Homework 12

Due: Sunday, Apr 21, 11.59 PM PST

Q1)

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: Given an undirected graph G(V, E), where V has an even number of vertices, find an equal partition of V into two 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 and prove the approximation ratio for the algorithm. (15 points)

Hint: Iteratively build A and B. At each step consider a pair of vertices, and put one in each of A and B to ensure they are equal. Decide which of the two vertices goes to which side to greedily maximize E(A, B) in that step.

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. Additionally, If uv itself is an edge, it will be in the cut in either of the options, so the ratio of half is still guaranteed. 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.

Q2)

650 students in the “Analysis of Algorithms” class in 2024 Spring take the exams onsite. The university provided 7 classrooms for exam use, each classroom i can contain C_i (capacity) students. The safety level of a classroom is defined as $\alpha_i(C_i - S_i)$, where α_i is the known parameter for classroom i, and S_i is the actual number of students placed to take the exams in the classroom. We want to maximize the total safety level of all the classrooms. Besides, to guarantee good spacing, the number of students in a classroom should not exceed half of the capacity of each classroom.

Express the problem as a integer linear programming problem to obtain the number of students to be placed in each room. You DO NOT need to numerically solve the program. (15 points)

a) Define your variables.
b) Write down the objective function.
c) Write the constraints.

Solution

Our variables are S_i which denotes the number of students in Room i. Our objective function is:

Max sum_i $\alpha_i(C_i - S_i)$

The constraints are:

S_i in Z^+, for i = 1,...,7

$S_i \leq \frac{1}{2} C_i$, for i= 1,...,7

$\sum_i S_i = 650$

Rubrics

a) Define the variables. (4pts)
    i) -1 pt: minor mistake
    ii) -2 pts: redundant variable such as x[i,j] in {0,1} which denotes student i takes course j
    iii) -4 pts: wrong answer
b) Write down the objective function. (5pts)
    i) -2 pts: minor mistake
    ii) -5 pts: wrong answer
c) Write down the constraints. (6pts)
    i) -2 pts for each constraint

Q3)

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. Assume that edge capacities are positive. (20 pts)

a) Define your variables.
b) Write down the objective function.
c) Write the constraints.

Solution:

Let x_u denote if u is on the s-side or t-side of the partition (if it is 1 or 0 respectively). Let $x_{(u,v)}$ denote if edge uv is to be counted in the cut.

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

Subject to:

$x_v - x_u + x_{(u,v)} \geq 0 \quad \forall (u,v) \in E$

$x_u \in \{0,1\} \quad \forall u \in V : u \mathrel{!}= s, u \mathrel{!}= t$

$x_{(u,v)} \in \{0, 1\} \quad \forall (u, v) \in E$

$x_s = 1, x_t = 0$

Explanation: The first constraint ensures that when $x_{(u,v)}$ <u>must be 1</u> (edge uv must get counted in the cut) when x_v is 0 (i.e. v on the t-side) and x_u is 1, i.e. u is on the s-side. Since the goal of the LP is to minimize the total weight, it would try to set $x_{(u,v)}$ to 0 whenever possible, so this constraint ensures that for certain edges, the corresponding $x_{(u,v)}$ need to be set to 1. Note that this constraint does not directly prevent an edge from being included if u and v are on the same side or if u is on t-side and v is on s-side, however, they will automatically be excluded from the solution to minimize the total weight.

The next two constraints simply make the variables binary. The final two constraints fix the values of x_s and x_t since they can't be freely assigned to anything, as per the variable definition. (Otherwise, the LP-solver would try to put all the variables on the same side so that no edges need to be included, but by forcing s and t to be on opposite sides, we ensure that we get a valid cut as the solution).

Ungraded problems

Q4)

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 (volume) in the knapsack. Formulate this problem as a linear programming problem. You DO NOT have to solve the resulting LP.

Solution:

Note that the objects are dividable.

Variables : Let xi denote the fraction of item i that should be chosen in the optimal solution.

Objective: $\sum_{i = 1 \text{ to } n} m_i x_i$

Constraints:

$\sum_{i = 1 \text{ to } n} w_i x_i \leq W$ … total weight at most W

$\sum_{i = 1 \text{ to } n} v_i x_i = V$ … total volume EXACTLY V

$0 \leq x_i \leq 1$ for all I = 1 to n

Q5)

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.

Prove that the algorithm is a ½-approximation, that is the number of edges crossing C_{max} is at most twice the number crossing C.

Solution:

For any node v, suppose Sv and Dv denote #neighbors of v that are on the SAME side of partition as v, and DIFFERENT side (w.r.t. cut C) at some point. Then, out of all edges incident on v, Dv of them cross the cut and Sv do not. If we were to move v to the opposite side, all of the Dv edges are now within a partition, and all of Sv edges now cross the cut.

Now, the while loop of the algorithm ends when no vertex v can be moved to increase the #cut-edges. Thus, at this point, for each v, Dv ≥ Sv. Since the total #edges incident on v is simply Dv + Sv, it means half of the #edges incident on v cross the cut. Since this holds for every v, summing over all nodes yields that half of |E| edges are in the cut. Since the optimal value can obviously be at most |E|, we have the ½ -approximation bound as needed.