CS 170 HW 8

Due on 2018-10-21, at 9:59 pm

1 (★) Study Group

List the names and SIDs of the members in your study group.

2 (★★) Jeweler

You are a jeweler who sells necklaces and rings. Each necklace takes 4 ounces of gold and 2 diamonds to produce, each ring takes 1 ounce of gold and 3 diamonds to produce. You have 80 ounces of gold and 90 diamonds. You make a profit of 60 dollars per necklace you sell and 30 dollars per ring you sell, and want to figure out how many necklaces and rings to produce to maximize your profits.

(a) Formulate this problem as a linear programming problem and find the solution (state the cost-function, linear constraints, and all vertices except for the origin).

(b) Suppose instead that the profit per necklace is C dollars and the profit per ring remains at 30 dollars. For each vertex you listed in the previous part, give the range of C values for which that vertex is the optimal solution.

Solution:

(a) x = number of necklaces
y = number of engagement rings

Maximize: $60x + 30y$

Linear Constraints:
$4x + y \leq 80$
$2x + 3y \leq 90$
$x \geq 0$
$y \geq 0$

Drawing a picture in 2 dimensions, we see that the vertices are $(x = 20, y = 0), (x = 15, y = 20), (x = 0, y = 30)$, and the objective is maximized at $(x = 15, y = 20)$, where $60x + 30y = 1500$.

(b) There are lots of ways to solve this part. The most straightforward is to write and solve a system of inequalities checking when the objective of one vertex is at least as large as the objective of the other vertices. For example, for $(x = 15, y = 20)$ the system of inequalities would be $C * 15 + 30 * 20 \geq C * 20, C * 15 + 30 * 20 \geq 30 * 30$. Doing this for each vertex gives the following solution:

$(x = 0, y = 30) : C \leq 20$
$(x = 15, y = 20) : 20 \leq C \leq 120$
$(x = 20, y = 0) : 120 \leq C$

One should note that there is a nice geometric interpretation for this solution: Looking at the graph of the feasible region, as C increases, the vector $(C, 30)$ starts pointing closer to the x-axis. The objective says to find the point furthest in the direction of this vector, so the optimal solution also moves closer to the x-axis as C increases. When $C = 20$ or $C = 120$, the vector $(C, 30)$ is perpendicular to one of the constraints, and there are multiple optimal solutions all lying on that constraint, which are all equally far in the direction $(C, 30)$.

3 (★★★★) Modeling: Tricks of the Trade

One of the most important problems in the field of *statistics* is the *linear regression problem.* Roughly speaking, this problem involves fitting a straight line to statistical data represented by points – $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ – on a graph. Denoting the line by $y = a + bx$, the objective is to choose the constants a and b to provide the “best” fit according to some criterion. The criterion usually used is the *method of least squares*, but there are other interesting criteria where linear programming can be used to solve for the optimal values of a and b. For each of the following criteria, formulate the linear programming model for this problem:

1. Minimize the sum of the absolute deviations of the data from the line; that is,

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

   (*Hint:* Define a new variable $z_i = y_i - (a + bx_i)$. Notice that z_i can be either positive or negative. Any number, positive or negative, however, can be represented as the difference of two non-negative numbers. Also define as non-negative variables z_i^+ and z_i^- such that $z_i = z_i^+ - z_i^-$. How can we minimize $|z_i|$ by either minimizing or maximizing some function of z_i^+ and z_i^-?)

2. Minimize the maximum absolute deviation of the data from the line; that is,

$$\min \max_{i=1\ldots n} |y_i - (a + bx_i)|$$

   (*Hint:* You'll need to start by using the same trick as above. Then consider how we can turn our objective function into a single minimization or maximization.)

Solution:

(i) Given n data points (x_i, y_i) for $i = 1, 2, \ldots n$, as well as variables a and b, define new variables $z_i = y_i - (a + bx_i)$ to denote the deviation of the i^{th} data point from the line $y = a + bx$. We know that any number, positive or negative, can be represented by the difference of two positive numbers, so that $z_i = z_i^+ - z_i^-$ and $z_i^+ \geq 0$ and $z_i^- \geq 0$ (so we are also introducing new variables z_i^+ and z_i^-).

Observe that minimizing $|z_i|$ (which is non-linear) is equivalent to minimizing $z_i^+ + z_i^-$ (which is linear) subject to the above three constraints.

