ShanghaiTech University

CS240 Algorithm Design and Analysis

Fall 2022

Problem Set 3 Answer

Due: 23:59, Dec. 18, 2022

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In "Account Settings" of Gradescope, set your FULL NAME to your Chinese name and enter your STUDENT ID correctly.
3. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
4. When submitting your homework, match each of your solution to the corresponding problem number.

Problem 1:

Suppose there is a finite set C and a collection of subsets of C. The SET-PACKING problem asks if some K subsets in the collection are pairwise disjoint(in other words, no two of them share an element). Show that SET-PACKING problem is NP-complete.
Hint: Reduction from Independent Set.

Solution:

1. Given a set of K subsets in the collection, there exists a poly-time certifier that checks whether: all $\frac{1}{2}K(K\text{-}1)$ pairs in subsets are pairwise disjoint. Therefore SET-PACKING problem is in NP.

2. Known that Independent Set is NP-complete, we will show that SET-PACKING problem is NP-complete by a polynomial-time reduction from Independent Set. Given an instance of Independent Set, we will construct an instance of SET-PACKING problem. W.L.O.G assuming that the graph G = (C, E). For every node $c_i \in$ C, we define a subset S(c_i) to be the set of all edges incident with c.

Here the subsets S(c) corresponds to the collection of subsets of C. We notice that two subsets S(c_1) and S(c_2) are disjoint if and only if c_1 and c_2 are not adjacent. Therefore some K subsets in the collection are pairwise disjoint if and only if we can find K pairwise disjoint subsets S(c_i) from S(c).

3. Independent Set is NP-complete and Independent Set p SET-PACKING problem, hence SET-PACKING problem is NP-complete

Problem 2:

Red and Xiaoyu are allocating a set of items $M = \{1, ..., m\}$ among themselves. Each of them evaluates the items respectively and gives each of the items a valuation to denote their preference on the item. Given a subset of items $S \subseteq M$, one's utility is defined as

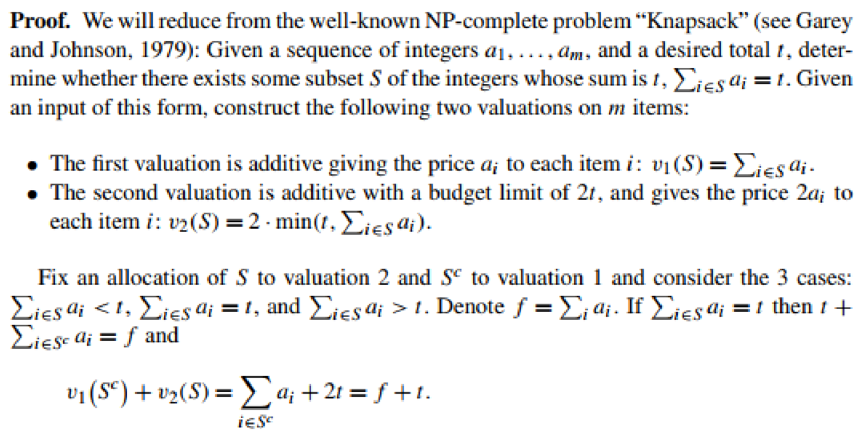
$$u_i(S) = max\{\sum_{j \in S} v_i(j), b_i\}$$

where $i \in \{Red, Xiaoyu\}$, $v_i(j)$ is i's valuation on item j and b_i is the upper bound of i's utility. Their goal is to find the optimal allocation S_1, S_2 that maximizes $u_R(S_1) + u_X(S_2)$. Prove this problem is NP-hard with "Knapsack".

Here is an example for you to understand the problem. There are 3 items indexed by $1, 2, 3$. Red's valuation on the items are $10, 20, 30$ and Xiaoyu's valuation is $30, 15, 10$ while the upper bound of their total utilities are 40 and 50. Then the best choice is to allocate item 3 to Red and $1, 2$ to Xiaoyu which brings a $30 + (30 + 15) = 75$ utility in total.

Solution:

While the "Knapsack" in the following answer is more similar to the problem "subset sum", so answers using 2-Knapsack or other methods will also be regarded as correct.



Proof. We will reduce from the well-known NP-complete problem "Knapsack" (see Garey and Johnson, 1979): Given a sequence of integers $a_1, \dots, a_m$, and a desired total t, determine whether there exists some subset S of the integers whose sum is t, $\sum_{i \in S} a_i = t$. Given an input of this form, construct the following two valuations on m items:

- The first valuation is additive giving the price a_i to each item i: $v_1(S) = \sum_{i \in S} a_i$.
- The second valuation is additive with a budget limit of $2t$, and gives the price $2a_i$ to each item i: $v_2(S) = 2 \cdot \min(t, \sum_{i \in S} a_i)$.

