CS 170 Homework 12

Due **Saturday 11/22/2025, 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, explicitly write "none".

2 Local Search for Max Cut (8 points)

Sometimes, local search algorithms can give good approximations to NP-hard problems. Recall that in the Max-Cut problem, we have an unweighted graph $G = (V, E)$ and we want to find a cut (S, T) that maximizes the number of edges "crossing" the cut (i.e., with one endpoint in each of S, T). Consider the following local search algorithm:

1. Start with any cut (e.g. $(S, T) = (V, \emptyset)$).
2. While there is some vertex $v \in S$ such that more edges cross $(S \setminus \{v\}, T \cup \{v\})$ than (S, T) (or some $v \in T$ such that more edges cross $(S \cup \{v\}, T \setminus \{v\})$ than (S, T)), move v to the other side of the cut.

Now, let us prove a couple of guarantees that this algorithm achieves.

(a) (3 points) Give an upper bound on the number of iterations this algorithm can run for (i.e., the total number of times we move a vertex).

(b) (5 points) Show that when this algorithm terminates, it finds a cut where at least half the edges in the graph cross the cut.

Hint: when we move v from S to T, v must have more neighbors in S than T. What does this observation suggest about the neighbors of each vertex once the algorithm terminates? Then, what can we say about the number of edges crossing the cut vs. the number of edges within each side of the cut?

3 Approximating Independent Set (11 points)

In the maximum independent set (MIS) problem, we are given a graph $G = (V, E)$, and our goal is to find the largest set of vertices such that no two vertices in the set are connected by an edge. For this problem, we will assume the degree of all vertices in G is bounded by d (i.e. $\forall v \in V,\ \deg(v) \leq d$).

(a) (3 points) Call a set $S \subseteq V$ a *maximal independent set* if its vertices are pairwise non-adjacent and adding any new vertex to S would make the set no longer an independent set (i.e., no longer satisfy the pairwise non-adjacency property). Give an efficient greedy algorithm for creating a maximal independent set.

(b) (3 points) Consider the following algorithm:

```
1: procedure IndSetApprox(V, E)
2:     Create some maximal independent set I
3:     return I
```

Provide an example where the above algorithm does not give an optimal solution.

(c) (5 points) Provide an approximation ratio for the algorithm from part (b) in terms of $|V|$, $|E|$, and/or d. Briefly justify your answer.

4 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 elements 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 they still describe 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 $\mathcal{P}_{>2,\text{odd}}(V)$*?*

5 Maximum Coverage (13 points)

In the maximum coverage problem, we have m subsets of the set $\{1, 2, \ldots, n\}$, denoted $S_1, S_2, \ldots S_m$. We are given an integer k, and we want to choose k sets whose union is as large as possible.

(a) (3 points) Briefly argue that, if we can solve the Maximum Coverage problem optimally, then we can also solve the Set Cover problem. (Note: this implies that Maximum Coverage is NP-hard.)

(b) Let us examine the following greedy algorithm which proceeds in k steps as follows: in each step, pick the subset that covers the maximum number of (currently) uncovered elements.

1. (2 points) Give a counterexample for which the greedy algorithm does not give an optimal solution.

2. (3 points) Show that, when $k = 2$, this algorithm is a 3/4-approximation algorithm for the problem (i.e., the two sets picked by the algorithm cover 3/4 as many elements as the optimal solution).

   Hint: It may be helpful to look at the hint from part (c).

3. (5 points) Generalize the analysis above to show that, for every k, the greedy algorithm is an $\left(1 - (1 - 1/k)^k\right)$-approximation algorithm for Maximum Coverage.

   Hint: Let x_i denote the number of elements covered by the greedy solution after picking i subsets. Try to show that $x_{i+1} - x_i \geq (OPT - x_i)/k$ where OPT is the total number of elements covered by the optimal solution. You might find it useful to consider the quantity $y_i = OPT - x_i$ as well.

