CS 170 Homework 12

Due **Monday 4/22/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

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

2 Approximating Independent Set

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. $\forall v \in V,\ \deg(v) \leq d$).

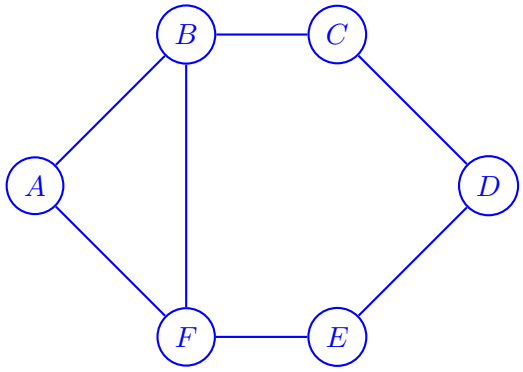
Consider the following greedy algorithm to approximate the maximum independent set:

```
1: procedure GREEDY-MIS(V, E)
2:     I ← ∅
3:     while G ≠ ∅ do
4:         choose a vertex v in G
5:         I = I ∪ {v}
6:         remove v and all its neighbors from G
7:     return I
```

(a) Provide an example where the greedy approximation does not give the optimal solution.

Solution:

Consider the graph below.



The maximum independent set would be of size 3, and can be composed of A, C, and E. However, if we follow the greedy algorithm, it's possible we choose vertex B, then one vertex from $\{D, E\}$, yielding an independent set of size 2.

(b) Provide an approximation ratio for the given greedy algorithm in terms of $|V|$, $|E|$, and/or d. Briefly justify your answer.

Solution: At each step of the while loop, one vertex is added to the independent set I. Also, at each step, at most $d+1$ vertices are removed from G, since the maximum degree of a vertex in the graph is d, and both v and its neighbors are removed from G. Then, there are at least $|V|/(d+1)$ iterations of the while loop. Since the maximum independent set OPT must be a subset of V, $|OPT| \leq |V|$. Then $|I| \geq \frac{|V|}{d+1} \geq \frac{|OPT|}{d+1}$. Hence, the approximation ratio is $d+1$.

3 $\sqrt{n}$ coloring

(a) Let G be a graph of maximum degree δ. Show that G is $(\delta+1)$-colorable.

(b) Suppose $G = (V, E)$ is a 3-colorable graph. Let v be any vertex in G. Show that the graph induced on the neighborhood of v is 2-colorable.

Note: the graph induced on the neighborhood of v refers to the following subgraph:

$$G' = (V' = \text{neighbors of } v, E' = \text{all edges in } E \text{ with both endpoints in } V').$$

(c) Give a polynomial time algorithm that takes in a 3-colorable n-vertex graph G as input and outputs a valid coloring of its vertices using $O(\sqrt{n})$ colors. Prove that your algorithm is correct.

Hint: think of an algorithm that first assigns colors to "high-degree" vertices and their neighborhoods, and then assigns colors to the rest of the graph. The previous two parts might be useful.

Solution:

(a) Let the color palette be $[\delta+1]$. While there is a vertex with an unassigned color, give it a color that is distinct from the color assigned to any of its neighbors (such a color always exists since we have a palette of size $\delta+1$ but only at most δ neighbors).

(b) In an induced graph, we only care about the direct vertices adjacent to v and v itself. In any 3-coloring of G, v must have a different color from its neighborhood and therefore the neighborhood must be 2-colorable.

(c) While there is a vertex of degree $\geq \sqrt{n}$, choose any such vertex v, pick 3 colors, use one color to color v, and the remaining 2 colors to color the neighborhood of v (2-coloring is an easy problem). Never use these colors ever again and delete v and its neighborhood from the graph. Since each step in the while loop deletes at least $\sqrt{n}$ vertices, there can be at most $\sqrt{n}$ iterations. This uses only $3(\sqrt{n}+1)$ colors. After this while loop is done we will be left with a graph with max degree at most $\sqrt{n}$. We can $\sqrt{n}+1$ color with fresh colors this using the greedy strategy from the solution to part a. The total number of colors used is $O(\sqrt{n})$.

4 Multiway Cut

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) Explain how each F_i can be found in polynomial time.

(b) Let F^* be the smallest multiway cut. Consider the 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 and F_i^*?

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

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

Solution:

(a) Consider adding a vertex t to the graph and connecting t to all special vertices except s_i with infinite capacity edges. Then F_i is the minimum s_i-t cut, which we know how to find in polynomial time.

(b) Each edge in F^* appears in exactly two of the sets F_i^*.

Note that F_i^* is the set of edges in a cut which disconnects s_i from the other special vertices. Then by definition F_i has fewer edges than F_i^* since F_i is the minimum cut disconnecting s_i from all other special vertices.

