Timur Garifullin
Due: 11:59pm Thursday, Mar 23

Homework 7

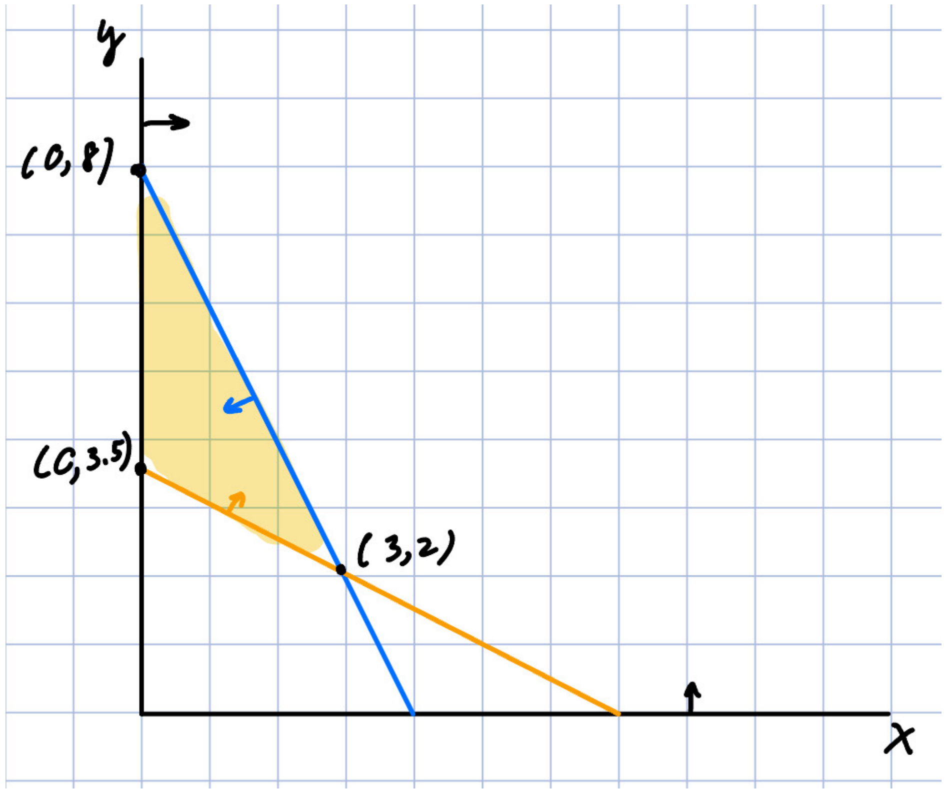


Figure 1: Inequalities Graph for part a)

This mini-assignment covers Linear Programming and Greedy Algorithms.

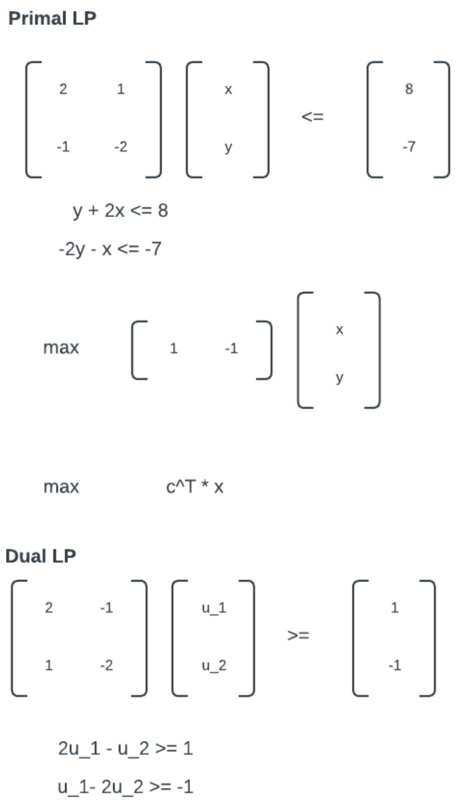
0. (0 points) (NOT OPTIONAL) Who did you work with on this homework? What sources did you consult? How long did this assignment take?

Solution: Consulting: Artur Idrissov, Caroline Guza
Source: Blackboard Notes, GeekForGeeks
Time Spent: 20 hours

1. (20 points) Here's an LP in two variables.

$$\begin{aligned} \max \quad & x - y \\ x, y &\geq 0 \\ y + 2x &\leq 8 \\ 2y + x &\geq 7 \end{aligned}$$

(a) (10 points) Sketch the inequalities of the LP below, and shade the feasible region. Find the optimum value of the objective and the value of the variables at that optimum. (Pay attention to the direction of the inequalities).



