CMSC 37000: Homework 3

Caleb Derrickson

February 26, 2024

Contents

Problem 1

You work for a cryptocurrency startup. the startup has a supercomputer that can mine two types of coins: CatCoins and DogCoins. The profit from mining these coins varies over time. the profit for mining CatCoins on day i is $c_i > 0$ and the profit for mining DogCoins on day i is $d_i > 0$. There are k days and your goal is to determine what coin to mine on day i for every $i \in \{1, ..., k\}$. However, the computer cannot immediately switch from mining CatCoins to mining DogCoins or vice versa, because each type of coins requires its own software, and it takes 1 day to load it. So, for example, if the computer mines CatCoins on day i, it can start mining DogCoins only on day $i + 2$ and there will be no profit on day $i + 1$.

Design a DP algorithm that given numbers $c_1, ..., c_k$ and $d_1, ..., d_k$ finds the maximum profit the startup can get. Please do the following:

1. Define subproblems.
2. Define the dynamic-programming table and explain the meaning of its entries.
3. Write the initialization step of your algorithm.
4. Write the recurrence formula for computing the entries of the table.
5. Explain why the recurrence formula is correct.
6. Find the running time of your algorithm.

Solution:

Since we need to take into consideration the cool-down on switching between coins, we should compare the values of the next two days for both coins. As such, we define the DP table T to have entries depicting the value of the next two chosen days. It would also be convenient to store an integer value in the DP table, so our table would be of size $2 \times n/2$, where the first row stores the max value of the next two days with respect to the recurrence, and the second row stores our choice; whether to stay with the current coin we are mining, or switch to the other. To this extent, we will define the recurrence of the FIRST row of the DP table as

$$T[0][i+1] = \max\{current[i+1] + current[i+2], other[i+2]\},$$

and the SECOND row of the DP table to store the index of the choice taken above. In other words: if the maximum of the two values were the next two values in the current array, we take the set $T[1][i+1] = 0$, else we set it to 1. We then have another 2d array which stores the two arrays of coin values, and update the current array to reflect the value of $T[1][i+1]$. This will ensure that we are mining the for the most amount of money in any given sprint. There are two things left to consider:

1. The initialization of our DP array.
2. What if, when choosing our next value in the DP array, the values we are choosing are the same?

3. What if k, the number of days we are considering, is odd?

The first is relatively simple: since our recurrence formula does not depend on the current iterative value (we are not accessing the i - th value in T), we can set T[0][0] = 0 and T[1][0] = -1, to represent a choice among zero days. We can simply remove this column in our DP table and shift everything over to the left by one if we so desire, but it simply doesn't matter. The next item to discuss is when in our DP table would there be any ambiguity in whether we shift to the other array, or remain with our current array. This will exactly happen when, in our maxing function, that the two values are equivalent. In this case, we would need to look at the *next two* days in the array, if there are such two days. If we see that in the next two days that the value of current is the maximum value, we then stick with the current, else we move to the other array. Finally, if the number of days in the array is odd, then when we get to the last choice in our DP table, we would only have one day to look at, and $i + 2$ would represent an out of bounds index value. To this extent, since we would only be losing mining time if we switched now, we would be better of sticking with our current array. Therefore, with all of this being considered, we would be maximizing our profit with the described algorithm. In discussing the running time of our algorithm, since we are only looping over both arrays once, and for each iteration of our algorithm, we are considering the next two values in our index, we can conclude we operate in $O(n/2) = O(\log_2 n)$ time, with also a memory cost of $O(\log_2 n)$.

Problem 2

Convert the following LP to the canonical form. Then write the dual LP

$$\text{maximize: } x - w$$

subject to

$$x + y + z \leq 5$$

$$x - z + w = 3$$

$$x \geq 0$$

$$y \geq 0$$

$$z \geq -2$$

$$w \geq 0$$

Is this LP feasible? If it is feasible, is it bounded or unbounded? Is the dual feasible? If it is feasible, is it bounded or unbounded?

Solution:

This problem has to it multiple subproblems. I will enumerate them via the following:

1. Converting the LP to canonical form.
2. Writing the corresponding Dual LP.
3. Assessing the Feasibility and boundedness of the primal.
4. Assessing the Feasibility and boundedness of the dual.