(c) We combine the answers to the previous part and note that F's size is at most the total size of all F_i to get:

$$|F| \leq \sum_i |F_i| \leq \sum_i |F_i^*| = 2|F^*|$$

(d) To get the $(2 - \frac{2}{k})$-approximation, after computing all F_i, we instead output F as the union of all F_i except for the one with the most edges. Let this be F_j. This is still a multiway cut because each s_j is still disconnected from all other s_i. Then:

$$|F| \leq \sum_{i \neq j} |F_i| \leq \left(1 - \frac{1}{k}\right) \sum_i |F_i| \leq \left(1 - \frac{1}{k}\right) \sum_i |F_i^*| = \left(2 - \frac{2}{k}\right) |F^*|$$

5 Relaxing Integer Linear Programs

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) 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 (1) round all of the element in $\vec{x}_0$, or (2) take the floor of every element of $\vec{x}_0$ to get good integer approximations.

(b) 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:

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

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) 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) 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 polytope are integral. First, interpret this constraint in words and explain why it still describes the MATCHING problem. Then, explain what this result implies about approximating ILPs with (special) LPs.

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

Solution:

(a) Neither of these work because our rounding/truncating could leave us with points outside of the feasible region. For example, if the constraints of the LP are:

$$\begin{aligned} \text{Maximize } & x + y \\ \text{subject to: } & 3x \leq 2 \\ & 3y \leq 2 \\ & -3x \leq -1 \\ & -3y \leq -1 \\ & x, y \geq 0 \end{aligned}$$

Then, the optimal point is at $\left(\frac{2}{3}, \frac{2}{3}\right)$, which rounds to $(1, 1)$ outside the feasible region. Then, rounding down would leave us with $(0, 0)$, which is not in the feasible region.

(b) Each x_e variable represents whether we include edge e in our matching. In our relaxation, we can allow for partially choosing edges. So, we want $f = \sum_e x_e$, meaning the maximum number of edges chosen in our matching.

(c) If we simply run the ellipsoid algorithm, we'll get an optimal vertex in polynomial time whose coordinates are within $\{0, \frac{1}{2}, 1\}$. Let's call this solution LP. We round the $\frac{1}{2}$ values to 0 or 1 in the following way:

while there still exists $e = (u, v)$ s.t. $x_e = \frac{1}{2}$ **do**
  $x_e \leftarrow 1$
  for all edges $e' \neq e$ incident to one of u or v **do**
    $x_{e'} = 0$

Let's call this rounded solution $\widetilde{LP}$.

We first bound $\widetilde{LP}$ from above. Since $\widetilde{LP}$ only contains the values $\{0, 1\}$ and satisfies the other constraints, it is contained in the ILP's feasible region. Thus, $\widetilde{LP} \leq OPT$.

Then, we bound $\widetilde{LP}$ from below. During each while loop iteration of the rounding process, we will modify up to 3 edges (e and 2 edges e'). In the worst case, the sum of the values of the three modified edges goes from $\frac{3}{2}$ to 1, and each edge is modified at most once. This gives us $\widetilde{LP} \geq \frac{2}{3} LP$. Finally, since the LP formulation is a relaxed version of ILP, we have

$$\widetilde{LP} \geq \frac{2}{3} LP \geq \frac{2}{3} OPT.$$

Therefore, we have $\frac{2}{3} OPT \leq \widetilde{LP} \leq OPT$, which yields an approximation ratio of $\frac{3}{2}$.

(d) This constraint states that for all odd-sized subsets S of the vertices, the number of edges in the matching is at most $\frac{1}{2}(|S| - 1)$. For example, out of any three vertices, at most one whole edge can be used in our matching, which disallows solutions where three adjacent edges are assigned to $\frac{1}{2}$. This property holds for every valid (integer)

matching since breaking this property means that at least one vertex in S is incident to 2 edges (one way we can see this is the Pigeonhole Principle).

Given this new LP, we can run the ellipsoid algorithm to get an optimal vertex in polynomial time with respect to n and m (number of variables and constraints). Since all vertices are integral, then an optimal vertex for the linear program is also an optimal vertex for the integer linear program. Therefore, we can simply take an output to the LP to solve the ILP, yielding an approximation ratio of 1. We can always solve the MATCHING ILP by converting it to an LP!

(e) Note that $\mathcal{P}_{>2,\text{odd}}$ has size $O\left(2^{|V|-1}\right) = O\left(2^{|V|}\right)$, meaning that there are $O\left(2^{|V|}\right)$ constraints. On the other hand, the size of G is $O\left(|V|^2\right)$. Therefore, we are passing an instance to the ellipsoid algorithm that is *exponential* in the size of the instance to MATCHING. Even though running the ellipsoid algorithm (or any other polynomial-time algorithm for solving LPs) is polynomial in the size of its input, it is actually exponential in the input to MATCHING.