CS 570 - HW 5

Due April 16, 2021 (by 4 AM Pacific Time)

1 Maximizing Profit (15 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 (with no wood resources leftover). However, this month there is a shortage of cabinet wood 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 + Rubric:
Explanation (3 points): Suppose the number of couches of each type that are produced *in addition* to the usual be x_1, x_2, x_3. The net additional profit is $10x_1 + 8x_2 + 5x_3$ which should be maximized (equivalent to minimizing decrease in profit). This objective is subject to the following constraints. The additional consumption of framing wood is $x_1 + 2x_2 + 2x_3$ which can be at most 100, while the additional consumption of cabinet wood is $3x_1 + 2x_2 + x_3$ which can be at most -600 (owing to a shortage of 600). The total number of couches of each type is $500 + x_1, 300 + x_2, 200 + x_3$ resp., all of which should be non-negative. The resultant LP formulation is as follows:

maximize:	$10x_1 + 8x_2 + 5x_3$	3 points
subject to:	$3x_1 + 2x_2 + x_3 \leq -600,$	3 points
	$x_1 + 2x_2 + 2x_3 \leq 100,$	3 points
	$x_1 \geq -500,$	1 point
	$x_2 \geq -300,$	1 point
	$x_3 \geq -200.$	1 point

Alternatively, define x_1, x_2, x_3 as the decrements in couch quantities produced to get the following LP:

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

Alternatively, define x_1, x_2, x_3 as the new couch quantities produced. This requires computing the default wood supplies as 2300 and 1500 units, and in turn, the new wood supplies as 1700 and 1600 units. Then, we get the following LP:

$$
\begin{array}{ll}
\text{maximize:} & 10x_1 + 8x_2 + 5x_3 \\
\text{subject to:} & 3x_1 + 2x_2 + x_3 \leq 1700, \\
& x_1 + 2x_2 + 2x_3 \leq 1600, \\
& x_1 \geq 0, \\
& x_2 \geq 0, \\
& x_3 \geq 0.
\end{array}
$$

More details on the rubric:

- The scoring break-up is the same in all 3 cases: 3 for explaining the variables and constraints, 3 for objective, 3+3+1+1+1 for constraints as shown.
- In each of the cases, the objective $\max f(x)$ can be switched to $\min -f(x)$ and vice versa.
- Any objective expression $f(x)$ can also be expressed as $f(x) + c$ for some constant c (e.g., the usual month's profit).
- 2 point penalty if max/min is incorrect
- 2 points penalty each if only the inequality signs are incorrect in the first 2 constraints (wood supply constraints)

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. You do not need to demonstrate intermediate steps.

Solution + Rubric:

Let dual variables y_1, y_2, y_3 correspond to the three $\leq$ constraints and let y_4 correspond to the $=$ constraint. Then, we get the dual LP as follows:

$$\begin{array}{ll} \text{min:} & 4y_1 + 6y_2 + 8y_3 + 3y_4 \\ \text{subject to:} & y_1 + 2y_2 + y_3 - y_4 \geq 3, \\ & -y_1 + y_2 + y_3 \geq 2, \\ & y_1 + 3y_2 + y_3 + 2y_4 \geq 1, \\ & y_1, y_2, y_3 \geq 0. \end{array}$$

Scoring breakup (3+3+5+4):

- 3 Points for having four dual variables.
- 3 points for having three constraints (excluding variable bounds).
- 5 points for all coefficients and inequality signs (2 point penalty for incorrect inequality signs, 2 point penalty for incorrect objective, neither to be repeated if the mistake is due to incorrect number of variables/constraints which is already penalized as aforementioned).
- 4 points for variable bounds (In particular, y_4 must be unrestricted).

3 Spectrum Management (15 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 $\{(s_i, s_j)\}$ for some $i, j \in [n]$. 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 + Rubric:
Define variables (2 points): Let binary x_{ij} denote whether station s_i uses frequency f_j or not. Let binary u_j denote whether frequency f_j is used or not.

