CSE 565: Algorithm Design and Analysis **Due 2022-11-18T23:59**

Problem Set 5

Instructor: Dr. Antonio Blanca
TA: Jeremy Huang

Release Date: 2022-11-04

Notice: Type your answers using LaTeX and make sure to upload the answer file on Gradescope before the deadline. Recall that for any problem or part of a problem, you can use the "I'll take 20%" option. For more details and the instructions read the syllabus.

Problem 1. Max Cut

In the `MAXIMUM CUT` problem we are given an undirected graph $G = (V,E)$ with a weight $w(e)$ on each edge, and we wish to separate the vertices into two sets S and $V - S$ so that the total weight of the edges between the two sets is as large as possible. For each $S \subseteq V$ define $w(S)$ to be the sum of all $w(e)$ over all edges u, v such that $|S \cap \{u,v\}| = 1$. Obviously, `MAX CUT` is about maximizing $w(S)$ over all subsets of V . Consider the following local search algorithm for `MAX CUT`:

start with any $S \subseteq V$
while there is a subset $S' \subseteq V$ such that
$||S'| - |S|| = 1$ and $w(S') > w(S)$ do: set $S = S'$

(a) Show that this is an approximation algorithm for `MAX CUT` with ratio 2.
(b) What is the time complexity of this algorithm?

Problem 2. Tree Jumper

In the `MINIMUM STEINER TREE` problem, the input consists of: a complete graph $G = (V,E)$ with distances d_{uv} between all pairs of nodes; and a distinguished set of terminal nodes $V' \subseteq V$. The goal is to find a minimum-cost tree that includes the vertices V'. This tree may or may not include nodes in $V - V'$.

Suppose the distances in the input are a metric (recall the definition on textbook page 273). Show that an efficient ratio-2 approximation algorithm for `MINIMUM STEINER TREE` can be obtained by ignoring the nonterminal nodes and simply returning the minimum spanning tree on V'. (Hint: Recall our approximation algorithm for the TSP.)

Problem 3. Partial Independence

Give a polynomial running time $\left(\frac{1}{d+1}\right)$-approximation algorithm for maximum independent set problem when the input graph G in undirected and the degree of all vertices of G is at most $d \in \mathbb{N}$.

Problem 4. Bad Packing

Consider the following bin packing problem: given n items of sizes $a_1, a_2, ..., a_n$, find a way to pack them into unit-sized bins so that the number of bins needed is minimized. Consider a greedy algorithm: process all items in an arbitrary order; for the current k-th item, if there exists a partially packed bin that can further fit it, then put it in that bin; otherwise, open a new bin and put the k-th item in it. Show that the approximation ratio of above greedy algorithm is at most 2. Design an instance such that above algorithm performs at least as bad as $5/3 \cdot OPT$.

Problem 5. Lightning Punch

Consider a minimal-weighted hitting set problem define as follows. We are given a set $U = \{u_1, u_2, ..., u_n\}$ and a collection $\{B_1, B_2, ..., B_m\}$ of subsets of U, i.e., B_j is a subset of U. Also, each element $u_i \in U$ has a weight $w_i \geq 0$. The problem is to find a hitting set $H \subset U$ such that the total weight of the elements in H is as small as possible, i.e., to minimize $\sum_{u_i \in H} w_i$. Note that we say H is a hitting set if $H \cap B_j \neq 0$ for any $1 \leq j \leq m$. Let $b = \max_{1 \leq i \leq m} |B_i|$ be the maximum size of any of the sets $\{B_1, B_2, ..., B_m\}$.

(a) Design a b-approximation algorithm for above problem using the LP + rounding schema.

(b) Design a b-approximation algorithm for above problem using the primal-dual schema.

Problem 6. UU Max Cut

Given an undirected graph $G = (V, E)$, we seek a cut $(A, B = V \backslash A)$ such that the number of cut-edges, denoted as $|E(A,B)|$ where $E(A,B) := \{(u,v) \in E \mid u \in A, v \in B\}$, is maximized. Consider the following greedy algorithm for this problem: process all vertices in V following an arbitrary order $\{v_1, v_2, \cdots, v_n\}$, and for the current vertex v_k, put it in B if $|N_A(v_k)| \geq |N_B(v_k)|$ and in A otherwise, where $N_A(v_k)$ is defined as $\{v_i \mid i < k, v_i \in A, (v_i, v_k) \in E\}$, and $N_B(v_k) := \{v_i \mid i < k, v_i \in B, (v_i, v_k) \in E\}$. Intuitively, this algorithm assigns a vertex to the different side with the majority of its adjacent vertices being already assigned to. Prove that this algorithm is an approximation algorithm. You will need to guess the approximation ratio, prove it, and design a tight example.