CS 570 - HW 5 Solution

Due Friday, April 24 (by 23:30)

1 Maximizing Profit (10 points)

A furniture company produces three types of couches. The first type uses 1 foot of framing wood and 3 feet of cabinet wood. The second type uses 2 feet of framing wood and 2 feet of cabinet wood. The third type uses 2 feet of framing wood and 1 foot of cabinet wood. The profit of the three types of couches is \$10, \$8, and \$5, respectively. The factory produces 500 couches each month of the first type, 300 of the second type, and 200 of the third type. However, this month there is a shortage of cabinet wood and the supply to the factory will have to be reduced by 600 feet, but the supply of framing wood is increased by 100 feet. How should the production of the three types of couches be adjusted to minimize the decrease in profit? Formulate this problem as a linear programming problem.

Solution:

Denote x_i as the number of change of couches of type i. The change in profits is

$$10x_1 + 8x_2 + 5x_3.$$

The change in the amount of cabinet wood used is

$$3x_1 + 2x_2 + x_3 \leq -600.$$

The change in the amount of framing wood used is

$$x_1 + 2x_2 + 2x_3 \leq 100.$$

The number of each type of couch produced should be non-negative, we have

$$x_1 \geq -500,\ x_2 \geq -300,\ x_3 \geq -200.$$

The goal is to minimize the decrease (or maximize the increase) in profit. We have the following linear programming:

$$
\begin{aligned}
\max \quad & 10x_1 + 8x_2 + 5x_3 \\
\text{subject to} \quad & 3x_1 + 2x_2 + x_3 \leq -600, \\
& x_1 + 2x_2 + 2x_3 \leq 100, \\
& x_1 \geq -500, \\
& x_2 \geq -300, \\
& x_3 \geq -200.
\end{aligned}
$$

Or equivalently,

$$
\begin{aligned}
\min \quad & -10x_1 - 8x_2 - 5x_3 \\
\text{subject to} \quad & 3x_1 + 2x_2 + x_3 \leq -600, \\
& x_1 + 2x_2 + 2x_3 \leq 100, \\
& x_1 \geq -500, \\
& x_2 \geq -300, \\
& x_3 \geq -200.
\end{aligned}
$$

2 Dual Program (15 points)

Consider the following linear program:

$$\max(3x_1 + 2x_2 + x_3)$$

subject to

$$\begin{aligned} x_1 - x_2 + x_3 &\leq 4 \\ 2x_1 + x_2 + 3x_3 &\leq 6 \\ -x_1 + 2x_3 &= 3 \\ x_1 + x_2 + x_3 &\leq 8 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Write the dual problem.

Solution:

Rewrite the equality constraint $-x_1 + 2x_3 = 3$ as follows:

$$\begin{aligned} -x_1 + 2x_3 &\leq 3 \\ x_1 - 2x_3 &\leq -3 \end{aligned}$$

- Multiply each equation by a new variable $y_k \geq 0$.

$$\begin{aligned} y_1(x_1 - x_2 + x_3) &\leq 4y_1 \\ y_2(2x_1 + x_2 + 3x_3) &\leq 6y_2 \\ y_3(x_1 + x_2 + x_3) &\leq 8y_3 \\ y_4(-x_1 + 2x_3) &\leq 3y_4 \\ y_5(x_1 - 2x_3) &\leq -3y_5 \end{aligned}$$

- Add up those equations.

$$\begin{aligned} &y_1(x_1 - x_2 + x_3) + y_2(2x_1 + x_2 + 3x_3) + y_3(x_1 + x_2 + x_3) + y_4(-x_1 + 2x_3) + y_5(x_1 - 2x_3) \\ \leq &4y_1 + 6y_2 + 8y_3 + 3y_4 - 3y_5 \end{aligned}$$

- Collect terms with respect to x_k.

$$\begin{aligned} &x_1(y_1 + 2y_2 + y_3 - y_4 + y_5) + x_2(-y_1 + y_2 + y_3) + x_3(y_1 + 3y_2 + y_3 + 2y_4 - 2y_5) \\ \leq &4y_1 + 6y_2 + 8y_3 + 3y_4 - 3y_5 \end{aligned}$$

- Choose y_k in a way that $A^T y \geq c$.
  Note that $b^T = (4, 6, 8, 3, -3)$ and $c^T = (3, 2, 1)$. The dual problem is:

$$\min(4y_1 + 6y_2 + 8y_3 + 3y_4 - 3y_5)$$

  subject to

$$\begin{aligned} y_1 + 2y_2 + y_3 - y_4 + y_5 &\geq 3 \\ -y_1 + y_2 + y_3 &\geq 2 \\ y_1 + 3y_2 + y_3 + 2y_4 - 2y_5 &\geq 1 \\ y_1, y_2, y_3, y_4, y_5 &\geq 0 \end{aligned}$$