Let E denote the set of pairs of adjecent stations. (Given as input)
Here is the IP formulation:

$min_{x,u}$:	$\sum_{k=1}^{n} u_k$	3 points
subject to:		
	$x_{ij} + x_{kj} \leq 1, \ \forall j \in [m] \ \forall (s_i, s_k) \in E$	3 points
	$\sum_{j=1}^{n} x_{ij} = 1 \ \ \forall i \in [n],$	3 points
	$x_{ij} \leq u_j \ \ \forall i \in [n] \ \forall j \in [m],$	3 points
	$u_j \in \{0,1\} \ \forall i \in [n],$ $x_{ij} \in \{0,1\} \ \forall i \in [n] \ \forall j \in [m].$	1 point

Explanation: The objective minimizes the number of frequencies used. Constraint 1 ensures any adjecent stations s_i, s_k do not have the same frequency f_j for any j. Constraint 2 ensures each station is assigned exactly one frequency. Constraint 3 ensures that any frequency j is counted as used ($u_j = 1$), if it is assigned to any station ($x_{ij} = 1$). Finally constraints 4,5 ensure that the variables take binary values.
Side Note: This is identical to the well-known Garph Coloring problem.
Rubric details:

- Full points for individual objective/constraints only if the explanation is provided. 1 point penalty for each missing/incorrect explanations (max penalty 3 if no explanations provided).
- 1 point penalty for each missing/incorrect $\forall$ descriptors (max penalty 2 if none are provided).
- A maximum of 8 points for a partially incorrect approach, such as not defining any u_j, and attempting to capture the objective with just x_{ij}'s

4 Short Questions (15 points)

Prove or disprove the following statements.

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 + Rubric:

1. False (2 points). This only suggests B is "at least as hard" as A. A need not be NP-hard.
2. True (5 points). Since $A \leq_p B$, we can construct an instance of B of size m where m is at most $O(n^c)$ for some integer c, with n being the size of the A instance. Further, since B is in NP, there exists a certificate of size $O(m^d)$ for some integer d, and a certifier that runs in time $O(m^f)$ for some integer f. To show that $A \in NP$, we provide a certificate for the B instance that is obtained from the reduction of the A instance. The size of this certificate is $O(m^d) = O(n^{cd})$. The certifier would first obtain the B instance in polynomial time (since $A \leq_p B$) and then run the certifier of B which runs in time $O(m^f) = O(n^{cf})$, thus, also polynomial in n. Thus, A has a certificate of polynomial size and a certifier running in polynomial time, hence, $A \in NP$.
   Scoring: 2 points for answer, 3 for explanation — 1.5 each for showing poly-size certificate and poly-time certifier.
3. True(2 points). 2-SAT is in P. Hence, 3-SAT $\leq_p$ 2-SAT would mean that 3-SAT is in P. 3-SAT being NP-hard, every NP problem can be reduced to 3-SAT, and in turn, can be solved in polynomial-time. Thus, P=NP.
4. True(4 points). Since each certificate is of polynomial size, say, $O(n^c)$, there are $O(2^{n^c})$ different possible certificates. We can check for each of them in polynomial time, and output yes if one is found. Thus, in total $O(2^{poly(n)})$ time suffices.
   Scoring: 2 points for answer, 2 for explanation.
5. False(2 points). $\leq_p$ is not a symmetric relation. As a counter-example, consider a case that A is a problem in P, while B is not.

No penalty if the explanation is omitted for 1,3,5.

5 Finding a Satisfying Assignment (20 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 + Rubric:
Let $\Phi(x_1, x_2, \ldots, x_n) = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ be the boolean formula corresponding to a 3-SAT instance where $c_1, c_2, \ldots, c_m$ are the clauses and $x_1, x_2, \ldots, x_n$ are the variables. Suppose the algorithm A takes a CNF formula and outputs 1 if it is satisifiable, and 0 if not.
If $A(\Phi((x_1, x_2, \ldots, x_n))) = 0$, then return that the instance is non satisfiable. If $A(\Phi((x_1, x_2, \ldots, x_n))) = 1$, then, i.e., given Φ is satisfiable, we can find an assignment for x_1 as follows. If $A(\Phi((1, x_2, \ldots, x_n))) = 1$, then we can set $x_1 = 1$ and be guaranteed that $\Phi_1(x_2, x_3, \ldots, x_n) := \Phi(1, x_2, \ldots, x_n)$ is satisfiable. Else, we can set $x_1 = 0$ and be guaranteed that $\Phi_1'(x_2, x_3, \ldots, x_n) := \Phi(0, x_2, \ldots, x_n)$ is satisfiable. We can continue iteratively with Φ_1 or Φ_1' resp. to find the subsequent assignments. (E.g., say Φ_1 is satisfiable. Then, in the second iteration, we find an assignment for x_2 by computing $A(\Phi_1(1, x_3, \ldots, x_n))$, and setting $x_2 = 1$ if it returns 1, or $x_2 = 0$ otherwise, and so on.)
This will lead to n iterations with one call to A per iteration. In each iteration, finding the new formula by plugging the value of a variable takes $O(m)$ time. Hence, we can find the assignment in polynomial time.
Rubric details:

- Any iterative/recursive algorithm that runs with linearly many calls to A is acceptable. Describing in words or a pseudo-code are both okay.
- 2 point penalty if polynomial runtime is not explained.
- 3 point penalty if it calls A more than linearly many times but still polynomial.
- At most 5 points allotted for an algorithm with exponentially many calls to A.

