CSCI270 Homework 8

Jacob Ma

November 13, 2022

Problem 1

You are trying to pick out the classes to take during your computer science major. There are n classes available, and for each of them, you know how rewarding it would be to take the class. That is captured by the integer number r_i, which could be positive or negative. (Some classes are not very rewarding. Of course, CSCI 270 has the largest r_i of all of them.) Sometimes, classes build on each other. While the program does not enforce prerequisites, if you are missing a prerequisite for a class, you have to spend extra time studying to catch up, which reduces the reward.

More specifically, for each pair of classes i, j such that i is a prerequisite for j, you are given an integer quantity $p_{i,j} > 0$ which gives you the amount of extra studying you will have to do for class j if you didn't also take class i. This $p_{i,j}$ is subtracted from your total reward if you take class j but not class i. We assume that prerequisites do not contain cycles: if you look at the graph of all edges (i, j) with $p_{i,j} > 0$, this graph contains no cycles.

To summarize the preceding discussion: if you take the set $S \subseteq \{1, \ldots, n\}$ of classes, your total reward is $R(S) := \sum_{i \in S} r_i - \sum_{j \notin S, i \in S} p_{j,i}$. We assume that there is no bound on the number of classes you are allowed to take.

Give (and analyze) a polynomial-time algorithm for finding a set S of classes maximizing $R(S)$.

Hint: In addition to the class material, you might find Section 7.11 in the textbook useful.

Algorithm 1: Class Picking Algorithm

Input: Classes $\{v_i\}_{i=1}^n$, Rewards $\{r_i\}_{i=1}^n$, Extra Work Without Pre-Requisite $\{p_{i,j}\}$

Output: Class Partition $\left(S, \overline{S}\right) \implies S$: select; $\overline{S}$: not select

1 Add a new source s and sink t ;

2 Connect s to all nodes v_i where $r_i \geqslant 0$ with capacity r_i ;

3 Connect all nodes v_i where $r_i < 0$ to t with capacity $-r_i < 0$;

4 All original edges $e = (v_i, v_j)$ where j is a prerequisite for i stay and have capacity $p_{j,i}$;

5 Use **Edmonds-Karp** to find minimum $s - t$ cut $\left(s + S, t + \overline{S}\right)$;

6 **return** the class node partition $\left(S, \overline{S}\right)$

Correctness Proof. Let's denote C as the set of all the coursers, C^+ as the set with $r_i \geqslant 0$, C^- as the set with $r_i < 0$. Intuitively, like the Network Segmentation problem discussed in class, we are trying to segment all the nodes where trying to choose nodes in C^+ to S and choose C^- to S^-.

Consider the equation

$$\begin{aligned}
R(S) :&= \sum_{i\in S} r_i - \sum_{j\notin S, i\in S} p_{j,i} \\
&= \sum_{i\in S\cap i\in C^+} r_i + \sum_{i\in S\cap i\in C^-} r_i - \sum_{j\notin S, i\in S} p_{j,i} \\
&= \sum_{i\in C^+} r_i - \sum_{i\in \overline{S}\cap i\in C^+} r_i + \sum_{i\in S, i\in C^-} r_i - \sum_{j\notin S, i\in S} p_{j,i} \\
&= \sum_{i\in C^+} r_i - \left(\sum_{i\in \overline{S}\cap i\in C^+} r_i - \sum_{i\in S, i\in C^-} r_i + \sum_{j\notin S, i\in S} p_{j,i} \right) \\
&= \sum_{i\in C^+} r_i - \left(\sum_{i\in \overline{S}\cap i\in C^+} r_i + \sum_{i\in S, i\in C^-} r'_i + \sum_{j\notin S, i\in S} p_{j,i} \right) \qquad \text{Let } r'_i = -r_i \text{ for } i \in C^-
\end{aligned}$$

Note that $\sum_{i\in C^+} r_i$ is a constant since we already know all node v_i such that $r_i \geqslant 0$, thus this value is a constant and will not be influenced by how we choose our classes.
Thus, trying to maximum $R(S)$ is equal to minimize $\left(\sum_{i\in \overline{S}\cap i\in C^+} r_i + \sum_{i\in S, i\in C^-} r'_i + \sum_{j\notin S, i\in S} p_{j,i}\right)$.
Knowing this, we are trying to construct a graph $G = (V, E)$ which minimum cut between $\left(S, \overline{S}\right)$ with capacity $\left(\sum_{i\in \overline{S}\cap i\in C^+} r_i + \sum_{i\in S, i\in C^-} r'_i + \sum_{j\notin S, i\in S} p_{j,i}\right)$.