Why is this the case? Let's try a small example. We could represent the number 4 as 4 - 0, 5 - 1, 6 - 2, 100 - 96, etc. But notice that the pair that gives us the smallest sum is 4 and 0. And in general for z_i, the z_i^+, z_i^- pair that gives us the smallest sum will always be either $z_i^+ = z_i$ and $z_i^- = 0$ for positive z_i, or $z_i^+ = 0$ and $z_i^- = z_i$ for negative z_i.

Hence

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

is equivalent to

$$\text{Minimize } \sum_{i=1}^{n} z_i^+ + z_i^-$$
$$\text{subject to} \begin{cases} y_i - (a + bx_i) = z_i^+ - z_i^- & \text{for } 1 \leq i \leq n \\ z_i^+ \geq 0 & \text{for } 1 \leq i \leq n \\ z_i^- \geq 0 & \text{for } 1 \leq i \leq n \end{cases}.$$

(ii) By the same reasoning, we introduce variables $z_i^+ - z_i^- = y_i - (a + bx_i)$, for $z_i^+ \geq 0$ and $z_i^- \geq 0$. Observe that minimizing $|z_i|$ (which is non-linear) is equivalent to minimizing $\max\{z_i^+, z_i^-\}$ (still non-linear) subject to the above constraints, which is equivalent to minimizing t such that $t \geq z_i^+$ and $t \geq z_i^-$ (which is linear). In simpler words, the smallest possible upper bound on z_i^+ and z_i^- will always be precisely equal to the larger of the two. Hence

$$\min \max_{i=1\ldots n} |y_i - (a + bx_i)|$$

is equivalent to

$$\text{Minimize } t$$
$$\text{subject to} \begin{cases} y_i - (a + bx_i) = z_i^+ - z_i^- & \text{for } 1 \leq i \leq n \\ z_i^+ \geq 0 & \text{for } 1 \leq i \leq n \\ z_i^- \geq 0 & \text{for } 1 \leq i \leq n \\ z_i^+ \leq t & \text{for } 1 \leq i \leq n \\ z_i^- \leq t & \text{for } 1 \leq i \leq n \end{cases}.$$

Remark: The following is an alternative solution to (i):

$$\text{Minimize } \sum_{i=1}^{n} t_i$$
$$\text{subject to} \begin{cases} y_i - (a + bx_i) \leq t_i & \text{for } 1 \leq i \leq n \\ y_i - (a + bx_i) \geq -t_i & \text{for } 1 \leq i \leq n \end{cases}.$$

The following is an alternative solution to (ii):

$$\text{Minimize } t$$
$$\text{subject to} \begin{cases} y_i - (a + bx_i) \leq t & \text{for } 1 \leq i \leq n \\ y_i - (a + bx_i) \geq -t & \text{for } 1 \leq i \leq n \end{cases}.$$

4 (★★) Repairing a Flow

In a particular network $G = (V, E)$ whose edges have integer capacities c_e, we have already found a maximum flow f from node s to node t where f_e is an integer for every edge. However, we now find out that one of the capacity values we used was wrong: for edge (u, v) we used c_{uv} whereas it should have been $c_{uv} - 1$. This is unfortunate because the flow f uses that particular edge at full capacity: $f_{uv} = c_{uv}$. We could redo the flow computation from scratch, but there's a faster way.

Describe an algorithm to repair the max-flow in $O(|V| + |E|)$ time. Also give a proof of correctness and runtime justification.

Solution:

Main Idea: In the residual graph with the original capacity for (u, v), use DFS to find a path from t to v and from u to s. Join these paths by adding the edge (v, u) in between them to get a path p. Update f by pushing 1 unit of flow from t to s on p (i.e. for each $(i, j) \in p$, reduce f_{ji} by 1). Finally, use DFS to see if the residual graph of the resulting graph, using the new capacity for (u, v), has an $s - t$ path. If so, push 1 unit of flow on this path.

Proof of Correctness: Consider the residual graph of G. If (u, v) is at capacity, then there is a flow of at least 1 unit going from v to t, and since f_e are integral there is a path from v to t with at least one unit of flow moving through it. So the residual graph will have a path from t to v. Similarly, there will be a path from u to s in the residual graph, so the algorithm can always find p correctly.

