Lecture 16: Linear Programming Continued

Sam McCauley

November 8, 2021

Williams College

Admin

- Mini-midterm 2 over!
- Assignments 3 and 4 graded
- Please delete your text files from the lab computer if you're not using them (can keep zip version)
- Assignment 6 out tonight. All LPs! Some where I ask to give and prove correct; some where I ask you to use GLPK to solve
- Class of 60's speaker Jon Kleinberg tonight and tomorrow
    - Very modern approach to algorithms, wrote the 256 textbook

Linear Programming

A linear program consists of:

- a linear *objective function*, and
- a set of linear *constraints*.

Goal: achieve the best possible objective function value while satisfying the constraints

Solving Problems with Linear Programming

Example 3 (hard): Group Grading

- The CS TAs at Williams have decided that all TAs will help do the grading for all assignments due in a given week.
- Each assignment is due during one of n hour-long time slots, and there are m courses total.
- Time slot i has t_i TAs available for grading
- Grading a single assignment from course j requires a total of h_j TA hours worth of time
- $w_{i,j}$ is the number of assignments from course j that arrive at time slot i
- Question: for each time slot i, how many (fractional) TAs should work on each course j to minimize the average time it takes each submission to be graded?

Example 3 (hard): Group Grading

First: it sounds like we should make a variable for the actual assignment we want. Let $x_{i,j}$ be the number of TAs working on course j in time slot i.

- Time slot i has t_i TAs available for grading
- Grading a single assignment from course j requires a total of h_j TA hours worth of time
- $w_{i,j}$ is the number of assignments from course j that arrive at time slot i
- Question: for each time slot i, how many TAs should work on each course j to minimize the average time it takes each submission to be graded?

Example 3 (hard): Group Grading

Can we constrain $x_{i,j}$?

Yep, $\sum_j x_{i,j} \leq t_i$

- Time slot i has t_i TAs available for grading
- Grading a single assignment from course j requires a total of h_j TA hours worth of time
- $w_{i,j}$ is the number of assignments from course j that arrive at time slot i
- Question: for each time slot i, how many TAs should work on each course j to minimize the average time it takes each submission to be graded?

Example 3 (hard): Group Grading

What if we can't finish all the work in a given timeslot? We need to keep track of what spills over. Let $r_{i,j}$ be the remaining work for course j after time slot i.

- Time slot i has t_i TAs available for grading
- Grading a single assignment from course j requires a total of h_j TA hours worth of time
- $w_{i,j}$ is the number of assignments from course j that arrive at time slot i
- Question: for each time slot i, how many TAs should work on each course j to minimize the average time it takes each submission to be graded?

Example 3 (hard): Group Grading

How much work is remaining? Well, during time slot i for course j, we assign $x_{i,j}$ TAs, so they can grade a total of $x_{i,j}/h_j$ assignments.

- Time slot i has t_i TAs available for grading
- Grading a single assignment from course j requires a total of h_j TA hours worth of time
- $w_{i,j}$ is the number of assignments from course j that arrive at time slot i
- Question: for each time slot i, how many TAs should work on each course j to minimize the average time it takes each submission to be graded?

Example 3 (hard): Group Grading

Time slot i starts with $r_{i-1,j}$ assignments remaining for course j. The TAs can grade $x_{i,j}/h_j$ assignments, and $w_{i,j}$ new assignments are turned in. Therefore, $r_{i,j} \geq r_{i-1,j} + w_{i,j} - x_{i,j}/h_j$.

- Time slot i has t_i TAs available for grading
- Grading a single assignment from course j requires a total of h_j TA hours worth of time
- $w_{i,j}$ is the number of assignments from course j that arrive at time slot i
- Question: for each time slot i, how many TAs should work on each course j to minimize the average time it takes each submission to be graded?

Cost?

- We want to minimize the average time it takes each submission to be graded.
- The total time all submissions of course j wait is $\sum_i r_{i,j}$
- The total number of submissions is $\sum_i \sum_j w_{i,j}$
- Need $r_{i,j} \geq 0$!
- Objective function: minimize $\left(\sum_j \sum_i r_{i,j}\right) / \left(\sum_i \sum_j w_{i,j}\right)$

Example 3: Final LP

Objective: $\min \left(\sum_j \sum_i r_{i,j}\right) / \left(\sum_j \sum_i w_{i,j}\right)$

Remember that h_j is a constant!