6 Multi-Lane Highway (20 points)

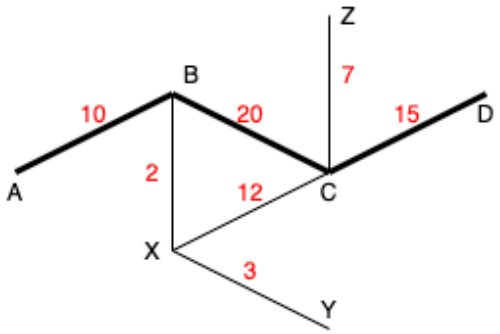
The government wants to build a multi-lane highway across the country. The plan is to choose a route and rebuild the roads along this route. We model this problem with a simple weighted undirected graph with the nodes denoting the cities and the edges capturing the existing road network. The weight of an edge denotes the length of the road connecting the corresponding two cities.

Let d_{uv} denote the shortest path distance between nodes u and v.

Let $d(v, P)$ denote the shortest path distance from a node v to the *closest* node on a path P (i.e. $\min_{u \in P} d_{uv}$).

Next, we define the aggregate remoteness of P as $r(P) = \sum_{v \in V} d(v, P)$.

In the example shown in the figure below, path $P = ABCD$ is the highway. Then, $d(A, P) = d(B, P) = d(C, P) = d(D, P) = 0$, while $d(X, P) = d_{XB} = 2$, $d(Y, P) = d_{YB} = 5$, and, $d(Z, P) = d_{ZC} = 7$. The remoteness of path ABCD is $0 + 0 + 0 + 0 + 2 + 5 + 7 = 14$.



The government wants a highway with the minimum aggregate remoteness, so that all the cities are somewhat close to the highway. Formally, we state the problem as follows, "Given a graph G, and a number k, does there exist a path P in G having remoteness $r(P)$ at most k"? Show that this problem is NP-complete by reduction from the Hamiltonian Path problem.

Solution + Rubric:

Call this problem REMOTE-PATH.

REMOTE-PATH $\in$ NP (6 points): A path will be a certificate (thus, poly-size). For each vertex v, $d(v, P)$ can be computed by first running Dijkstra from v and then finding the closest point on P. Adding $d(v, P)$ for all the nodes v gives the total remoteness. Thus, the certifier runs in polynomial time.

NP-hardness (14 points): We show this with a reduction from Hamiltonian Path (HP). Consider an HP instance graph $G = (V, E)$. Construct a weighted graph H by assigning arbitrary positive weights to all the edges in E. Pass this weighted graph H and $k = 0$ as input to REMOTE-PATH. We show that G has a hamiltonian path if and only if H has a path with zero remoteness. To

prove the first direction, suppose P is a hamiltonian path of G. Since, all the nodes lie on P, we have $d(v, P) = 0\ \forall v \in V$, and hence, $r(P) = 0$. Hence, REMOTE-PATH outputs 'Yes'. For the other direction, suppose H has a path with zero remoteness, say P. If there is a node $v \notin P$, then $d(v, P) > 0$ since all the edge-weights are positive. Hence, as $r(P) = 0$, P must contain all the nodes, thus, G indeed has a hamiltonian path.
Since assigning the weights is done in $O(|E|)$ time, the reduction is poly-time.

Rubric details:

- Showing in NP: 2 points for the certificate, 3 points for the certifier (need to explain how the remoteness can be computed). 1 point for mentioning polynomial time.
- Reduction for NP-hardness: 5 for construction (3 for Adding **strictly positive** weights to unweighted HP instance, 2 for setting $k = 0$). 4+4 for proving each direction (partial scoring for unclear/missing proof steps). 1 for mentioning that the reduction is poly-time.
- Maximum 5 points for a reduction TO Hamiltonian Path. (i.e. taking the city graph and checking if it has a hamiltonian path)