Note that the size of the maximum flow in the new network will be either the same or 1 less than the previous maximum flow (because changing one edge can change the capacity of the min-cut by at most 1). In the former case, after pushing flow from t to s on p, it is possible to push more flow from s to t, so there must be an $s - t$ path in the new residual graph, which the algorithm will find and push at least 1 unit of flow on because all capacities and flow values are integral. Thus the algorithm finds the new max flow correctly. In the latter case, we will already have the max flow after pushing flow backwards on p, so our algorithm is still correct.

Runtime: The runtime is $O(|V| + |E|)$ because the algorithm just calls DFS thrice.

5 (★★★) Generalized Max Flow

Consider the following generalization of the maximum flow problem.

You are given a directed network $G = (V, E)$ where edge e has capacity c_e. Instead of a single (s, t) pair, you are given multiple pairs $(s_1, t_1), ..., (s_k, t_k)$, where the s_i are sources of G and t_i are sinks of G. You are also given k (positive) demands $d_1, ..., d_k$. The goal is to find k flows $f^{(1)}, ..., f^{(k)}$ with the following properties:

(a) $f^{(i)}$ is a valid flow from s_i to t_i.

(b) For each edge e, the total flow $f_e^{(1)} + f_e^{(2)} + ... + f_e^{(k)}$ does not exceed the capacity c_e.

(c) The size of each flow $f^{(i)}$ is at least the demand d_i.

(d) The size of the *total* flow (the sum of the flows) is as large as possible.

Write a linear problem using the variables $f_e^{(i)}$ whose optimal solution is exactly the solution to this problem. For each constraint as well as the objective in your linear program briefly explain why it is correct. (Note: Since linear programs can be solved in polynomial time, this implies a polynomial-time algorithm for the problem)

Solution: We have variables $f_{(u,v)}^{(i)}$ for all $1 \leq i \leq k$ and $(u, v) \in E$. The total flow across each edge should be at most the capacity of the edge. Hence, we have the constraint

$$\forall (u,v) \in E \qquad \sum_{i=1}^{k} f_{(u,v)}^{(i)} \leq c_{(u,v)}$$

Also, flow conservation for the flow $f^{(i)}$ must hold for all vertices except s_i and t_i. This gives the set of constraints

$$\forall 1 \leq i \leq k, \forall v \in V \backslash \{s_i, t_i\} \qquad \sum_{(u,v)\in E} f_{(u,v)}^{(i)} = \sum_{(v,w)\in E} f_{(v,w)}^{(i)}$$

Finally, the for each i, $f^{(i)}$ must be greater than the corresponding demand.

$$\forall 1 \leq i \leq k \qquad \sum_{(s_i,u)\in E} f_{(s_i,u)}^{(i)} \geq d_i$$

The objective is simply to maximize the sum of all the flows.

$$\max \quad \sum_{i=1}^{k} \sum_{(s_i,u)\in E} f_{(s_i,u)}^{(i)}$$

6 (★★★★) Reductions Among Flows

Show how to reduce the following variants of Max-Flow to the regular Max-Flow problem, i.e. do the following steps for each variant: Given a directed graph G and the additional variant constraints, show how to construct a directed graph G' such that

(1) If F is a flow in G satisfying the additional constraints, there is a flow F' in G' of the same size,

(2) If F' is a flow in G', then there is a flow F in G satisfying the additional constraints with the same size.

Prove that properties (1) and (2) hold for your graph G'.

(a) **Max-Flow with Vertex Capacities:** In addition to edge capacities, every vertex $v \in G$ has a capacity c_v, and the flow must satisfy $\forall v : \sum_{u:(u,v)\in E} f_{uv} \leq c_v$.

(b) **Max-Flow with Multiple Sources:** There are multiple source nodes $s_1, \ldots, s_k$, and the goal is to maximize the total flow coming out of all of these sources.

Solution:

(a) Split every vertex v into two vertices, v_{in} and v_{out}. For each edge (u, v) with capacity c_{uv} in the original graph, create an edge (u_{out}, v_{in}) with capacity c_{uv}. Finally, if v has capacity c_v, then create an edge (v_{in}, v_{out}) with capacity c_v. If F' is a flow in this graph, then setting $F(u, v) = F'(u_{out}, v_{in})$ gives a flow in the original graph. Moreover, since the only outgoing edge from v_{in} is (v_{in}, v_{out}), and incoming flow must be equal to outgoing flow, there can be at most c_v flow passing through v. Likewise, if F is a flow in the original graph, setting $F'(u_{out}, v_{in}) = F(u, v)$, and $F'(v_{in}, v_{out}) = \sum_u F(u, v)$ gives a flow in G'. One can easily see that these flows have the same size.