---

1. Canonical form of the LP:

   In order to convert this Linear program into its canonical form, we seek to convert it to the form

   $$\text{maximize:} c^\intercal x$$

   subject to

   $$Ax \leq b$$

   $$x \geq 0$$

   We see that, the form of the given constraints is almost in this form - with the exception of the z variable it is correct. To assuage this, we introduce two new variables z_+ and z_- such that $z = z_+ - z_-$. Note

that, when $z = -2$, then $z_+ = 0$ and $z_- = 2$. Therefore, the minimum value with which z can feasibly achieve is faithfully represented by two positive variables z_+, z_-. We also see for strictly positive values of z, $z_- = 0$. Therefore, the LP now becomes

$$\text{maximize: } x - w$$

$$\text{subject to}$$

$$x + y + z_+ - z_- \leq 5$$

$$x - z_+ + z_- + w = 3$$

$$x \geq 0$$

$$y \geq 0$$

$$z_+ \geq 0$$

$$z_- \geq 0$$

$$w \geq 0$$

The system of constraints can now be written as $Ax \leq b$, $x \geq 0$, where

$$A = \begin{bmatrix} 1 & 1 & 1 & -1 & 0 \\ 1 & 0 & -1 & 1 & 1 \end{bmatrix}, \quad x = \begin{pmatrix} x \\ y \\ z_+ \\ z_- \\ w \end{pmatrix}, \quad b = \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \quad c = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}.$$

2. Writing the Dual LP:

In writing the dual of an LP, it is important to keep in mind the following: [1]

- Each primal constraint has a dual constraint.
- Each primal variable becomes a dual constraint.
- The coefficient of a dual variable in the dual constraint is the coefficient of its primal variable in its primal constraint.

To this end, we note that we have 2 constraints in our primal LP, which will give way to two variable in the dual, which I'll call y_1 and y_2, mind the overlapping notation. The second tells us that our dual problem will as many constraints as there are variables in the original problem. Therefore, we will have 5 constraints imposed on our dual, some of which might become redundant upon further inspection. Finally, in order to write our system of constraints, we need to take in mind that our coefficients for our dual variables will be the coefficient of its corresponding primal constraint. We then have the following

[1] This comes from the Wikipedia page for Dual Linear programs.

system of constraints:

$$\begin{aligned} y_1 + y_2 &\geq 1 \\ y_2 &\geq 0 \\ -y_1 + y_2 &\geq 0 \\ y_1 - y_2 &\geq 0 \\ y_1 &\geq -1 \end{aligned}$$

Therefore, we can write our dual LP as the following:

$$\begin{aligned} &\text{minimize } b^\intercal y \\ &\text{subject to} \\ &\qquad A^\intercal y \geq c \\ &\qquad y \geq 0 \end{aligned}$$

Where,

$$A^\intercal = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix},\ b = \begin{pmatrix} 5 \\ 3 \end{pmatrix},\ c = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

Note the problem just asks for us to write just the Dual LP, not requiring us to write it in canonical form.

3. Feasibility and boundedness of the Primal LP:

   Boundedness of an LP says that the LP itself has an optimal solution. What would be a better way to show feasibility than to find the optimal solution? I used an online Linear program solver[2] to solve my linear program. Here is my input to the solver:

```
var x1 >= 0;
var x2 >= 0;
var x3 >= 0;
var x4 >= 0;
var x5 >= 0;

maximize z:     x1 - x5;
subject to c11: x1 + x2 + x3 - x4 <= 5;
subject to c12: x1 - x3 + x4 + x5 = 3;

end;
```

[2]Hosted at https://online-optimizer.appspot.com.

The optimal value the solver found was $(x, y, z_+, z_-, w) = (4, 0, 1, 0, 0)$, giving a maximum value of 4. Therefore, the problem is bounded, which notably also implies feasibility.

