CSCI 570 - Fall 2021 - HW 12

Due November

Graded Problems

Problem 1

800 students in the “Analysis of Algorithms” class in 2021 Fall 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

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

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

$$\sum_{i=1}^{9} S_i = 800$$

Problem 2

A triangle is a set of three vertices, every two of which is connected by an edge. Consider the following minimization problem that we refer to as triangle removal. The input is a graph $G(V, E)$. The goal is to select a minimum subset of edges whose removal from the graph gives a new graph with no triangles.

Design a strongly polynomial time algorithm that provides a factor 3 approximation for triangle removal.

Solution

We design a greedy algorithm for reaching a 3-approximation. Start with the graph G. In each step find a triangle (this can be done by checking each (u, v) edge, and checking all other nodes, e.g., x to find out if $(u, x) \in E(x, v) \in E$). Add all edges (u, v), (v, x), and (x, u) to the solution set F (and remove them from E), and continue until G contains no more triangles.

Time Complexity Finding a triangle require $O(mn)$, the removing process is performed at most m times. So the overall time complexity is $O(m^2 n)$

3-approximation Consider the maximum number of edge disjoint triangles to be $\alpha(G)$ and the optimal answer of the question to be $\beta(G)$. We know that $\alpha(G) \leq \beta(G)$ because at least of edge from each disjoint triangle should be removed.
Note that by design of our algorithm, the set of triangles found in subsequent iterations are all edge disjoint. If there's κ of them, this must be $\leq$ the maximum, i.e. $\alpha(G)$. The set of edge disjoint triangles that we have found at the end of this process (κ) is a lower bound on the size of maximum set of disjoint triangles. Therefore $\kappa \leq \alpha(g) \leq \beta(G)$. (We have proved $\alpha(g) \leq \beta(G)$ in the previous paragraph.) Our solution set F has size $|F| = 3\kappa \leq 3\alpha(G) \leq 3\beta(G)$. Therefore, this greedy procedure is a 3-approximation algorithm.

Problem 3

A company makes three products and has 4 available manufacturing plants. The production time (in minutes) per unit produced varies from plant to plant as shown below:

		Manufacturing Plant			
		1	2	3	4
Product	1	5	7	4	10
	2	6	12	8	15
	3	13	14	9	17

Similarly the profit (\$) contribution per unit varies from plant to plant as below:

		Manufacturing Plant			
		1	2	3	4
Product	1	10	8	6	9
	2	18	20	15	17
	3	15	16	13	17

If, one week, there are 35 working hours available at each manufacturing plant how much of each product should be produced given that we need at least 100 units of product 1, 150 units of product 2 and 100 units of product 3. Formulate this problem as a linear program. You do not have to solve the resulting LP.

Solution

Variables At first sight we are trying to decide how much of each product to make. However on closer inspection it is clear that we need to decide how much of each product to make at each plant. Hence let x_{ij} = amount of product i ($i = 1, 2, 3$) made at plant j ($j = 1, 2, 3, 4$) per week. Although (strictly) all the x_{ij} variables should be integer they are likely to be quite large and so we let them take fractional values and ignore any fractional parts in the numerical solution. Note too that the question explicitly asks us to formulate the problem as an LP rather than as an IP.

Constraints We first formulate each constraint in words and then in a mathematical way. Limit on the number of minutes available each week for each workstation

$$5x_{11} + 6x_{21} + 13x_{31} \leq 35(60)$$
$$7x_{12} + 12x_{22} + 14x_{32} \leq 35(60)$$
$$4x_{13} + 8x_{23} + 9x_{33} \leq 35(60)$$
$$10x_{14} + 15x_{24} + 17x_{34} \leq 35(60)$$

Lower limit on the total amount of each product produced

$$x_{11} + x_{12} + x_{13} + x_{14} \geq 100$$
$$x_{21} + x_{22} + x_{23} + x_{24} \geq 150$$
$$x_{31} + x_{32} + x_{33} + x_{34} \geq 100$$

All variables are greater than equal to zero.

Objective Maximize total profit. Maximize

$$10x_{11}+8x_{12}+6x_{13}+9x_{14}+18x_{21}+20x_{22}+15x_{23}+17x_{24}+15x_{31}+16x_{32}+13x_{33}+17x_{34}$$

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.