**CSC505 HW5 Problem 1-3**

**Yiwei Li**

**Zirun Han**

1. Purpose. Understanding NP-completeness proofs and special cases of NP-complete problems.

The min-ones satisfiability problem (M1-Sat) is defined as follows.

Given a formula F in conjunctive normal form and an integer k, does there exist a satisfying assignment A for F such that the number of true variables in A is ≤ k?

(a) Prove that M1-Sat is NP-complete. Be sure to include and clearly mark all parts of the proof, even those that may seem trivial.

(b) Let p be the number of variables that have at least one positive occurrence – a variable x has a positive occurrence if some clause contains x as a literal (as opposed to x = ¬x). Give a O(n) algorithm for solving M1-Sat when p is constant, where n is the total number of literals in F.

**Answer:**

1a. To prove the M1-Sat is NP complete, we need to satisfy two requirements.

First requirement: Showing the M1-Sat is in NP.

We will non-deterministically choose all the variable true or false in the assignment. After getting this polynomial-sized certificate, we can evaluate the assignment is suitable for the formula F or not. This evaluation will take polynomial time. Then, we will traverse all the variables and count the number of true variables. Comparing the number of true variables with the integer k, we could come up the decision result. Traversing, counting and verifying also takes polynomial time. So that M1-Sat is in NP. The pseudocode is listed as follows.

Pseudocode:

M1-Sat(X,C,k):

for each xi in X

non-deterministically to choose whether xi is true or false

for each clause ci in C

if none of the terms in ci are true

reject

for each xi in X

if xi = true

count +=1

if count > k

reject

accept

Second requirement: Reduce SAT to M1-Sat. According to book Theorem 34.9, we know SAT is NP complete.

To prove we can reduce SAT to M1-Sat, we need to show that (1) transformation from SAT to M1-Sat takes polynomial time; (2) The answers are the same for SAT and M1-Sat. That is, the answer for SAT is “yes” if and only if the answer for M1-Sat is also “yes”.

1. Transformation from SAT to M1-Sat is easy. We can simply use one instance which from the SAT formula(f) with k variables as one instance of M1-Sat (F).
2. If m is one of the truth assignment of SAT, m will be no larger than k. Cause we only have k variables in f. Since this same instance is transformed as the instance in F, we also know that this truth assignment has no more than k true variables. We know that this m is also a truth assignment for M1-Sat.

SAT formula f is an instance of M1-Sat. So if n is a truth instance of M1-Sat, n will also be able to satisfy f. n is a truth instance for SAT.

1b.

1. Do Problem 34-4, parts (a) and (b) on page 1104. The problem statement is identical to that of Problem 1 of Homework 3, but there is no restriction on the tj’s. Part (a) asks you to formulate the problem as a decision problem; part (b) asks for an NP-completeness proof. You already did parts (c) and (d) in Homework 3.

**Answer:**

2a. The decision problem will be: Can we find a sequence of tasks that all of them can be completed before their deadlines and return the total profit no less than k.

2b. To prove the decision problem is NP complete, we need to satisfy two requirements.

First requirement: Showing this problem in NP.

We will sort all the ai in A by their deadlines dj. Then non-deterministically choose whether ai is scheduled or not and add scheduled tasks in the sequence S. After getting this polynomial-sized certificate, we can evaluate the sequence of scheduled tasks with our decision question. The evaluation including two part. First, we are going to see if all the tasks can be finished before their deadlines. If it can be finished before it deadlines, we take the profit of that task in count. If it can’t, we don’t consider it profit. After summing up all the profit, we can compare the total profit with the integer k. The evaluation step takes polynomial time. So this decision question is in NP. The pseudocode is listed as follows.

Pseudocode:

Profit(A,t,k):

sort A by the deadlines #All the elements ai in A is sorted by their deadlines di.

#certificate

for each ai in A

non-deterministically to choose whether ai is scheduled or not

add scheduled tasks in sequence S

#verifier

t = 0

profit = 0

for each ai in S

if t + ti <= di

t = t + ti

profit = profit + pi

if profit < k

reject

accept

Second requirement: Reduce Subset Sum to this Job Scheduling problem. According to the lecture slides, we know Subset Sum is NP complete.

To prove we can reduce Subset Sum to this Job Scheduling problem, we need to show that (1) transformation from Subset Sum to Job Scheduling problem takes polynomial time; (2) The answers are the same for Subset Sum and Job Scheduling problem. That is, the answer for Subset Sum is “yes” if and only if the answer for Job Scheduling problem is also “yes”.

1. Transformation from from Subset Sum to Job Scheduling problem:

Subset sum: given a set A and target value k, is there a subset S of A that adds up to k. Considering the Job Scheduling problem with tasks whole set A (same as the subset sum set A), we assume the profit and the processing time of each element are the same as each element value. Each element has the deadline di – di-1 = ti and when i=1, d1 = t1. The final deadline for earning profit no less than k is also k.

1. If S is one instance of Subset Sum question, we know the summation of S will be k ( Σ(all ai in S) = k ). Thus, as for the Job Scheduling problem, Σ(all ai in S ) > = k and Σ(all ai in S) < = k are both satisfied, which means the profit of S no less than k and the whole processing time of S is no later than deadline k. So that S is also a truth assignment of Job Scheduling problem.

If X is one instance of Job Scheduling problem, we know that Σ(all ai in X) > = k and Σ(all ai in X) < = k are both satisfied, which means the profit of X no less than k and the whole processing time of X is no later than deadline k. To both satisfy the two formula, we know Σ(all ai in X) = k. This result can be one instance of subset Sum problem also cause the subset X summation is k.