4. Feasibility and boundedness of the Dual LP:

   Before we throw or problem into an LP solver, it is worthwhile to note some of the constraints of the Dual LP are extraneous. The second constraint is a given, since we require y_1 and $y_2 \geq 0$. From the third and the fourth constraint, we see that $y_1 - y_2$ is both greater than and less than zero, implying $y_1 - y_2 = 0$. In order to reasonably apply the primal LP to the online linear solver, we should write it in canonical form, which would just introduce two new variables y_2^+ and y_2^-, where we substitute $y_2^+ - y_2^-$ for y_2. The following was fed into the online solver:

```
var y1 >= 0;
var y2 >= 0;
var y3 >= 0;

minimize z:     5*y1 + 3*y2 - 3*y3;
subject to c11: y1 + y2 - y3 >= 1;
subject to c12: y1 - y2 + y3 = 0;

end;
```

   We see that the optimal value is achieved for $(y_1, y_2^+, y_2^-) = (0.5, 0.5, 0)$, giving a minimal value of 4, which is the same as the primal maximal value. Therefore, the primal LP is shown to be both feasible and bounded.

Problem 3

In this problem, we will design an algorithm for solving the Minimum Vertex Cover problem in bipartite graphs. Let $G = (X, Y, E)$ be a bipartite graph with parts X and Y. Consider the following linear programming formulation of the Minimum Vertex Cover Problem. There is an LP variable x_u for every vertex $u \in X \cup Y$.

$$\text{minimize: } \sum_{u \in X \cup Y} x_u$$

subject to

$$x_u + x_v \geq 1 \text{ for every edge } (u, v) \in E$$

$$x_u \geq 0$$

Problem 3, part 1

Prove that the value of this program is at most the minimum vertex cover size.

Solution:

A simple way (maybe the only way) to show this is to show that the optimal value (the minimum vertex cover, OPT) is a feasible solution to the LP. Let us suppose that the minimum vertex cover can be represented via a binary array, $\hat{x}$, which indicates whether or not vertex i is within the minimum vertex cover. Clearly, for each u, $\hat{x}_u \geq 0$ since each value in $\hat{x}$ is either zero or one. All that is left to show is that for any $(u, v) \in E, \hat{x}_u + \hat{x}_v \geq 1$.

Suppose false, that is, there exists an edge $(u, v) \in E$ such that $\hat{x}_u + \hat{x}_v < 1$. This would imply that both $\hat{x}_u$ and $\hat{x}_v$ are equal to zero, since $\hat{x}$ is a binary array. Since G is a bipartite graph, then without loss of generality assume $u \in X$, $v \in Y$, since there is an edge between them. Since the cover is minimum vertex cover, then we require at least one external vertex, call it e, to connect to u. And since G is bipartite, e cannot connect to v, since it is already connected to u. Therefore, we need an additional vertex, call it d to cover v. Since we require two external vertices to cover these two, and it would be more efficient to just connect the two vertices (u, v), then $\hat{x}$ is not a minimum vertex cover, which leads to a contradiction. Therefore, $\hat{x}_u + \hat{x}_v \geq 1$ for every edge $(u, v) \in E$.

Since $\hat{x}$ is a feasible solution for the LP, then the value of the minimum cover, OPT, will be at least the value of the output of the LP, since the LP finds the minimum of its feasible solutions. Therefore, LP $\leq$ OPT.

Problem 3, part 2

Let $\hat{x}$ be an optimal solution. Prove that $\hat{x}_u \in [0, 1]$ for every $u \in X \cup Y$.

Solution:

Suppose false. That is, there exists a vertex $u \in X \cup Y$ for which its value $\hat{x}_u > 1$.[3] By the first constraint condition, we have that, for every neighbor v of u, $\hat{x}_u + \hat{x}_v \geq 1$, but $\hat{x}_u > 1$, so $\hat{x}_u + \hat{x}_v > 1 + \hat{x}_v$. Since the output is optimal, and setting the values of all neighbors of u to zero is feasible, then $\hat{x}_v = 0$ for all neighbors of v of u. Denote the set of neighbors of u as $N(u)$. Without loss of generality we assume the graph is connected. In this case, we have that $|N(N(u))| \geq |N(u)|$, since for each vertex $v \in N(u)$, it has to have at least one neighbor. Since all vertices $v \in N(u)$ have their value $\hat{x}_v = 0$, then by the first constraint, every vertex $w \in N(N(u))$, the neighbors of the neighbors of u, need to have their values $\hat{x}_u \geq 1$. This would then imply

$$\sum_{u \in X \cup Y} \hat{x}_u = \sum_{\text{All other } u} \hat{x}_u + \sum_{w \in N(N(u))} \hat{x}_w + \hat{x}_u$$

But since $|N(N(u))| \geq |N(u)|$, it would be more efficient to have that some values $\hat{x}_v$ for $v \in N(u)$ to be nonzero. Therefore, $\hat{x}$ is note an optimal solution, giving a contradiction.