Thus, we can construct a graph $G = (V, E)$ such that

$$\begin{aligned}
V &= \{s, t\} \cup \{v_i\}_{i=1}^n \\
E &= \{(s, v_i \in C^+) \text{ with capacity } r_i\} \cup \{(v_i \in C^-, t) \text{ with capacity } -r_i\} \\
&\quad \cup \{(v_i, v_j) \text{ where } j \text{ is prerequisite of } i \text{ with capacity } p_{i,j}\}
\end{aligned}$$

Here is a sample graph with only 3 classes, where $r_1, r_2 \geqslant 0$, $r_3 < 0$, and v_3 is a prerequisite of v_2.

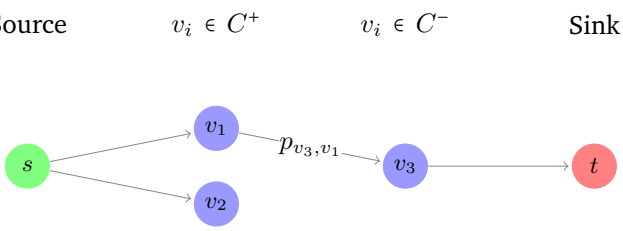

Figure 1: Sample Constructed $G = (V, E)$ with 3 courses

We want to show that any cut $\left(s + S, t + \overline{S}\right)$ in $G = (V, E)$ will have the capacity $\left(\sum_{i\in \overline{S}\cap i\in C^+} r_i + \sum_{i\in S, i\in C^-} r'_i + \sum_{j\notin S, i\in S} p_{ij}\right)$. Let's consider in a cut $\left(s + S, t + \overline{S}\right)$, only these three types of edges could be cut

(1) Edges from s to v_i for $v_i \in \overline{S}$. Since $s \in \{s + S\}$ and $v_i \in \{t + \overline{S}\}$, this is a valid cut. And also because there only exit edges between s and v_i where $v_i \in C^+$, and its capacity is r_i. Thus, the total cut in this category is

$$\sum_{i\in \overline{S}, i\in C^+} c(s, v_i) = \sum_{i\in \overline{S}, i\in C^+} r_i$$

(2) Edges from $v_i \in S$ to t. Since $t \in \{t + \overline{S}\}$ and $v_i \in \{s + S\}$, this is a valid cut. Similarly, there only exist edges between t and v_i where $v_i \in C^-$, and its capacity is $-r_i = r'_i$ by algorithm's construction. Thus, the

total cut in this category is

$$\sum_{i\in S, i\in C^-} c(v_i, t) = \sum_{i\in S, i\in C^-} r_i'$$

(3) Edges from $v_j \in \overline{S}$ to $v_i \in S$ and v_j is a prerequisite of v_i, such edge has capacity $p_{j,i}$. Thus, the total cut in this category is

$$\sum_{i\in S, j\in \overline{S}} c(v_i, v_j) = \sum_{i\in S, j\in \overline{S}} p_{j,i}$$

Thus, we now know that the total cut is summing up all three cases, which is

$$\sum_{i\in \overline{S}\cap i\in C^+} r_i + \sum_{i\in S, i\in C^-} r_i' + \sum_{j\notin S, i\in S} p_{j,i}$$

We know that Edmond-Karps will give minimum cut of graph $G = (V, E)$, giving the partition of $\left(S, \overline{S}\right)$ thus minimizing the equation above, which is equals to maximizing $R(S)$, thus the output will be correct.
We also know that our algorithm does not contain any while loop from line 1 to line 4, and Edmond-Karps will terminate, thus our algorithm will terminate.
Based on the reasoning above, the algorithm corrects maximized $R(S)$ and outputs correct partition $\left(S, \overline{S}\right)$.

□

Runtime Analysis. Line 1 takes constant time.
Line 2 and 3 each takes at worst $\mathcal{O}(n)$ by linking all the nodes to s and t.
Line 4 takes at worst if there exist an edge between every distinct pair of nodes, and in n^2 iterations each takes $\mathcal{O}(1)$ time to initiate capacity.
Line 5 applied Edmonds-Karp, which takes $\mathcal{O}\left(|V||E|^2\right) = \mathcal{O}\left(n \cdot (m+n)^2\right)$, wdwpolynomial.
While the Edmonds Karps dominate other operations, the total runtime is $\mathcal{O}\left(n(m+n)^2\right)$, which is polynomial.

□

Problem 2

In class, we used a duality-based proof to show that the Ford-Fulkerson algorithm returns a maximum flow and a minimum cut. Here, you will explore another duality-based proof of optimality, for a completely different problem (or rather, two completely different problems).
In fact, one of the problems is one we already solved, way in the beginning of class: unweighted interval selection. Just for concreteness, here it is again: you are given intervals $I_1 = [s_1, f_1], I_2 = [s_2, f_2], \ldots, I_n = [s_n, f_n]$. Your goal is to select as many of these intervals as possible without any selected pair overlapping. Formally, your goal is to select a set J of indices such that $I_j \cap I_{j'} = \varnothing$ for all $j, j' \in J, j \neq j'$. The goal is then to make $|J|$ as large as possible.
Here is the second problem: Interval Hitting. The input is the same, namely intervals $I_1, \ldots, I_n$. Your goal now is to select a set of points $T = \{t_1, t_2, \ldots, t_k\}$, such that for each interval, you have included at least one point. That is, $T \cap I_j \neq \varnothing$ for all j. Your goal is to achieve this with a smallest possible T. For a motivation, think about the last week of the semester. You want to go thank all of your CPs for their amazing help. So you want to visit SAL at a few different points in time, so as to hit the office hours of all CPs. Sometimes, office hours overlap, so you can thank multiple CPs with just one visit. Your goal is to thank all of them with

as few visits as possible.