Primal LP

$$\begin{bmatrix} 2 & 1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} <= \begin{bmatrix} 8 \\ -7 \end{bmatrix}$$

y + 2x <= 8

-2y - x <= -7

$$\max \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

max c^T * x

Dual LP

$$\begin{bmatrix} 2 & -1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} u\_1 \\ u\_2 \end{bmatrix} >= \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

2u_1 - u_2 >= 1

u_1- 2u_2 >= -1

Figure 2: Different Method of Finding the Dual LP

Solution: After sketching the inequalities we get a region (shaded in gray) and three different points:
(0,8)
(0, 3.5)
(3,2)
After plugging in x and y values of the points above, we find max $x - y$ at each point
$0 - 8 = -8$
$0 - 3.5 = -3.5$
$3 - 2 = 1$
Therefore, the optimum value of the objective is $\max x - y = 1$ and the value of the variables at that optimum: $x = 3$, $y = 2$

(b) (10 points) Find the dual LP, and express it as a system of inequalities. Sketch the inequalities of the LP below, and shade the feasible region. Find the optimum value of the objective and the value of the variables at that optimum. (Pay attention to the direction of the inequalities).

Solution: Converting the original LP (Second method is shown on a separate image):

- In the original LP there are 2 variables and 2 constraints, => in dual LP there must also be 2 constraints and 2 variables
- In the original LP, the objective function is maximizing => in dual LP the objective function must be minimizing
- Since the Original function is maximizing => all "$\geq$" can be converted to "$\leq$" (by multiplying both sides by -1):
  $x + 2y \geq 7 => x + 2y \leq -7$
- In the original LP, the coefficient of the objective function $c_1 = 1, c_2 = -1 =>$ These coefficients become the right-hand side constants in dual LP.
- In the original LP, the right-hand side constants $v_1 = 8, v_2 = -7 =>$ These constants become coefficients of the objective function in dual LP

Now let's construct the new dual LP:
Let u_1, u_2 be the variables in dual LP

$$\begin{aligned} \min\ & 8u_1 - 7u_2 \\ u_1, u_2 &\geq 0 \\ 2u_1 - u_2 &\geq 1 \\ u_1 - 2u_2 &\geq -1 \end{aligned}$$

After sketching the inequalities for Dual LP we get a region (shaded in gray) and two different points:
(1,1)
(0.5, 0)
After plugging in u_1 and u_2 values of the points above, we find $\min 8u_1 - 7u_2$ at each point
$8 * 0.5 - 7 * 0 = 4$
$8 * 1 - 7 * 1 = 1$
Therefore, the optimum value of the objective is $\min 8u_1 - 7u_2 = 1$ and the value of the variables at that optimum: $u_1 = 1$, $u_2 = 1$

2. (40 points) Somebody has already written a massive LP with thousands of variables and millions of constraints, describing a factory's yearly operation, with variables for things like steel, iron, copper, worker-hours, electricity, widgets produced, gizmos produced, numbers of intermediate parts, worker accidents, and many more things, as well as many existing constraints between them. The LP is currently trying to maximize the single profit variable p. You company runs this LP to decide all sorts of things, like how many gizmos to manufacture, how many workers to hire, for which projects, etc.

   You do not need to think about this massive pre-existing LP, but to describe how to add/change