[3] It is obvious by positivity of vertex values why we neglect the case when $\hat{x}_u < 0$.

Problem 3, part 3

Assume additionally that $\hat{x}$ is a vertex solution. Prove that then $\hat{x}_u \in \{0, 1\}$. To this end, consider the set of graph vertices A:

$$A = \{u \in X \cup Y : \hat{x}_u \neq 0 \text{ and } \hat{x}_u \neq 1\}.$$

We need to prove that A is empty. Assume to the contrary that A is not empty. Denote $\varepsilon = \min\{\hat{x}_u, 1 - \hat{x}_u : u \in A\}$. Consider solutions x' and x'' defined by

$$x'_u = \begin{cases} \hat{x}_u & \text{if } u \notin A \\ \hat{x}_u + \varepsilon & \text{if } u \in A \cap X \\ \hat{x}_u - \varepsilon & \text{if } u \in A \cap Y \end{cases}, \quad x''_u = \begin{cases} \hat{x}_u & \text{if } u \notin A \\ \hat{x}_u - \varepsilon & \text{if } u \in A \cap X \\ \hat{x}_u + \varepsilon & \text{if } u \in A \cap Y \end{cases}$$

Prove that x' and x'' are feasible solutions. Conclude that $\hat{x}$ is not a vertex solution.

Solution:

The bulk of the analysis was given to us in the problem, so we just need to show that x' and x'' are feasible solutions. To show this, we just need to show that x' and x'' satisfy the equality constraints. This will be shown case-by-case:

1. x' is a feasible solution:

   (a) $x'_u \geq 0$ for all $u \in X \cup Y$:

   Since we have two options for ε, it would be the most straightforward to break it into two cases.

   i. Case 1: $\varepsilon = \hat{x}_v$ for some $v \in A$.

   By the construction of x', for all $u \notin A$, $x'_u = \hat{x}_u$, which since $\hat{x}$ is an optimal solution, $\hat{x}_u \geq 0$ for $u \notin A$. Next we consider $u \in X \cap A$. Since $\varepsilon \geq 0$, (since $\hat{x}$ is a feasible solution), this implies $\hat{x}_u + \hat{x}_v \geq 0$ for all $u \in X \cap A$, $\implies \hat{x}_u + \varepsilon \geq 0$. We next consider $u \in A \cap Y$. Since $\hat{x}_v$ is chosen to be the minimal value for all $u \in A$, this implies that $\hat{x}_u \geq \hat{x}_v$ for all $u \in A \cap Y$. Then, $\hat{x}_u - \hat{x}_v \geq 0$, $\implies \hat{x}_u - \varepsilon \geq 0$.

   ii. Case 2: $\varepsilon = 1 - \hat{x}_v$ for some $v \in A$.

   Again, by construction of x' we need to only consider $u \in A$. Since $\hat{x}$ is a feasible solution, then we have that $\hat{x}_v \in [0, 1]$, this implies that $1 - \hat{x}_v \in [0, 1]$, i.e. $\varepsilon = 1 - \hat{x}_v \geq 0$. Since ε is chosen as $1 - \hat{x}_v$, then this value is less than (or equal to) $\hat{x}_u$ for all $u \in A$ (if it wasn't, then it wouldn't have been chosen). This implies that $1 - \hat{x}_v \leq \hat{x}_u$ for all $u \in A$, which implies that $\varepsilon \leq \hat{x}_u$, so $\hat{x}_u - \varepsilon \geq 0$.

   Therefore, in either case, $x'_u \geq 0$ for all $u \in X \cup Y$.

