CSCI 570 - Summer 2023 - HW 8

June 20, 2023

Graded Problems

Problem 1

Consider the following heuristic to compute a vertex cover of a connected graph G. Pick an arbitrary vertex as the root and perform depth first search. Output the set of non-leaf vertices (vertices of degree greater than 1) in the resulting depth-first search tree.

- Show that the output is a vertex cover for G.(10pts)

  Solution: Let $G = (V, E)$ denote the graph, $T = (V, E_0)$ the resulting depth first search tree, L the set of leaf vertices in the depth-first search tree and $N = V - L$ the set of non leaf vertices. Assume there is an edge $e = (u, v)$ in the edge set E that is not covered by N (which is the vertex cover). This implies that both u and v are in L. Without loss of generality, assume that DFS explored u first. At this stage, since e was available to DFS to leave u, DFS would have left u to explore a new vertex, thereby making u a non-leaf node. Hence our assumption is incorrect that both u and v are in L. Therefore N does indeed cover every edge in E.

- How good of an approximation to a minimum vertex cover does this heuristic assure? That is, upper bound the ratio of the number of vertices in the output to the number of vertices in a minimum vertex cover of C. (15pts)

  Solution: Let $G = (V, E)$ denote the graph, $T = (V, E_0)$ the resulting depth first search tree, L the set of leaf vertices in the depth-first search tree and $N = V - L$ the set of non leaf vertices.

  We next create a matching M in G from the structure of T as follows. Recall that a matching is a set of edges such that no two distinct edges in the set share a vertex. For every vertex u in N, pick one edge that connects u to one of its descendants in T and call it c_u .We call the set of odd level non-leaf vertices in T as ODD and the set of even level non-leaf vertices $EVEN$. If the set ODD is bigger than or equal to the set $EVEN$, set $BIG := ODD$; else set $BIG := EVEN$. Since the total number of non leaf vertices in the tree T is $|N|$, BIG has size at least $|N|/2$. Without loss of generality, we assume that $BIG = EVEN$. Now the edge-pair set $\{(e_1, e_2)|$the common end of e_1 and e_2 belongs to $BIG\}$ has at least the size of BIG of disjoint edge pairs, which is $\frac{|N|}{2}$, to cover every edge in the above set, the optimal vertex cover A has to contain at least $\frac{|N|}{2}$ vertices, which is because in a matching no two distinct edges share a vertex . Thus A has to contain at least $|N|/2$ vertices while our solution has $|N|$ vertices. Hence our solution is at worst a 2-approximation.

It can be shown that our solution can be indeed twice as bad by considering $E = \{(a, b), (b, c)\}$ with DFS rooted at a

Rubric (25pt):

(a) 10 pt: Correctly proof that for all edges at least one of their vertices exist in vertex cover.

(b) 5 pt: Correctly state that the size of the vertex cover provided by the heuristic is not larger than two times of the optimal vertex cover. 10 pts: Correctly explain the reason of 2-approximation.

Problem 2

A Max-Cut of an undirected graph $G = (V, E)$ is defined as a cut $C_{\max}$ such that the number of edges crossing $C_{\max}$ is the maximum possible among all cuts. Consider the following algorithm.

- Start with an arbitrary cut C;
- While there exists a vertex v such that moving v from one side of C to the other increases the number of edges crossing C, move v and update C.

Questions:

- Does the algorithm terminate in time polynomial in $|V|$?

  Solution: The algorithm will terminate within at most $|E|$ steps. Note that the cut value belongs to the set $\{0, 1, 2, \ldots, |E|\}$. After each step, the cut value increases by at least 1. Therefore, the above process ends within $|E| = O(|V|^2)$ steps.

- Prove that the algorithm is a 2-approximation, that is the number of edges crossing $C_{\max}$ is at most twice the number crossing C.

  Solution: First, we upper bound the optimal max-cut value. Specifically, $OPT \leq |E| = \frac{1}{2}\sum_{x \in V} deg(x)$. To lower bound the cut value of our algorithm, let the cut be (A, B) where $A \cup B = V$. For each node $x \in A$ or $x \in B$, let E_x be the number of edges from x to B. According to our algorithm, we know that $E_x \geq \frac{1}{2}deg(x)$. Taking summation over all nodes in A and B and letting the cut value of the algorithm be ALG, we have $ALG = \frac{1}{2}\sum_{x\in V} E_x \geq \frac{1}{4}\sum_{x\in V} deg(x) \geq \frac{1}{2}OPT$, proving the conclusion.

Rubric (25pt): (a) 10 pt: Correctly explain why it can be run in polynomial time; (b) 15 pt: Correctly explain the reason of 2-approximation.

