CS 170 Homework 13

Due **Sunday 5/4/2025, at 10:00 pm (grace period until 11:59pm)**

1 Study Group (1 point)

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Multiway Cut (10 points)

In the multiway cut problem, we are given a graph $G = (V, E)$ with k special vertices $s_1, s_2, \ldots, s_k$. Our goal is to find the smallest set of edges F which, when removed from the graph, disconnect the graph into at least k components, where each s_i is in a different component. When $k = 2$, this is exactly the min s-t cut problem, but if $k \geq 3$ the problem becomes NP-hard.

Consider the following algorithm: Let F_i be the set of edges in the minimum cut with s_i on one side and all other special vertices on the other side. Output F, the union of all F_i. Note that this is a multiway cut because removing F_i from G isolates s_i in its own component.

(a) (3 points) Explain how each F_i can be found in polynomial time.

(b) (4 points) Let F^* be the smallest multiway cut. Consider the connected components that removing F^* disconnects G into, and let C_i be the set of vertices in the component with s_i. Let F_i^* be the set of edges in F^* with exactly one endpoint in C_i. How many different F_i^* does each edge in F^* appear in? Which is larger: F_i or F_i^*?

(c) (3 points) Using your answer to the previous part, show that $|F| \leq 2|F^*|$.

(d) **Extra Credit (2 points):** how could you modify this algorithm to output F such that $|F| \leq (2 - \frac{2}{k})|F^*|$?

3 Relaxing Integer Linear Programs (17 points)

As discussed in lecture, Integer Linear Programming (ILP) is NP-complete. In this problem, we discuss attempts to approximate ILPs with Linear Programs and the potential shortcomings of doing so.

Throughout this problem, you may use the fact that the ellipsoid algorithm finds an optimal vertex (and corresponding optimal value) of a linear program in polynomial time.

(a) (3 points) Suppose that $\vec{x}_0$ is an optimal point for the following arbitrary LP:

$$\begin{aligned} \text{maximize } & c^\top x \\ \text{subject to: } & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Show through examples (i.e. by providing specific canonical-form LPs and optimal points) why we cannot simply (i) round all of the element in $\vec{x}_0$, or (ii) take the floor of every element of $\vec{x}_0$ to get good integer approximations.

(b) (2 points) The MATCHING problem is defined as follows: given a graph G, determine the size of the largest subset of disjoint edges of the graph (i.e. edges without repeating incident vertices).

Find a function f such that:

$$\begin{aligned} \text{maximize } & f(\vec{x}) \\ \text{subject to: } & \sum_{e \in E, v \in e} x_e \leq 1 \qquad \forall v \in V \\ & 0 \leq x_e \leq 1 \qquad \forall e \in E \end{aligned}$$

is an LP relaxation of the MATCHING problem. Note that the ILP version (which directly solves MATCHING) simply replaces the last constraint with $x_e \in \{0, 1\}$.

(c) (6 points) It turns out that the polytope of the linear program from part (b) has vertices whose coordinates are all in $\{0, \frac{1}{2}, 1\}$. Using this information, describe an algorithm that approximates MATCHING and give an approximation ratio with proof.

Hint: round up, then fix constraint violations.

(d) (4 points) There is a class of linear program constraints whose polytopes have only integral coordinates. Let $\mathcal{P}_{>2,\text{odd}}(V)$ be the set of subsets of the vertices with size that is odd and greater than 2. It turns out that, if we simply add to the LP from part (b) the following constraints:

$$\sum_{e \in E(S)} x_e \leq \frac{|S| - 1}{2} \qquad \forall S \in \mathcal{P}_{>2,\text{odd}}(V),$$

then all vertices of the new feasible region polytope are integral. First, interpret these constraints in words and explain why it still describes the MATCHING problem. Then, explain what this result implies about approximating ILPs with (special) LPs.

(e) (2 points) Why doesn't the observation in part (d) imply that MATCHING $\in$ P?

Hint: what is the size of set $\mathcal{P}_{>2,\text{odd}}(V)$*?*

4 Chameleon Vectors (22 points)

Let L be a vector of integers in $[-M, M]^n$ given to us as input, where $M < 2^n/(4n)$. For any other vector x, we will use $\langle L, x\rangle$ to denote $\sum_{i=1}^n L_i x_i$. In this problem, we look to define a randomized algorithm that finds two distinct vectors, x_1 and x_2, in $\{\pm 1\}^n$ such that $\langle L, x_1\rangle = \langle L, x_2\rangle$, i.e. they "look indistinguishable" to L.