(b) $x'_u + x'_v \geq 1$ for all $(u, v) \in E$:

We then need to consider all cases for $(u, v) \in E$. Note that if $u, v \notin A$, then $x'_u = \hat{x}_u$ and $x'_v = \hat{x}_v$, and since $\hat{x}$ is a feasible solution, $\hat{x}_u + \hat{x}_v \geq 1$. Thus we consider the two cases when either one is in A, or both are in A.

i. $u \in A,\ v \notin A$:

Note by the symmetry of the construction of x', it doesn't matter which vertex is in A, so letting $u \in A$ is sufficient for showing this case. Then, $x'_u + x'_v$ is equal to either $\hat{x}_u + \hat{x}_v + \varepsilon$ or $\hat{x}_u + \hat{x}_v - \varepsilon$. Note that since $\varepsilon \geq 0$, then the former cases implies that $x'_u + x'_v \geq 1$. We then need to show the latter. Since we assume $v \notin A$, then $\hat{x}_v \in \{0, 1\}$. If $\hat{x}_v = 0$, then this would imply that $\hat{x}_u \geq 1$, but since $\hat{x} \in [0, 1]$ for all elements, this would imply that $\hat{x}_u = 1$. But we assume that $\hat{x}_u \in A$, which is a contradiction. So we then consider the case when$\hat{x}_v = 1$. This then gives $\hat{x}_u + 1 - \varepsilon$, but by the following:

$$\begin{aligned}
&\hat{x}_u + 1 - \varepsilon = \hat{x}_u + 1 - \min\{\hat{x}_a,\ 1 - \hat{x}_a : a \in A\} \\
&\geq \{\hat{x}_u + 1 - \hat{x}_a, \hat{x}_u + 1 - 1 + \hat{x}_a\} \\
&\geq \{\hat{x}_u + 1 - \hat{x}_u, \hat{x}_u + \hat{x}_a\} \\
&\geq \{1,\ 1\},
\end{aligned}$$

we see that $x'_u + x'_u \geq 1$ for either choice of v. I condensed down the two cases (for ε) above for brevity.

ii. $u \in A,\ v \in A$:

Note that it we can assume that $u \in X,\ v \in Y$, since the graph is bipartite. We then have that $x'_u + x'_v = (\hat{x}_u - \varepsilon) + (\hat{x}_v + \varepsilon) = \hat{x}_u + \hat{x}_u \geq 1$.

Therefore, we see that in any case, $x'_u + x'_v \geq 1$ for all $(u, v) \in E$.

2. x'' is a feasible solution:

By the work done to show that x' is a feasible solution, is is easy to see that x'' is also a feasible solution. By the construction of x' and x'', they assign identical values to $u \notin A$, but swapped for $u \in A$. By constructing a new bipartite graph, which simply swaps the places of vertex sets X and Y, we get that x'' in the new graph is the same as x' in the original graph, which was just shown to be a feasible solution. Therefore, x'' is a feasible solution in the original graph (by the proposed isomorphism).

Thus, by the work done above, we have shown that x' and x'' are feasible solutions. Furthermore, note that

$$\hat{x} = \frac{x' + x''}{2}$$

Since $\hat{x}$ was assumed to be a vertex solution - that is, there does not exist two feasible solutions for which $\hat{x}$ is a convex combination of the two - we assume that there does not exist an $\alpha \in (0, 1)$ for which

$$\hat{x} = \alpha x' + (1 - \alpha)x''.$$

But this was shown to be false, for $\alpha = \frac{1}{2}$. Thus, we get a contradiction, therefore, $|A| = 0$, implying that $\hat{x}_u \in \{0, 1\}$ for all $u \in X \cup Y$.

Problem 3, part 4

Design a polynomial-time algorithm that solves the Minimum Vertex Cover in bipartite graphs. Assume that you have an LP solver that given a linear program finds an optimal vertex solution $\hat{x}$.

Solution:

In the previous parts of this problem, we showed that our relaxation of the integer linear program returns the same minimal value as the integer linear program (ILP = OPT = LP). Furthermore, we are given that the ILP is constructed to match the minimum vertex cover. Therefore, our linear program will return the minimum vertex cover. Thus, to solve the minimum vertex cover for bipartite graphs, we need to form the linear program in the form given in part 1, call an oracle (the LP solver, say an interior point method) to solve the LP, then return the solution. We assume the LP solver will provide a solution in at most polynomial time. To form the standard problem, we note that the matrix A will just be the adjacency matrix, of the bipartite graph, and our b will be the vector of ones. we will then solve for the value x, and return the vertices which have a nonzero value - $R = \{x_u \in x : x_u = 1\}$.