Homework 12

Graded Problems

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.

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

   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 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, the DFS would have left u to explore a new vertex, thereby making u a non leaf. Hence our assumption is incorrect that both u and v are in L. Therefore N does indeed cover every edge in E.

   (b) 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 G.(15pts)

   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 e_u . We call the set of odd level non leaf vertices in T, ODD and the set of even level non leaf vertices EVEN. If the set ODD is bigger 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 $|N|/2$. Now the edge set M= $\{uEe|u\ BIG\}$ is a matching 2 of size at least $|N|/2$. Since M is a matching, to cover every edge in the matching the optimal vertex cover A has to contain at least |M| vertices. (This 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: Correctly explain the reason of 2-approximation.

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.

   i Start with an arbitrary cut C.

   ii 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.

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

   This algorithm is very similar to the one explained in Section 12.4, page 677 in the Text by using single-flip neighborhood. Here instead of checking the weight of edges we check the number of edges passing the cut to assign each vertex to one of the groups of vertices in A or in B and select the Max-Cut. Even though the single-flip neighborhood is only pseudo-polynomial but still this approximation algorithm can be run in polynomial time. The change to the algorithm to make it to be run in polynomial time is to stop it when there is no big enough improvements.

b 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.

Similar to Analyzing the Algorithm in page 677.

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.

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)

$$\textbf{minimize} \sum_{(u,v)\in E} c(u,v).x_{(u,v)}$$

$$\begin{aligned}
\textbf{subject to}: \quad x_v - x_u + x_{(u,v)} &\geq 0 && \forall (u,v) \in E && (1)\\
x_u &\in \{0,1\} && \forall u \in V : u \neq s, u \neq t && (2)\\
x_{(u,v)} &\in \{0,1\} && \forall (u,v) \in E && (3)\\
x_s &= 1 && && (4)\\
x_t &= 0 && && (5)
\end{aligned}$$

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, $x(u, v)$ indicating 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

4. 750 students in the "Analysis of Algorithms" class in 2022 Spring take the exams onsite. The university provided 9 classrooms for exam use, each classroom can contain C_i (capacity) students. The safety level of a classroom is proportional to $\alpha_i(C_i - S_i)$, where α_i is the area size of the classroom and S_i is the actual number of students taking the exams in the classroom. Due to the pandemic, we want to maximize the total safety level of all the classroom. Besides, to guarantee 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. Our objective function is:

$$maximize \sum_{i=1}^{9} \alpha_i (C_i - S_i)$$

subject to

$$S_i \geq 0, for i = 1, ..., 9$$

$$S_i \leq \frac{1}{2} C_i, for i = 1, ..., 9$$

$$\sum_{i=1}^{9} S_i = 750$$

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. Each object i 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

Let's denote the quantity of each object by a_i. Our objective function is:

$$maximize \sum_{i=1}^{n} a_i m_i$$

subject to

$$\sum_{i=1}^{n} a_i w_i \leq W$$

$$0 \leq a_i \leq 1, for i = 1, ..., n$$

$$\sum_{i=1}^{n} a_i v_i = V$$

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 one endpoint in B. The **Max Equal Cut** problem is defined as follows

Instance Graph $G(V, E)$, $V = \{1, 2, ..., 2n\}$.

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, and perform n iterations:

In iteration i, pick vertices $2i-1$ and $2i$, and place one of them in A and the other in B, according to which choice maximizes $|E(A,B)|$.

(i.e., if $|E(A \cup \{2i-1\}, B \cup \{2i\})| \geq |E(A \cup \{2i\}, B \cup \{2i-1\})|$ then add $2i$ - 1 to A and $2i$ to B, else add $2i$ to A and $2i$ - 1 to B.)

In a particular iteration, when we have cut (A, B) and we add u and v. Suppose u has N_{Au}, N_{Bu} neighbours in A, B respectively and suppose v has N_{Av}, N_{Bv} neighbours in A, B respectively. Then, adding u to A and v to B adds $N_{Bu} + N_{Av}$ edges to the cut, whereas doing the other way round adds $N_{Bv} + N_{Au}$ 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 bigger 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 edges) are added, our algorithm is bound to add at least half of the total $\|E\|$ edges. Naturally since the max equal cut capacity OPT $\leq \|E\|$, our solution is 1/2-approximation.