(a) (2 points) Prove that there exists distinct $x_1, x_2 \in \{\pm 1\}^n$ such that $\langle L, x_1\rangle = \langle L, x_2\rangle$. *Hint: try to prove this using the pigeonhole principle.*

(b) (2 points) Let $\mathbf{x}$ be sampled uniformly at random from $\{\pm 1\}^n$. Use Chebyshev's inequality to prove that:

$$\Pr[|\langle L, \mathbf{x}\rangle| > 10M\sqrt{n}] \leq \frac{1}{100}.$$

Hint: For this part, as well as future parts, Discussion 13 Q3 may be useful for reference.

(c) (5 points) Let $\mathbf{Y}_1, \ldots, \mathbf{Y}_{\lceil 2k \ln r\rceil}$ each be $2k \ln r$ independent and identical random variables that are equal to 1 with probability p and 0 otherwise. Prove that if $p \geq \frac{1}{k}$, then:

$$\Pr[\mathbf{Y}_1 + \cdots + \mathbf{Y}_{\lceil 2k \ln r\rceil} < 2] \leq \frac{2}{r}.$$

Hint: you may use the fact that $\left(1 - \frac{1}{n}\right)^n \leq \frac{1}{e}$ without proof. If you're stuck, try exploring using this property for various n.

Now, define $\ell = \frac{20M\sqrt{n}+1}{0.99}$. Consider the following algorithm: sample $\lceil 2\ell \ln n\rceil$ independent vectors, $\mathbf{x}_1, \cdots, \mathbf{x}_{\lceil 2\ell \ln n\rceil}$, and store $(\langle L, \mathbf{x}_i\rangle, \mathbf{x}_i)$ for every i. Then, sort these by their $\langle L, \mathbf{x}_i\rangle$ value. Lastly, iterate through the list and find the first adjacent pair of pairs, $(\langle L, \mathbf{x}_j\rangle, \mathbf{x}_j)$ and $(\langle L, \mathbf{x}_{j+1}\rangle, \mathbf{x}_{j+1})$ such that $\langle L, \mathbf{x}_j\rangle = \langle L, \mathbf{x}_{j+1}\rangle$ but $\mathbf{x}_j \neq \mathbf{x}_{j+1}$. Output $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$. Note that the algorithm fails if no such adjacent pair of pairs exists in the sorted list.

(d) (3 points) Prove that the runtime of this algorithm is $O(Mn^{3/2} \log n)$. You may assume that sampling k bits takes $O(k)$ time.

We now aim to analyze the probability that this algorithm succeeds. In particular, over the remainder of the problem, we aim to show that the probability of failure is $O(1/\sqrt{n})$. Define F to be the most frequent value of $\langle L, \mathbf{x}_i\rangle$ in the range $[-10M\sqrt{n}, 10M\sqrt{n}]$ over all $\mathbf{x}_i$.

(e) (4 points) Show that $\Pr[\exists\, i \neq j \text{ s.t. } \langle L, \mathbf{x}_i\rangle = \langle L, \mathbf{x}_j\rangle = F] \geq 1 - \frac{2}{n}$.

Hint: Why was ℓ chosen as such? Parts (b) and (c) will be helpful here.

(f) (3 points) Show that there exists some constant $c \in \mathbb{R}$ such that, no matter what M and n are, we have:

$$\Pr[\mathbf{x}_i = \mathbf{x}_j \mid \langle L, \mathbf{x}_i\rangle = \langle L, \mathbf{x}_j\rangle = F] \leq \frac{c}{\sqrt{n}}.$$

Hint: At least how many $v \in \{\pm 1\}^n$ satisfy $\langle L, v\rangle = F$?

(g) (3 points) Using parts (e) and (f), conclude that this algorithm has a failure probability in $O(1/\sqrt{n})$.

Hint: argue why it must be the case that

$$\Pr[\text{algorithm succeeds}] \geq \Pr[\exists i \neq j \text{ s.t. } \mathbf{x}_i \neq \mathbf{x}_j \text{ and } \langle L, \mathbf{x}_i \rangle = \langle L, \mathbf{x}_j \rangle = F].$$

5 [Coding] TSP Usually-Fast Algorithm (8 points)

For this week's homework, you'll implement code for the Traveling Salesman Problem that runs efficiently in most cases (although not in the worst case). There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw13` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   https://github.com/Berkeley-CS170/cs170-sp25-coding

   and navigate to the `hw13` folder. Refer to the `README.md` for local setup instructions.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "Homework 13 Coding Portion."
- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:

  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