1. Prove the following weak duality lemma:

   Lemma 0.0.1

   For all inputs $I_1, \ldots, I_n$, and any valid solution J for interval selection and T for interval hitting, we have $|J| \leq |T|$.

2. Use Lemma 1 to give an alternative proof (not using Greedy Stays Ahead or even explicit induction) that the greedy algorithm for Interval Selection finds an optimum solution.

   Just as with our analysis of Ford-Fulkerson, you should achieve this by modifying the algorithm to output not just the selected intervals (i.e., the set J), but also some set T of time points, and then relating the two.

Proof of Weak Duality Lemma. For all inputs $I_1, \cdots, I_n$, any valid solutions J for interval selection will not contain any pair of $I_i, I_j, i \neq j$ such that they overlap based on the definition. Thus, for any hitting point in T, it will **at most** hit 1 interval in J. In order to hit all intervals in I, and since $J \in \{I_i\}_{i=1}^n$, T would need at least hit all elements in J, which would take at least $\frac{|J|}{1} = |J|$ hitting. Thus $|J| \leqslant |T|$.

□

As for the alternative proof of the greedy algorithm, we first modify the algorithm by additionally adding a set T where

$$T_i := f_i \text{ where } I_i \in J.$$

In other word, T is composited of all the finish time of intervals in J. The detailed algorithm is as follows.

Algorithm 2: Interval Selection Greedy Algorithm Modified

Input: Intervals $\{I_i\}_{i=1}^n$ with corresponding $\{s_i\}_{i=1}^n$ and $\{f_i\}_{i=1}^n$

Output: Set of indices J maximizing number of non-overlapping intervals, Set of indices T minimizing number of points to hit all intervals

```
Sort intervals {I_i}_{i=1}^n by non-decreasing finish times {f_i}_{i=1}^n ;
Start with remaining interval R = all intervals, accepted intervals J = {}, accepted hitting points T = {} ;
while R! = ∅ do
    Let I_i a remaining interval with earliest finish time f_i;
    Add I_i to J, add f_i to T ;
    Remove from R the interval I_i and all intervals intersecting it.
return J and T
```

Alternative Proof of Greedy Interval Selection. We have shown in class and previously that this greedy algorithm will indeed select a maximum and valid J, and according to the definition, all intervals in J are disjoint (or the answer will not be valid). Since each element in T is added by an element in J with its finish time, there is a

bijective relationship between T and J. Thus we know

$$|J| = |T|$$

We have also shown that $|J| \leqslant |T|$, so according to the duality, the equality above could only be achieved if

$$\max_{\text{All Possible } J} |J| = \min_{\text{All Possible } T} |T|$$

So in order tom show J is the maximum possible J, it suffices to show that T is a valid set of interval hitting.

Now consider a random interval I_i, it could only be in one of the two cases below:

(1) $I_i \in J$: In this situation, the corresponding $f_i \in T$, thus I_i will at least be hit at f_i.

(2) $I_i \notin J$: In this situation, we will have the lemma

Lemma 0.0.2

$$s_i \leqslant f_j \leqslant f_i \qquad f_j \text{ is the finish time of some } I_j \in J$$

Proof of Lemma. This could be proved by contradiction. Suppose the equation is not true, it could only be one of the two cases since we know $s_i \leqslant f_i$:

(1) $f_j < s_i$: it means that we will have two disjoint interval I_j and I_i where

$$s_j \leqslant f_j < s_i \leqslant f_i$$

This means that I_j is a valid interval to be selected, but is not selected, contradicting the optimum of J, thus contradicting.

(2) $f_j > f_i$: it means that we would have chose f_i instead of f_j in this section according to the greedy algorithm since $f_i < f_j$. Thus, exists contradiction with the greedy algorithm.

The lemma follows the contradiction. □

Knowing this lemma, we will know that there must exist a f_j will hit I_i since $s_i \leqslant f_j \leqslant f_i$, and such f_j is added through some $I_j \in J$.

Thus, we know that every interval in $\{I_i\}_{i=1}^{n}$ will be hit by some element in T, thus T is valid.
So we could say that both J and T are both optimal solution by greedy algorithm for interval selection and interval hitting. □