CSC373 Fall'20

Assignment 4: Mathematical Monarch

Due Date: December 7, 2020, 11:59pm ET

Instructions

1. Typed assignments are preferred (e.g., PDFs created using LaTeX or Word), especially if your handwriting is possibly illegible or if you do not have access to a good quality scanner. Either way, you need to submit a single PDF named "hwk4.pdf" on MarkUS at https://markus.teach.cs.toronto.edu/csc373-2020-09

2. You will receive 20% of the points for a (sub)question when you leave it blank (or cross off any written solution) and write "I do not know how to approach this problem." If you leave it blank but do not write this or a similar statement, you will receive 10%. This does not apply to any bonus (sub)questions.

3. You may receive partial credit for the work that is clearly on the right track. But if your answer is largely irrelevant, you will receive 0 points.

Preface:
You're the monarch of a state, which you've ascended to the monarchy not because of the family you were born into, but because you always offer the most ingenious mathematical solutions to the everyday problems that the citizens bring to you. Today, four citizens have come to you seeking advice. Let's hope you can live up to their expectations!

Q1 [20 Points] Communications

A communication engineer in your realm is facing the following problem. There are n communication towers arranged in a circle: that is, tower i connected to tower $i+1$ for $i \in \{1, \ldots, n-1\}$, and tower n connected to tower 1. Two towers being "connected" means there are two cables between them, one for carrying communication traffic in each direction. Thus, there are $2n$ cables in total.

Every second, the central communication planning system receives a set of requests E. Each request k is given by a triplet (i_k, j_k, c_k), which indicates that c_k bytes of data must be sent from tower i_k to tower j_k; here, c_k is a positive real number and $i_k, j_k \in \{1, \ldots, n\}$ with $i_k \neq j_k$. This request can be fulfilled by routing the data either in the clockwise direction or in the anticlockwise direction; for example, if there are $n = 5$ towers and a request requires routing data from tower 2 to tower 4, then we can send it either via the route $2 \to 3 \to 4$ or via the route $2 \to 1 \to 5 \to 4$. The load on each cable e, denoted L_e, is the total amount of data sent across the cable. The overall load is $L = \max_e L_e$.

The engineer needs your help designing an algorithm which decides the direction (clockwise / anticlockwise) in which each request should be routed to minimize the overall load.

(a) [5 Points] Write a binary integer linear program which solves this problem exactly. Clearly indicate the meaning of your variables.

(b) [15 Points] Design an efficient 2-approximation algorithm which relaxes the ILP from part (a) to an LP and rounds its optimal solution.

[Hint: Even though the decision corresponding to each request k can be modeled as a single binary variable, you might find it more convenient to model it using two binary variables x_k and y_k.]

Q2 [20 Points] Task Scheduling

Congratulations! Your realm now has a brand new factory, which has n workers and m machines. There are ℓ tasks which must be performed each day in the factory. Each task i must be performed by a specific worker w_i on a specific machine m_i and this takes a specific amount of time p_i.

Given all this data, the factory owner needs to "schedule" the tasks, i.e., assign a start time $s_i \geq 0$ to each task i. Then, each task i will be performed by worker w_i on machine m_i during the interval $[s_i, s_i + p_i)$. The only constraint is that no worker can ever work on multiple tasks simultaneously and no machine can ever be used for multiple tasks simultaneously. It is OK for a worker or a machine to be idle between tasks.

The goal is to minimize the overall finish time T, i.e., the time by which all tasks finish. The factory owner, not being as mathematically astute as you, comes to you for advice. You begin by analyzing the following simple greedy algorithm:

Algorithm 1: Task Scheduling

```
while there still exist unscheduled tasks do
    For each unscheduled task i, compute the earliest time s_i at which it can start, i.e., the
      earliest time at which both worker w_i and machine m_i become available
    Let i* be the unscheduled task with the smallest s_{i*} (break ties arbitrarily)
    Schedule task i* to start at time s_{i*}
end
```


(a) [5 Points] Briefly argue that this algorithm produces a feasible schedule meeting the constraints.

(b) [5 Points] Briefly argue that the schedule produced by this algorithm has the following property: if worker j is idle at time t, then for every task i yet to be performed by the worker (i.e. with $w_i = j$ and $s_i > t$), the required machine m_i must be occupied for another task i' (i.e. there must exist i' such that $m_{i'} = m_i$ and $t \in [s_{i'}, s_{i'} + t_{i'})$). In other words, a worker is never idle if she could be working on a yet-to-be-done task.

(c) [10 Points] Use part (b) to prove that this algorithm achieves a 2-approximation.