Fix an allocation of S to valuation 2 and S^c to valuation 1 and consider the 3 cases: $\sum_{i \in S} a_i < t$, $\sum_{i \in S} a_i = t$, and $\sum_{i \in S} a_i > t$. Denote $f = \sum_i a_i$. If $\sum_{i \in S} a_i = t$ then $t + \sum_{i \in S^c} a_i = f$ and

$$v_1(S^c) + v_2(S) = \sum_{i \in S^c} a_i + 2t = f + t.$$

If $\sum_{i \in S} a_i < t$ then

$$v_1(S^c) + v_2(S) = \sum_{i \in S^c} a_i + 2\sum_{i \in S} a_i = f + \sum_{i \in S} a_i < f + t.$$

If $\sum_{i \in S} a_i > t$ then $\sum_{i \in S^c} a_i + t < f$, and

$$v_1(S^c) + v_2(S) = \sum_{i \in S^c} a_i + 2t < f + t.$$

Thus we see that the auction has an allocation $(S^c : S)$ with value $f + t$ if the knapsack problem has a positive answer S, and otherwise the allocation has a lower value. $\square$

Problem 3:

The problem: Suppose there is an undirected graph $G = (V, E)$ and a positive integer $M \leq |V|$. Does the gragh G contain a path which visits vertex at most once and has the number of path's edges $N \geq M$?
Prove this problem is NP-complete.

Solution:

Firstly show this problem is in NP: Given an instance G, we check a set of edges of size at least M and examine if it is a path which visits vertex at most once in G. This can be done in polynomial time.

Then do reduction from Hamiltonian Path. Given an instance G of Hamiltonian Path, we create an instance graph G' and integer M' of this problem as follows: Take $G' = G$ and set $M' = |V| - 1$. Then there exists the path which visits vertex at most once of length M' in G' if and only if G contains a Hamiltonian path.

Problem 4:

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 a true, if such an assignment exists. Prove that STINGY SAT is NP-complete.

Solution:

NP: STINGY SAT can be verified a true assignment in polynomial time, so it's NP.

Reduction: If there is a SAT with k variables, it can be transformed to STINGY SAT, which only has k variables and k of STINGY SAT is k. Given a SAT formula f with k variables we simply chose (f, k) as an instance of Stingy SAT.

$\Rightarrow$ Suppose that f is a yes-instance of SAT. No more than k variables can be true, because there are a total of k variables. So any satisfying assignment of instance f for SAT will be a satisfying assignment of instance (f, k) for Stingy SAT. So (f, k) is a yes-instance of Stingy SAT.

$\Leftarrow$ Suppose that (f, k) is a yes-instance of Stingy SAT. Any satisfying assignment of that instance will be also a satisfying assignment of instance f for SAT. So f is a yes-instance of SAT.

Because SAT is NP-Complete, the STINGY SAT is NP-Complete.

Problem 5:

SIST allows students to work as TAs but would like to avoid TA cycles. A TA cycle is a list of TAs (A_1, A_2, . . . , A_k) such that A_1 works as a TA for A_2 in some course, A_2 works as a TA for A_3 in some course, $\cdots$, and finally A_k works as a TA for A_1 in some course. We say a TA cycle is simple if it does not contain the same TA more than once. Given the TA arrangements of SIST, we want to find out whether there is a simple TA cycle containing at least K TAs. Prove this problem is NP-complete.

Solution:

NP problem: We can just check whether there is a simple TA cycle containing k TAs in plolynomial time, so it's NP.

Reduction: Directed Hamiltonian Cycle $\leq_p$ this problem.

Construction: We construct graph G = (V, E) and TA cycle C, where each vertex V_i represents each TA A_i, and a directed path from V_i to V_j represents A_i works as a TA for A_j in some course. We let $K = |V|$. Then we prove that G is a Directed Hamiltonian Cycle iff C contains a simple TA cycle.

$\Rightarrow$: If G is a Directed Hamiltonian Cycle with vertex $|V|$, then there exists a simple TA cycle containing $K = |V|$ TAs.

$\Leftarrow$: If C is a TA cycle containing a simple TA cycle with K TAs, then there exists a Directed Hamiltonian Cycle.

So the problem is NP-complete.