $employee$ is marked more than once, then this subset of P cannot be completed. If all $employee$s are not marked more than once, then this subset of P can be completed. This algorithm has $O(m * \binom{n}{k})$ time complexity.
- ***3D-matching***: *let X, Y, Z be finite sets, $T = X \times Y \times Z$, positive integer k. We need to decide whether there exists a set $M \subseteq T$ such that $|M| \geq k$ and all triples in M are disjoint(for $(x_1, y_1, z_1), (x_2, y_2, z_2)$, $x_1 \neq x_2, y_1 \neq y_2, z_1 \neq z_2$). This is proved to be* ***NP-complete***.
- We reduce 3D-matching to assigning projects problem. Suppose we could solve the assigning project problem, and we reduce 3D-matching to a special case of it. For each $p \in P$, $E_p = \{e_1, e_2, e_3\} \subseteq E$ employees must be assigned to complete p, where $e_i \in E_i, i = 1, 2, 3$ and $E_1 \cup E_2 \cup E_3 = E, E_1 \cap E_2 = \varnothing, E_2 \cap E_3 = \varnothing, E_3 \cap E_1 = \varnothing$.
- If we could find an assignment to complete k or more projects, then we could find a subset $M \subseteq E_1 \times E_2 \times E_3, |M| \geq k$ that all triples in M are disjoint. The reduction takes polynomial time. Thus, project assignment problem is **NP-hard**.

$\square$

5. Let $d \in \mathbb{N}$. The d-COLORABILITY PROBLEM is to decide whether a given graph $G = (V, E)$ can be colored by d colors. i.e., whether there exists a function $f : V \to \{1, 2, \ldots, d\}$ such that for every $u, v \in V$ with $(u, v) \in E$ we have $f(u) \neq f(v)$. Formulate d-COLORABILITY as a search problem. Give a reduction from 4-COLORABILITY to 7-COLORABILITY.

Reduction:

- Suppose we have an arbitrary graph $G = (V, E)$. We add three vertices x, y, z to V, and add all edges between $\{x, y, z\}$ and V:

$$N = \{x, y, z\}, V' = V \cup N, E' = E \cup \{(u, v) \mid u \in N, v \in V'\}, G' = (V', E')$$

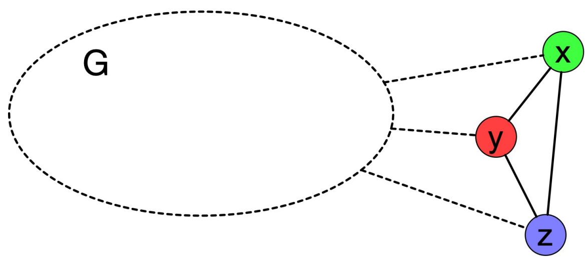

Figure 3: Reduction from 4-COLORABILITY to 7-COLORABILITY

- See Figure 3, suppose we have a function $f' : V' \to \{1, 2, 3, 4, 5, 6, 7\}$ for the graph G', then we have three different colors for x, y, z (suppose the three colors are $1, 2, 3$). Since all vertices in V are connected to x, y, z, they must have colors different from $1, 2, 3$, which must be $4, 5, 6, 7$. Thus, we find $f : V \to \{4, 5, 6, 7\}$ for G, and G is 4-COLORABLE.

6. In the MAX CUT problem, we are given an undirected graph G and an integer K and have to decide whether there is a subset of vertices S such that there are at least K edges that have one endpoint in S and one endpoint in $\bar{S}$. Prove that this problem is NP-complete.

Proof.

- Given a set of vertices S, we can get the number of edges that have one endpoint in S and one endpoint in $\bar{S}$ in $O(E)$ time by traversing all edges in G, and compare the number to K. Thus the **max cut** problem is NP.

- We reduce **independent set** to **max cut** to prove NP-complete. For a graph $G = (V, E)$, we have $G' = (V', E')$ where

$$V' = V \cup \{x\} \cup \{[e, u], [e, v] | e = (u, v) \in E\}$$

$$\begin{cases} E' = E_1 \cup E_2 \\ E_1 = \{(x, v) \mid v \in V\} \\ E_2 = \{(x, [e, u]), (x, [e, v]), ([e, u], [e, v]), ([e, u], u), ([e, v], v) \mid e = (u, v) \in E\} \end{cases}$$