Problem 3

Write down the problem of finding a Min-s-t-Cut of a directed network with source s and sink t as problem as an Integer Linear Program. (15pts)

Solution:

$$\min_{\{x_u\}_{u\in V}, \{x_{(u,v)}\}_{(u,v)\in E}} \sum_{(u,v)\in E} c(u,v) \cdot x_{(u,v)}$$

$$\begin{aligned} s.t \qquad & x_{(u,v)} - x_u + x_v \geq 0, \\ & x_u \in \{0,1\}, \forall u \in V, \\ & x_{(u,v)} \in \{0,1\}, \forall (u,v) \in E, \\ & x_s = 1, x_v = 0. \end{aligned}$$

The variable x_u, indicates if the vertex u is on the side of s in the cut. That is, $x_u = 1$ if and only if u is on the side of s. Setting $x_s = 1$ and $x_t = 0$ ensures that s and t are separated. Likewise, the variable $x_{(u,v)}$ indicates if the edge (u,v) crosses the cut. The first constraint ensures that if the edge (u,v) is in the cut, then u is on the side of s and v is on the side of t. For completeness, you should argue that with the above correspondence (that is, $x_{(u,v)}$ indicates if an edge crosses the cut), every min-s-t-cut corresponds to a feasible solution and vice versa.

Rubric (20pt): 15 pt: Correctly write LP; 5 pt: Correctly explain LP

Problem 4

100 students in the "Analysis of Algorithms" class in 2023 Summer take the exams on-site. The university provided 4 classrooms for exam use. The capacity of the i-th classroom is C_i. The safety level of the i-th classroom is proportional to $(C_i - S_i)$. Specifically, the safety level of the i-th classroom is defined as $\alpha_i(C_i - S_i)$ where α_i is the area size of the classroom and S_i is the actual number of students taking the exams in this classroom. Due to the pandemic, we want to maximize the total safety level of all the classroom. Besides, to guarantee that students have a comfort environment, the number of students in a classroom should not exceed half of the capacity of each classroom. Express the problem as a linear programming problem. You DO NOT need to solve it.

Solution: Our variables are $\{S_i\}_{i=1}^4$. The linear program is defined as follows:

$$\begin{aligned} &\max_{\{S_i\}_{i=1}^4} \sum_{i=1}^4 \alpha_i (C_i - S_i), \\ s.t \quad & S_i \geq 0, \forall i \in \{1,2,3,4\}, \\ & S_i \leq \frac{1}{2} C_i, \forall i \in \{1,2,3,4\}, \\ & \sum_{i=1}^4 S_i = 100. \end{aligned}$$

Rubric (25pt): 10 pt: The objective function is correct. 5 pt: For each condition.

Ungraded Problems

Problem 1

Suppose you have a knapsack with maximum weight W and maximum volume V. We have n *dividable* objects. The i-th object has value m_i, weights w_i and takes v_i volume. Now we want to maximize the total value in this knapsack, and at the same time, we want to use up *all* the space in the knapsack. Formulate this problem as a linear programming problem. You DO NOT have to solve the resulting LP.

Solution:

$$\begin{aligned} &\max_{\{x_i\}_{i=1}^n} \sum_{i=1}^n x_i m_i, \\ s.t \quad & \sum_{i=1}^n x_i w_i \leq W, \\ & \sum_{i=1}^n x_i v_i = V, \\ & 0 \leq x_i \leq 1, \forall i \in [n]. \end{aligned}$$

Problem 2

Given a graph G and two vertex sets A and B, let $E(A, B)$ denote the set of edges with one endpoint in A and the other endpoint in B. The Max-Equal Cut problem is defined as follows:

Instance: Graph $G(V, E)$, $V = \{1, 2, \ldots, 2n\}$ **Question:** Find a partition of V into two n-vertex sets A and B maximizing the size of $E(A, B)$. Provide a factor 1/2-approximation algorithm for solving the Max-Equal Cut problem.

Solution: Start with empty sets A, B, we perform n iterations.

In iteration i, pick vertices with index $2i - 1$ and $2i$, and place one of them in A and the other in B according to which choice maximizes $E(A, B)$ in the current step. More specifically, if $|E(A \cup \{2i - 1\}, B \cup \{2i\})| \geq |E(A \cup \{2i\}, B \cup \{2i - 1\})|$, then add vertex $2i - 1$ to A and vertex $2i$ to B; otherwise, add $2i$ to A and $2i - 1 to B$. The process ends in n iterations and is a polynomial-time algorithm.

To analyze the approximation ratio, consider a particular iteration when we have cut (A, B) and we add u and v. Suppose u has N_{A_u} and N_{B_u} neighbours in A and B respectively. Also suppose v has N_{A_v} and N_{B_v} neighbours in A and B respectively. Then, adding u to A and v to B adds $N_{B_u} + N_{A_v}$ edges to the cut, whereas doing the other way round adds $N_{B_v} + N_{A_u}$ edges to the cut. Since the sum of these two options is nothing but the total number of edges being added to this partial subgraph, the larger number of the two must be at least half the total number of edges being added to this partial subgraph. Since this is true for each iteration, at the end when all the nodes and the corresponding edges are added, our algorithm results in a cut with at least half of the total $|E|$ edges. Since the max-equal cut capacity $OPT \leq |E|$, our solution is 1/2-approximation.