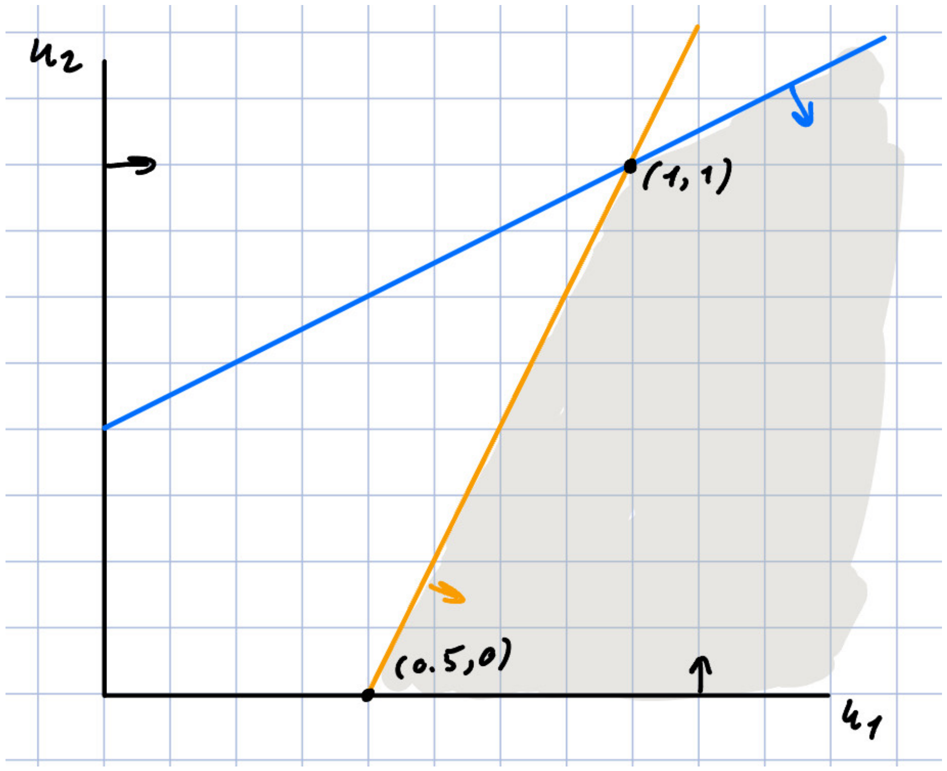


Figure 3: Inequalities Graph for part b)

this LP to achieve your goals, which are not currently a part of the LP.

You can modify the LP by saying things like “Add a variable …”, “Add the contraint…”, “Modify/change the objective by doing …”. For each edit to the LP that you make, explain what that edit is for. All of the below questions are separate questions, and remember that you do not know what the rest of the LP is - it could be anything.

(a) (EXAMPLE) Your company is going to buy special storage units for its products. Each storage unit has a form fitted space for one gizmo g, one widget w, one doohickey d and one thingamajig t, and costs u dollars. You need to buy enough storage units to store all your manufactured goods. Keep in minds that a gimzo can’t fit in a spot designed for a widget, etc., so any excess storage space can’t be reused.

How would you modify the LP to achieve this? Explain. Existing variables in the LP:

g - number of gizmos

w - number of widgets

d - number of doohickeys

t - number of thingamajigs

Solution: We did this in class. Add the variable s for the total number of storage units we buy.

We need enough storage for the items we manufacture, so add the constraints $s \geq g$, $s \geq w$, $s \geq d$, and $s \geq t$, to ensure that we buy enough storage units to fit each type of item.

Homework 7

Finally, change the objective from $maxp$ to $maxp - s * u$ because the cost of the storage units lower our profit.

Note: Because each storage unit will count against the profit, there will be no incentive for the LP to buy unnecessary storage units.

(b) (EXAMPLE) Let W be the set of all workers. Your company's workers all work h_w hours per week. These hours are allocated to tasks by other constraints in the LP. Due to new rules, they are now forbidden from working more than H hours per week, and also get a retirement bonus of c_w per hour that they work. This bonus counts against the profit p of the company.

Modify the LP to take this into account.

Existing variables in the LP:

For each worker w: h_w - number of hours worker w works.

Solution: For each worker w: Add the constraint $h_w \leq H$ because each worker can work at most H hours.

Modify the objective to be: $maxp - \sum_{winW} c_w \cdot h_w$ to subtract the retirement bonus from the profit.

(c) (10 points) Your company's reputation is based on gizmos. You must ensure that your profit from gimzos is at least the total profit from widgets, doohickeys, and thingamajigs combined. You are given the amount of profit p_g, p_w, p_d, and p_t per gizmo, widget, doohickey, and thingamajig respectively.

How would you modify the LP to achieve this? Explain.

Existing variables in the LP:

g - number of gizmos produced

w - number of widgets produced

d - number of doohickeys produced

t - number of thingamajigs produced

Solution: Add constraints:
$p_g * g \geq p_w * w + p_d * d + p_t * t$ to ensure that the profit from gizmos, $p_g * g$, is at least as much as the combined profit from widgets, doohickeys, and thingamajigs.