Constraints:

For all i: $\sum_j x_{i,j} \leq t_i$

For all $i > 0$ and all j: $r_{i,j} = r_{i-1,j} + w_{i,j} - x_{i,j}/h_j$

$r_{0,j} = w_{0,j} - x_{0,j}/h_j$

For all i and all j: $x_{i,j} \geq 0$ and $r_{i,j} \geq 0$

- What are the variables? What are the constants?
- Is this an LP? What is its size? How many dimensions?
- How can we go from a feasible LP solution to a real-world schedule?

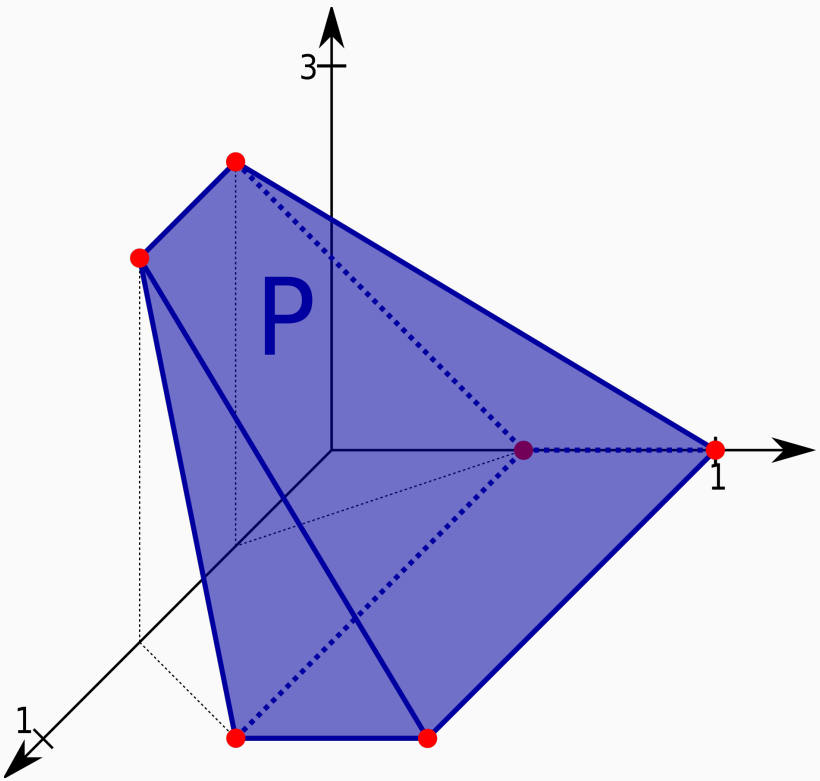
Structure of Linear Programs

Canonical Form

- Without loss of generality, can always put all constants on the right; can ensure variable appears once per line