  Or equivalently, letting $w = y_4 - y_5$ unconstrained.
  The dual problem is:

$$\min(4y_1 + 6y_2 + 8y_3 + 3w)$$

  subject to

$$\begin{aligned} y_1 + 2y_2 + y_3 - w &\geq 3 \\ -y_1 + y_2 + y_3 &\geq 2 \\ y_1 + 3y_2 + y_3 + 2w &\geq 1 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}$$

3 Spectrum Management (10 points)

Spectrum management is the process of regulating the use of radio frequencies to promote efficient use and gain a net social benefit. Given a set of broadcast emitting stations $s_1, \ldots, s_n$, a list of frequencies $f_1, \ldots, f_m$, where $m \geq n$, and the set of adjacent stations $E = \{(s_i, s_j)\}$. The goal is to assign a frequency to each station so that adjacent stations use different frequencies and the number of used frequencies is minimized. Formulate this problem as an integer linear programming problem.

Solution:

Note that n frequencies are enough to be assigned to any set of stations. We introduce n^2 binary variables x_{ij}, for $1 \leq i, j \leq n$, where x_{ij} denotes whether station s_i uses frequency f_j. In addition, we introduce one binary variable u_j for each frequency f_j, denoting whether frequency f_j is used or not. We have the following ILP:

$$\min_{x,u} \sum_{k=1}^{n} u_k$$

subject to

$$\begin{aligned}
& x_{ij} + x_{i'j} \leq 1, && \forall j \text{ and } \forall (i, i') \text{ s.t. } (s_i, s_j) \in E \\
& \sum_{j=1}^{n} x_{ij} = 1, && i \in [n] \\
& x_{ij} \leq u_j, && i, j \in [n] \\
& u_j, x_{ij} \in \{0, 1\}, && i, j \in [n]
\end{aligned}$$

The first inequality means that adjacent stations should have different frequency. The second inequality means that one frequency for each station. The third inequality means we use active frequencies only.

4 Short Questions (15 points)

True or false? Shortly explain your answers.

1. If $A \leq_p B$ and B is in NP-hard, then A is in NP-hard.
2. If $A \leq_p B$ and B is in NP, then A is in NP.
3. If $3-SAT \leq_p 2-SAT$, then $P = NP$.
4. Any NP problem can be solved in time $O(2^{poly(n)})$, where n is the input size and $poly(n)$ is a polynomial.
5. If a problem $A \leq_p B$, then it follows that $B \leq_p A$.

Solution:

1. False. One can transform the empty language $\varnothing$ to any NP-hard problem, but the empty language is not NP-hard.
2. True. One can construct a certifier for A by composing the certifier for B and the polynomial reduction map.
3. True. Since $2-SAT$ is in P and $3-SAT$ is in NP and the reduction shows that $2-SAT$ is in NP, we have $P = NP$.
4. True. Since one can use exponential time to try all possible certificates, each of them can be checked by a polynomial time deterministic Turing machine.
5. False. Let A be a problem in P and B be a problem that requires exponential time to solve. If $B \leq_p A$, then there exists a polynomial time algorithm for B, which leads to an contradiction.

5 Finding a Satisfying Assignment (10 points)

Assume that you are given a polynomial time algorithm that given a 3-SAT instance decides in polynomial time if it has a satisfying assignment. Describe a polynomial time algorithm that finds a satisfying assignment (if it exists) to a given 3-SAT instance.

Solution:

Given an instance of 3-SAT consisting of variables $X = \{x_1, \ldots, x_n\}$ and clauses $C_1, \ldots, C_k$. Let $\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_k$ be the corresponding Boolean formula. Define ϕ_i and $\overline{\phi_i}$ as the CNF formula obtained from ϕ by setting x_i to TRUE and FALSE, respectively. Let $A(\phi)$ be the given algorithm that decides whether a given instance has a satisfying assignment. Suppose $A(\phi)$ returns TRUE if ϕ is satisfiable and FALSE otherwise. Note that ϕ_i is satisfiable if and only if ϕ has a satisfying assignment where x_i = TRUE. Then the following algorithm constructs satisfying assignment for ϕ if possible, or returns that no such assignment exists.