Objective:

Same as before, we want LP to maximize single profit p. We don't need to modify this because there are no additional costs involved with our constraints.

(d) (10 points) Due to new laws, your company needs to wrap all its gizmos, widgets, doohickeys, and thingamajigs in bubble wrap. You need to use b_g sqft wrap per gizmo, b_w sqft wrap per widget, etc. Bubble wrap is sold at c dollars per sqft, and you can only buy at most B sqft total (more than this isn't available).

How would you modify the LP to accommodate this? Explain.

Existing variables in the LP:

g - number of gizmos

w - number of widgets

d - number of doohickeys

t - number of thingamajigs

Solution:

Add constraints:

$b_g * g + b_w * w + b_d * d + b_t * t \leq B$ to ensure that the total sqft of bubble wrap used doesn't exceed the available limit.

Modify the objective from maximizing single profit p to maximizing $p - c(b_g * g + b_w * w + b_d * d + b_t * t)$, c - dollars per sqft of wrap, to subtract the cost of purchasing bubble wrap from the profit.

This will lead to LP trying to maximize the profits with the cost of wrapping each item taken into account.

(e) (10 points) You are opening a new factory, and just beginning the basics of the LP for this new factory. There are some number of products that need to be done J. Each job $j \in J$ requires h_j total hours of work to be done.

At that factory, there are workers W, and each worker $w \in W$ knows how to do work on a subset $J_w \subseteq J$ of the jobs, has up to h_w total hours available to work, and earns c_w dollars per hour as salary.

Like your exam question, workers can split up their hours across multiple jobs, and each job can have multiple workers working on it. Unlike the exam question, the goal isn't to determine if all jobs can be done, but to determine the *cheapest way* to get all jobs done.

Exisiting variables in the LP:

None

Hint: The hardest part for this problem is choosing the right variables.

Solution:

Add variables:

$w_{hj} \in W$ - number of hours worker w spent on the job j

Add constraints:

For all $J \notin J_w$ then $w_{hj} \leq 0$ and $w_{hj} \geq 0$

$$\text{For each worker } w\text{: } \sum_{j \in J} w_{hj} \leq h_w$$

- because each worker can't work longer than h_w

$$\sum_{w \in W} w_{hj} \leq h_j$$

and

$$\sum_{w \in W} w_{hj} \geq h_j$$

- to check if all the jobs are completed. If workers have spent in total h_j hours on a job, then the job is completed

Modify the objective to

$$\text{Minimize} \sum_{w \in W} \sum_{j \in J} c_w * w_{hj}$$

- to minimize the total cost by summing up the cost of each worker's contribution to each job they work on, where the cost is calculated as the product of the hourly wage rate and the hours contributed

(f) (10 points) Your company's bad PR r.e. worker injuries has forced them to put reducing worker accidents as their first priority. They must seek to limit the total number of worker accidents foremost, and then maximize profit as a second priority. (Making \$300 million with 100 accidents is now considered better than making \$1 billion with 101 accidents).

How would you do this optimization?

Existing already in the LP:

a_w - accidents per worker w

p - profit.

Hint: Run more than one LP.

Solution:
1. First LP:
Modify the objective from maximizing single profit p to

$$\text{Minimize} \sum_{w \in W} a_w$$

2. Second LP:
Add variable:
a_{min} - minimum of total possible accidents

Add the constraint:
$\sum_{w \in W} a_w \leq a_{min}$ to limit the amount of total accidents possible

Modify the objective from $\sum_{w \in W} a_w$ to maximize p

3. (40 points) A classic greedy algorithm used in everyday life is *currency representation.* If you need to represent k dollars using available currency bills b, a natural greedy algorithm is to repeatedly choose the largest available bill and add it to your current set of bills. For example, if I wanted to represent $k = 133$ using the US bill set $B = [1, 5, 10, 20, 50, 100]$, I would be left with one 100 bill, one 20 bill, one 10 bill, and three 1 bills.

(a) (10 points) Implement this greedy algorithm (provide psuedocode or code) as quickly as possible. This function should take as input a list of bills and a target value k. You may assume that the bills are given to you in already sorted order. You should return a dictionary or array with the total number of each type of currency. You will get fewer points for an asymptotically slower algorithm.