[Hint: It might be helpful to relate four quantities:

- T (the overall finish time of the greedy schedule),
- T^* (the minimum possible overall finish time of any schedule),
- $W_{\max} = \max_j W_j$, where W_j is the total processing time of all jobs i with $w_i = j$,
- $M_{\max} = \max_k M_k$, where M_k is the total processing time of all jobs i with $m_i = k$.

One way to relate these quantities is to focus on a task i that finishes last in the schedule produced by the greedy algorithm and relate the time spent by worker w_i up to the start time of i to the above quantities. Note that $T = s_i + p_i$.]

Q3 [20 Points] Toll Booths

Congratulations! Your realm now has roads. The road network can be represented as an undirected graph $G = (V, E)$ with edges in E representing the roads and nodes in V referred to as junctions. The urban planner wants to build toll booths at the fewest possible junctions such that at least one of the two ends of each road has a toll booth. Upon hearing the problem, you instantly smile: *this is the (unweighted) minimum vertex cover problem.*

Tapping into your infinite wisdom, you tell the planner that this problem is NP-complete, but you know a simple 2-approximation algorithm. The planner pushes back: *Can't you do any better?* You begin to think: in the worst case, 2-approximation is known to be the best possible assuming a widely believed conjecture ("unique games conjecture"). But what if our graph G has a nice structure? Then it hits you: your graph G is planar, and, by a popular theorem, every planar graph has a 4-coloring (i.e. a function $c : V \to \{0, 1, 2, 3\}$ such that $c(u) \neq c(v)$ for all $(u, v) \in E$).

Suppose you are given a 4-coloring c of G. For $t \in \{0, 1, 2, 3\}$, define V_t to be the set of vertices v with $c(v) = t$. Without loss of generality, assume $|V_3| \geq |V_2| \geq |V_1| \geq |V_0|$. Your job is to use this 4-coloring to find a better vertex cover of G. Consider the ILP for minimum vertex cover:

$$\begin{aligned} \min \ & \textstyle\sum_{v \in V} x_v \\ & x_u + x_v \geq 1, \forall (u, v) \in E \\ & x_u \in \{0, 1\}, \forall v \in V \end{aligned}$$

It is known that the LP relaxation of this ILP has the following property: it has an optimal solution x^* in which $x^*_v \in \{0, \frac{1}{2}, 1\}$ for each $v \in V$. Suppose such an optimal solution x^* is also given to you. Given x^* and c, you define $S = \{v \in V : x^*_v = 1 \vee (x^*_v = \frac{1}{2} \wedge v \notin V_3)\}$.

(a) [10 Points] Argue that S is indeed a vertex cover of G.

(b) [10 Points] Prove that the algorithm achieves a 3/2-approximation (i.e. $|S|$ will be at most 3/2 times the size of the smallest vertex cover).

Q4 [20 Points] CNF?

An upstanding citizen from your realm, Shah, comes to you with the following problem, but refuses to say where this problem comes from. You don't like this very much as you've grown fond of hearing the stories, but being a good monarch, you still help him out.

Suppose, he says, you are given a CNF formula φ consisting of clauses $C_1, \ldots, C_m$ over variables $x_1, \ldots, x_n$. Your goal is to find an assignment of variables satisfying the maximum number of clauses (MAX-SAT). For each clause C_j, let P_j denote the set of indices i such that C_j contains the literal x_i and N_j denote the set of indices i such that C_j contains the literal $\bar{x}_i$.

(a) [5 Points] Fill in the blanks in the following ILP for this problem:

maximize	_____
subject to	_____ for all $j \in [m]$
	$y_i, z_j \in \{0, 1\}$ for all $i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$

(b) [7 Points] Suppose you are given a function $f : [0, 1] \to \mathbb{R}$ such that $1 - 4^{-y} \leq f(y) \leq 4^{y-1}$ for all $y \in [0, 1]$. Consider the following approximation algorithm for this problem.

Algorithm $\mathcal{A}$:

1. Solve the LP relaxation of the above ILP. Let the optimal solution be (y^*, z^*).
2. Construct an assignment by independently setting each x_i to TRUE with probability $f(y_i^*)$.

Show that for each clause C_j, the probability that C_j is not satisfied is at most $4^{-z_j^*}$.

(c) [8 Points] Use part (b) to show that $\mathcal{A}$ is a 4/3-approximation algorithm.

[Hint: You can assume without proof that the function $g(z) = 1 - 4^{-z}$ is a concave function on $[0, 1]$, and hence, satisfies $g(\lambda) \geq \lambda \cdot g(1)$ for all $\lambda \in [0, 1]$.]