$$k' = k + 4 * |E|$$

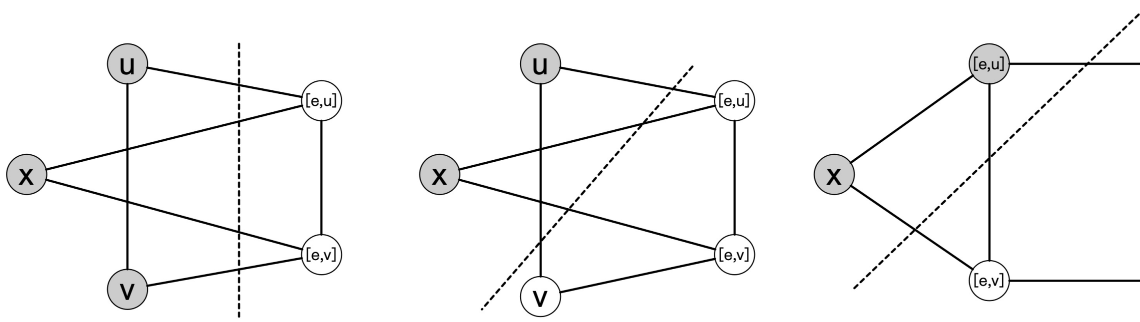

Figure 4: Reduction from independent set to max cut

- We run **max cut** on $G' = (V', E')$, and find the subset of vertices $S' \subseteq V'$, such that for the set of edges $E_0 \subseteq E'$ that have one endpoint in S' and one endpoint in $\bar{S'}$, $|E_0| \geq k'$.

- See Figure 4, for all edges in E, the added edges in E_2 can be 3 or 4, and

$$k + 4 * |E| - |E_0 \cap E_1| \leq |E_0| - |E_0 \cap E_1| = |E_2 \cap E_0| \leq 4 * |E|$$

$$\Rightarrow |E_0 \cap E_1| \geq k \Rightarrow |E_0 \cap E_1| = k + l, l \geq 0$$

- Then we have l edges $e \in E$ that added 3 edges in E_2, which form $E_v \subseteq E$. Remove one of the endpoints of $e_v \in E_v$ from V, then we have $S \subseteq V$. We say S is a independent set of G and $|S| \geq k$, since no edge has more than one endpoints in S, and $|E_0 \cap E_1| = k + l \Rightarrow |S| \geq k$.

$\square$

7. Let QUADEQ be the language of all satisfiable sets of *quadratic equations* over $0/1$ variables(a quadratic equations over $u_1, \ldots, u_n$ has the form $\sum_{i,j\in[n]} a_{i,j}u_iu_j = b$)where addition is modulo 2. Show that QUADEQ is NP-complete.

Proof.

- Firstly, given $0/1$ variables $u_1, \ldots, u_n$, we can test whether equation $\sum_{i,j\in[n]} a_{i,j}u_iu_j = b$ holds in $O(n(n-1)/2)$ time. Thus, **QUADEQ** is NP.
- Then we reduce **clique** to **QUADEQ**. In graph $G = (V, E), V = \{v_1, v_2, \ldots, v_n\}$, let

$$a_{i,j} = \begin{cases} 0, if\ (v_i, v_j) \notin E \\ 1, if\ (v_i, v_j) \in E \end{cases}, \quad b = k(k-1)/2, \ \ 1 \leq k \leq n$$

- If we could decide whether equation $\sum_{i,j\in[k]} a_{i,j}u_iu_j = b$ is satisfiable, then we have

$$u_1 = u_2 = \ldots = u_k = 1$$

- and $\{v_1, v_2, \ldots, v_k\} \subseteq \{v_1, v_2, \ldots, v_n\}$ to be a **clique** in G. Thus **QUADEQ** is NP-complete.

□

8. In a typical auction of n items, the auctioneer will sell the ith item to the person that gave it the highest bid. However, sometimes the items sold are related to one another(e.g., think of lots of land that may be adjacent to one another)and so people may be willing to pay a high price to get, say, the three items $\{2, 5, 17\}$, but only if they get all of them together. In this case, deciding what to sell to whom might not be an easy task. The COMBINATORIAL AUCTION PROBLEM is to decide, given numbers n, k, and a list of pairs $\{(S_i, x_i)\}_{i=1}^{m}$ where S_i is a subset of $[n]$ and x_i is an integer, whether there exist disjoint sets $S_{i_1}, \ldots, S_{i_l}$ such that $\sum_{j=1}^{l} x_{i_j} \geq k$. That is, if x_i is the amount a bidder is willing to pay for the set S_i, then the problem is to decide if the auctioneer can sell items and get a revenue of at least k, under the obvious condition that he can't sell the same item twice. Prove that COMBINATORIAL AUCTION is NP-complete.

Proof.

- Given sets $S_{i_1}, \ldots, S_{i_l}(l \leq m)$, we can calculate $\sum_{j=1}^{l} x_{i_j}$ in polynomial time $O(m)$ and compare to k. Also, we can check whether these sets are disjoint in $O(n * m)$ time. Thus, the problem is NP.
- Then we reduce **independent set** problem to **combinatorial auction** problem. In graph G, all edges are marked from 1 to n, where $n = |E|$. For each vertex $v_i \in V$, all edges connected to v form set S_i. Suppose we could found disjoint sets $M = \{S_{i_1}, \ldots, S_{i_l}\}$ such that $\sum_{j=1}^{l} x_{i_j} \geq k$, where $x_{i_j} = 1$. Then we have $\sum_{j=1}^{l} 1 = |M| \geq k$, which is the number of disjoint edge sets. Thus we solve the independent set problem, and the reduction takes polynomial time. Thus, COMBINATORIAL AUCTION is NP-complete.

$\square$