Solution:

```
def currency_v2(B, k):
  leftover = k
  out = {}

  # Iterating a set of bills in reversed order
  for r in range(len(B)-1, -1, -1):
      # Calculating the number of bills needed
      count = leftover // B[r]

      # Updating the count of each necessary bill
      if count > 0:
          out[B[r]] = count

          # Updating the leftover amount
          leftover = leftover % B[r]

  return out
```

(b) (10 points) Invent a currency system where the the greedy algorithm may not always return an optimal representation with the fewest number of bills.

Solution: Counterexample:
B = [1, 10, 15] (Ex: CS dollar)
k = 20

The greedy algorithm will first choose a 15-dollar bill, as its largest bill. Then the remaining amount will be 5, and because the next largest bill after 15 is 10 (which is bigger than 5), the greedy algorithm would choose five 1-dollar bills instead.
This will result in the greedy algorithm selecting 6 bills in total.

However, the optimal algorithm would choose two 10-dollar bills, satisfying k = 20.
This will result in the optimal algorithm selecting ONLY 2 bills in total.

(c) (20 points) Finding general criteria for currency systems to determine whether or not the greedy algorithm *is* always optimal for that currency system is mathematically complicated and way beyond this course.

However, it is not so hard to prove that the greedy algorithm is optimal for the US system specifically. Prove that the US system $[1, 5, 10, 20, 50, 100]$ is optimal using the strategy of *minimal counterexample*, outlined below, which is related to proof by induction.

Here is the outline for your proof.

1. Do a proof by contradiction
2. Let k be the *smallest* counter example target value for which greedy isn't optimal. Call the greedy solution ALG and the true optimum solution OPT.
3. Argue that greedy and optimal solutions for k must share no bills in common.

4. Let b be the smallest denomination used across all the bills in both the greedy and optimal solution. Argue by cases that none of the bills in the US system can be b. Hint: Think about greatest common divisors.

Solution:
Proof by contradiction:

- Assume that the greedy algorithm is not optimal for the US currency system.
- Let k be the smallest counter-example target value for which greedy isn't the optimal solution.
- Assume both greedy algorithm and optimum solution share the same bill y
- If I subtract y from both solutions then we get $k - y = k'$
- Since k' is less than k, therefore k' is the smaller counter-example than k
- This contradicts that k is the smallest counter-example
- Therefore greedy and optimal solution for k must share no bills in common

Case 1: $b = 1$
- The 1 bill is the smallest denomination, and by the nature of the U.S. currency system, if $b = 1$, then the greedy algorithm is already optimal since any number of 1 bills can be used to make up any amount without requiring a larger denomination for efficiency. Thus, this case inherently does not lead to a contradiction but instead affirms the optimality of the greedy algorithm for any counterexample involving 1 bills.

Case 2: $b = 5$
- If $b = 5$ and is part of OPT but not ALG (or vice versa), and given that any target value requiring a 5 bill could also be achieved by five 1 bills (which the greedy algorithm would not prefer over a single 5 bill for values $k \geq 5$), it indicates that the greedy algorithm would already use the 5 bill optimally. Any scenario where 5 is needed but not selected by the greedy algorithm contradicts the assumption of it being a counterexample for non-optimality.

Case 3: $b = 10$
- The argument follows as for 5. If the greedy solution does not select a 10 bill where it is optimal, this contradicts the premise of the greedy algorithm, which would select a 10 bill for any amount $k \geq 10$ that is a multiple of 10 before resorting to smaller denominations.

Case 4: $b = 20$
- For $b = 20$, if the optimal solution uses a 20 bill and the greedy solution does not (or cannot), considering the greedy algorithm's preference for larger denominations to minimize the number of bills, suggests a misunderstanding of how the greedy algorithm operates. The 20 bill would be chosen by the greedy algorithm for any amount $k \geq 20$ that benefits from its inclusion, reinforcing the optimality of the greedy approach for such values.

Case 5: $b = 50$
- If $b = 50$, the reasoning is the same as for previous cases. The greedy algorithm would prefer a 50 bill for any amount $k \geq 50$ where its inclusion reduces the total number of bills. Any assumption that a 50 bill is optimal but not chosen by the greedy algorithm contradicts the algorithm's fundamental approach.

Case 6: $b = 100$
- The greedy algorithm would always opt for a 100 bill when the target value $k \geq 100$, as it minimizes the total number of bills required. Any contradiction here directly conflicts with the basic principle of the greedy algorithm, highlighting a misunderstanding of its operation rather than proving its non-optimality.