- If $A(\phi)$ = FALSE, then return that ϕ not satisfiable.
- For $i = 1, \ldots, n$
  - If $A(\phi_i)$ = TRUE
    - $\phi \leftarrow \phi_i$
    - $x_i \leftarrow$ TRUE
  - Else
    - $\phi \leftarrow \overline{\phi_i}$
    - $x_i \leftarrow$ FALSE
- Return $x = (x_1, \ldots, x_n)$

6 Shipping Goods (15 points)

Given n positive integers $x_1, \ldots, x_n$. The Partition Problem asks if there is a subset $S \subseteq [n]$ such that:

$$\sum_{i \in S} x_i = \sum_{i \notin S} x_i.$$

It can be proven that the Partition Problem is NP-complete. You do not to prove it, but rather use it in the following problem.
Assume that you are consulting for a shipping company that ships a large amount of packages overseas. The company wants to pack n products with integer weights $p_1, \ldots, p_n$ into a few boxes as possible to save on shipping costs. However, they cannot put any number of products into a box due to the shipping capacity restriction. The total weight of products in each box should not exceed W? You may assume that for each i-th product $p_i \leq W$. You task is to advise the company if n products can be placed into at most $k < n$ boxes. Show that the problem is NP-Complete by reduction from the Partition Problem.

Solution:

1. Claim: this problem is in **NP**.

   We can be verify whether n products are placed into at most k boxes such that the total weight of products in each box does not exceed W in linear time.

2. Claim: There is a subset S such that $\sum_{i \in S} x_i = \sum_{i \notin S} x_i$ if and only if n products with weight $p_i = x_i$ for all i can be placed into at most $k = 2$ boxes, where the weight of each box is no more than $W = \frac{1}{2} \sum_i p_i$.

   Let $k = 2$ and $W = \frac{1}{2} \sum_i p_i$, where $p_i = x_i$. If there is a solution to the Partition problem, then the sum of elements in each of S and $[n] \setminus S$ is $W = \frac{1}{2} \sum_i p_i$. By transforming these two subsets into two boxes, we have a solution to this problem. Conversely, if there is a solution to this problem with 2 boxes and $W = \frac{1}{2} \sum_i p_i$, then all products are in those 2 boxes and the total weight of them are the same. By transforming these two boxes into two subsets, we have a solution to the Partition problem.

Therefore, we can conclude that this problem is NP-complete.

7 Longest Path (15 points)

Given a graph $G = (V, E)$ and a positive integer k, the longest-path problem is the problem of determining whether a simple path (no repeated vertices) of length k exists in a graph. Show that this problem is NP-complete by reduction from the Hamiltonian path.

Solution:

1. Claim: this problem is in **NP**.

   A simple path of length k can be verified by traversing the path in linear time, including checking whether there is no repeated vertices, the length is k, and all edges of the path are in E.

2. Claim: G has a Hamiltonian path if and only if G has a simple path of length $k = n - 1$.

   Given an instance of the Hamiltonian path problem: a graph with n vertices. Since a Hamiltonian path in G is equivalent to a simple path of length $n - 1$ in G, it turns out that there is a polynomial time reductions from the Hamiltonian path problem to this problem. Specifically, if there is a Hamiltonian path, then that path is a simple path of length $n - 1$. On the other hand, if there is a simple path of length $n - 1$, then it is a Hamiltonian path.

Therefore, we can conclude that this problem is NP-complete.

8 Helping with the COVID-19 Crisis (10 points)

Given an integer k, a set C of n cities $c_1, \ldots, c_n$, and the distances between these cities $d_{ij} = d(c_i, c_j)$, for $1 \leq i < j \leq n$, where d is the standard Euclidean distance. We want to select a set H of k cities for building mobile hospitals in light of the coronavirus outbreak. The distance between a given city c and the set H is given by $d(c, H) = \min_{h \in H} d(c, h)$. The goal is to select a set H of k cities that minimizes $r = \max_{c \in C} d(c, H)$. Namely, the maximum distance from a city to the nearest mobile hospital is minimized. Give a 2-approximation algorithm for this problem.

Solution:

Consider the following algorithm:

1. Assume $k \leq |C|$ (Otherwise define $H = C$)
2. Select any city $c \in C$ and let $H = \{c\}$
3. While $|H| \leq k$
   - Select a city $c \in C$ that maximizes $d(c, H)$
   - Add city c to H
4. Return H as the selected set of cities for building mobile hospitals

Claim: Let H^* be an optimal set of k points. Then this algorithm returns a set H of k points such that $\max_{c \in C} d(c, H) \leq 2 \max_{c \in C} d(c, H^*)$.
Let $r = \max_{c \in C} d(c, H)$ and $r^* = \max_{c \in C} d(c, H^*)$. We want to prove this is an 2-approximation by contradiction. Assume $r > 2r^*$.
Since $r^* < \frac{1}{2}r$, for each $h_i \in H$, there is exactly one $h_i^* \in H^*$ such that $d(h_i, h_i^*) < \frac{1}{2}r$. Consider any city $c \in C$ and its closest hospital $h_j^* \in H^*$. Then

$$d(c, H) \leq d(c, h_j) \leq d(c, h_j^*) + d(h_j^*, h_j) \leq 2r^*.$$

This contradicts the assumption that $r > 2r^*$. Hence $r \leq 2r^*$.