(b) Create one "supersource" S with edges (S, s_i) for each s_i, and set the capacity of these edges to be infinite. Then if F is a flow in G, set $F'(S, s_i) = \sum_u F(s_i, u)$. Conversely, if F' is a flow in G', just set $F(u, v) = F'(u, v)$ for $u \neq S$, and just forget about the edges from S. One can easily see that these flows have the same size.

7 (★★) Provably Optimal

For the linear program

$$\begin{aligned} \max\ \ & x_1 - 2x_3 \\ & x_1 - x_2 \leq 1 \\ & 2x_2 - x_3 \leq 1 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

show that the solution $(x_1, x_2, x_3) = (3/2, 1/2, 0)$ is optimal **using its dual**. You do not have to solve for the optimum of the dual.

(*Hint:* Recall that any feasible solution of the dual is an upper bound on any feasible solution of the primal)

Solution: The objective value at the claimed optimum is $3/2$. By the duality theorem, this would be optimum if and only if there is a feasible solution to the dual LP with the same objective value. The dual of the given LP is

$$\begin{aligned} \min\ \ & y_1 + y_2 \\ & y_1 \geq 1 \\ -y_1 + {}& 2y_2 \geq 0 \\ & -y_2 \geq -2 \\ & y_1, y_2 \geq 0 \end{aligned} \quad \equiv \quad \begin{aligned} \min\ \ & y_1 + y_2 \\ & y_1 \geq 1 \\ & y_2 \geq \frac{y_1}{2} \\ & y_2 \leq 2 \\ & y_1, y_2 \geq 0 \end{aligned}$$

Greedily trying to make y_1, y_2 as small as possible results in finding that $y_1 = 1, y_2 = 1/2$ is a feasible dual solution, with the objective value 3/2. Thus, the claimed primal optimal is indeed an optimal solution.

8 (★★★) Bimetallism

There is a blacksmith who can produce n different alloys, where alloy i sells for p_i dollars per unit. One unit of alloy i takes g_i grams of gold and s_i grams of silver to produce. The blacksmith has a total of G grams of gold and S grams of silver to work with, and can produce as many units of each type of alloy as they want within the material constraints. The blacksmith is allowed to produce and sell a non-integer number of units of each alloy.

1. Formulate the linear program to maximize the revenue of the blacksmith, and explain your decision variables, objective function, and constraints.
2. Formulate the dual of the linear program from part (a), and explain your decision variables, objective function, and constraints. The explanations provide economic intuition behind the dual. We will only be grading the dual formulation.
   Hint: Formulate the dual first, then think about it from the perspective of the blacksmith when negotiating prices for buying G grams of gold and S grams of silver if they had already signed a contract for the prices for the output alloys p_i. Think about the breakeven point, from which the blacksmith's operations begin to become profitable for at least one alloy.

Solution:

(a) The decision variables x_i correspond to the number of units of each alloy created.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} x_i p_i && \text{(Maximize revenue)} \\ \sum_{i=1}^{n} x_i g_i &\le G && \text{(Use at most } G \text{ grams of gold)} \\ \sum_{i=1}^{n} x_i s_i &\le S && \text{(Use at most } S \text{ grams of silver)} \\ x_i \ge 0 \quad & \forall i \in [1...n] && \text{(Cannot produce negative amounts of metal)} \end{aligned}$$

(b) The decision variables y_G, y_S correspond to the prices of the means of production: gold, and silver. The dual poses the following question: if the prices of gold and silver were originally too high for the blacksmith's operations to be profitable, how low can they drop before breaking even? The solution returns the breakeven point; lower prices would finally allow the blacksmith to become profitable.

$$\begin{aligned} \min \quad & G y_G + S y_S && \text{(Minimize total cost of materials)} \\ g_i y_G + s_i y_S \ge p_i \quad & \forall i \in [1...n] && \text{(Cost for producing mixture } i \text{ is at least } p_i\text{)} \\ y_G, y_S &\ge 0 && \text{(Cannot set negative prices)} \end{aligned}$$