

Homework Assignment 3

October 10, 2002

Due Date: October 24

Core Problems

1. (5 pts) An investor has decided to invest a total of \$50,000 among three investment opportunities: savings certificate, municipal bonds, and stocks. The annual return on each investment is estimated to be 7%, 9%, and 14%, respectively. The investor does not intend to invest her annual interest returns. She would like to maximize her yearly return (based on the estimates for the returns) while investing a minimum of \$10,000 in bonds. Also, the investment in stocks should not exceed the combined total investment in bonds and savings certificates. And finally, she should invest between \$5,000 and \$15,000 in savings certificates. Formulate a linear program to optimally solve this problem.
2. (10 pts) In SUBSET-SUM the input is a set $S = \{x_1, x_2, \dots, x_n\}$ and the integer t . In the COMPOSITE problem you are given as input an integer y and the question to answer is whether or not y has a factor besides 1 and itself. SUBSET-SUM is NP-complete and COMPOSITE is in NP. Clearly explain whether or not each of the following statements follow from these two facts.
 - (a) SUBSET-SUM \leq_p COMPOSITE.
 - (b) If there is an $O(n\sqrt{t})$ algorithm for SUBSET-SUM, then P = NP.
 - (c) If there is an $O(n^3 \log t)$ algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE.
 - (d) If there is an $O(\log y)$ algorithm for COMPOSITE, then P = NP.
 - (e) if P \neq NP, then no problem in NP can be solved in polynomial time.
3. (20 pts) In the scheduling with deadlines (SWD) problem, you are given n events where no two events can run at the same time. Event i is specified by three non-negative integers, an earliest start time s_i , a length ℓ_i and a latest allowed finishing time f_i . (Event i can be started at any time t that satisfies $s_i \leq t$ AND $t + \ell_i \leq f_i$.) The question is whether or not it is possible to legally schedule all n events. In the PARTITION problem, the input is a set $X = \{x_1, \dots, x_n\}$ of non-negative integers, and the question is whether or not there is a subset $S \subseteq X$ such that $\sum_{x \in S} x = \sum_{x \in X-S} x$. PARTITION is NP-complete.
 - (a) Professor P.T. Reduction believes SWD is NP-hard. To prove this he proposes the following transformation function T from a PARTITION input to a SWD input. Given $X = \{x_1, \dots, x_n\}$, first compute $B = \sum_{i=1}^n x_i$. The input for SWD is as follows. There will be $n + 1$ events. For event $1 \leq i \leq n$, let $s_i = 0$, $\ell_i = x_i$ and $f_i = B + 1$. Finally, he includes a $(n + 1)^{st}$ event which will serve as a divider to split the other events into two sets. For this event, set $s_{n+1} = 0$, $\ell_{n+1} = 1$ and $f_{n+1} = B + 1$. Does this reduction satisfy the requirements to show that SWD is NP-hard? If not, state exactly which required condition fails and prove that it fails.
 - (b) Either prove there is a polynomial time algorithm for SWD or prove SWD is NP-complete.

4. (10 pts) Given a set of m linear constraints over n variables, the **integer-programming problem** asks whether there is an *integer* n -vector x giving values for each of the n variables such that all the constraints are satisfied. Prove that integer programming is NP-complete by using a reduction from 3-CNF-SAT.

Hint: The arithmetic expression $1 - x$ performs a negation when x is 0 or 1.

5. (25 pts) For each of the following problems either give a polynomial-time algorithm to find an optimal solution *by formulating it as a linear program* or prove that the decision version of the problem (which you should clearly state) is NP-complete by using a reduction from one of: CIRCUIT-SAT, SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER.

- (a) The CS department needs to select a set of courses to offer during Summer 2002. All students are invited to submit a list of courses which they would like to take. There are s students who submit such a list where student i gives list L_i . The goal is to find a set of courses S to offer in Summer 2002 where $|S|$ is minimized such that S contains at least one element from L_i for $i = 1, \dots, s$.
- (b) A governmental planning agency for California wishes to determine the sources to purchase fuel for use by n depots from among m bidders. The maximum quantity offered by bidder i is a_i gallons at c_i dollars per gallon. The demand at depot j is b_j gallons. Let d_{ij} be the cost per gallon for the delivery from bidder i to depot j . The goal is to meet the demands at all j depots with the minimum possible total cost (fuel and shipping).

Advanced Problems (Required only for CS 539T students)

6. (10 pts) Prove that there is a polynomial time algorithm A that solves SAT (i.e. given a boolean formula ϕ , A respond “yes” if and only if ϕ has a satisfying assignment) if and only if there is some polynomial time algorithm B that when given a boolean formula ϕ it computes a satisfying assignment for ϕ (or reports none exists).
7. (15 pts) Consider the following problem. A gold digger is given a map (undirected graph) of a territory that consists of a set of towns (vertices) and trails (edges) between them. Each trail is labeled with a dollar value of the gold you would find along the trail the first time you travel it. Traveling a given trail a subsequent time yields no gold. Each town is labeled with the dollar cost of visiting it (hotels, meals, etc.), charged every time you enter the town. An “expedition” (cycle in the graph) is said to have profit k if k is the total amount of gold collected on the trails, less the total cost of visiting the towns along the way. The goal is to find an expedition that maximizes the profit. From this optimization problem we create the following decision problem that we will call the GOLD-DIGGER problem: Given an integer p and an undirected weighted graph G where each edge and vertex have a non-negative weight, is it possible to find an expedition with profit at least p .

Either prove that there is a polynomial time algorithm for the GOLD-DIGGER problem or prove that GOLD-DIGGER problem is NP-complete. If you try to prove it is NP-complete, you can use any of the following for your reduction: SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, PARTITION, HAM-CYCLE, TSP.