- Some solvers need other constraints (like all $\leq$); ours doesn't

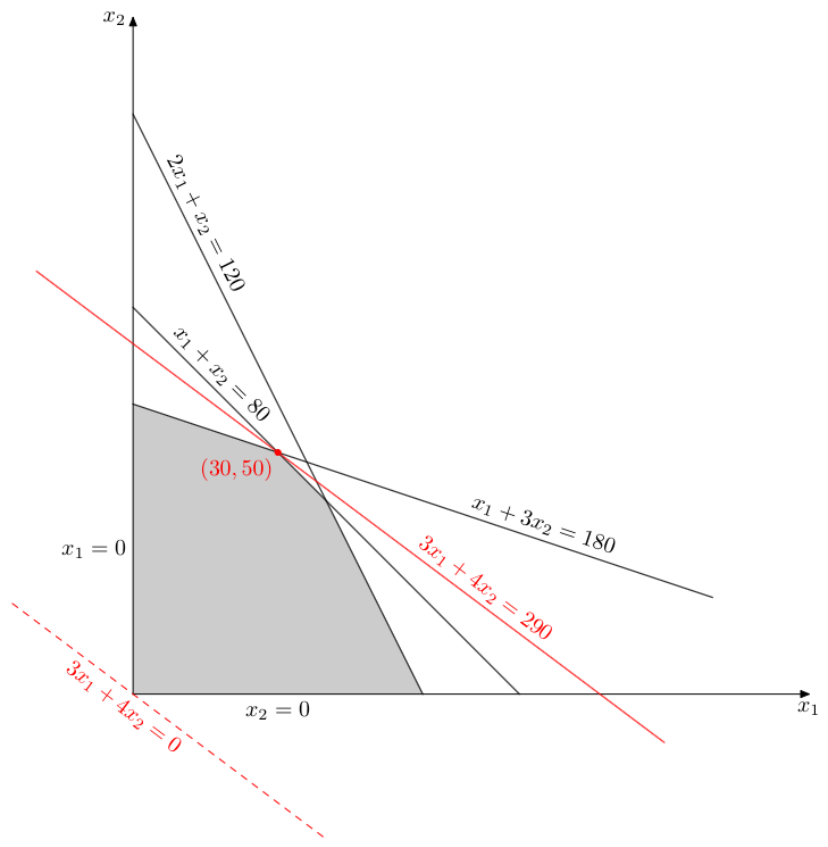
Extreme Points



- Where can a solution lie?
- Can't ever be *inside* the polytope
- In fact, don't need to look along a line either
- All solns at *extreme point*
- Defn: does not lie on a line between two other points in the polytope

Solving Linear Programs (Theory)

First Steps



- For small programs, draw them out and solve them
- This is not a bad tactic for solving these by hand

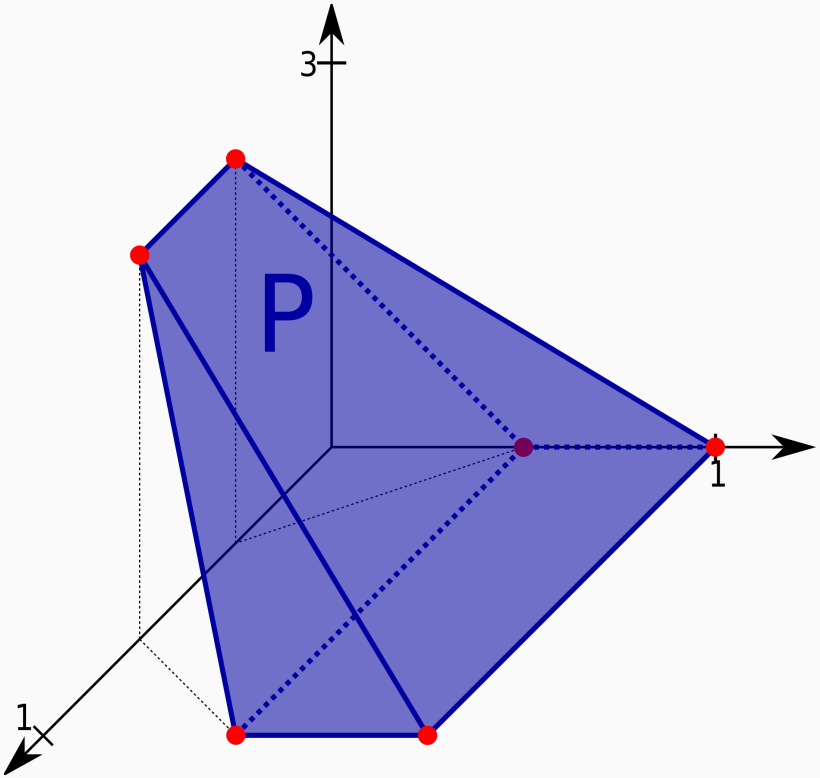
Some theory

- $O(n)$ time for constant dimensions
- Also: polynomial time algorithm in general!
  - "Ellipsoid method" (Khachiyan 1979)
  - "Interior point methods" (Karmarkar 1984)
  - Best known currently: Cohen, Lee, Song, Zhang 2019
  - "Strongly" polynomial still open

Simplex Algorithm

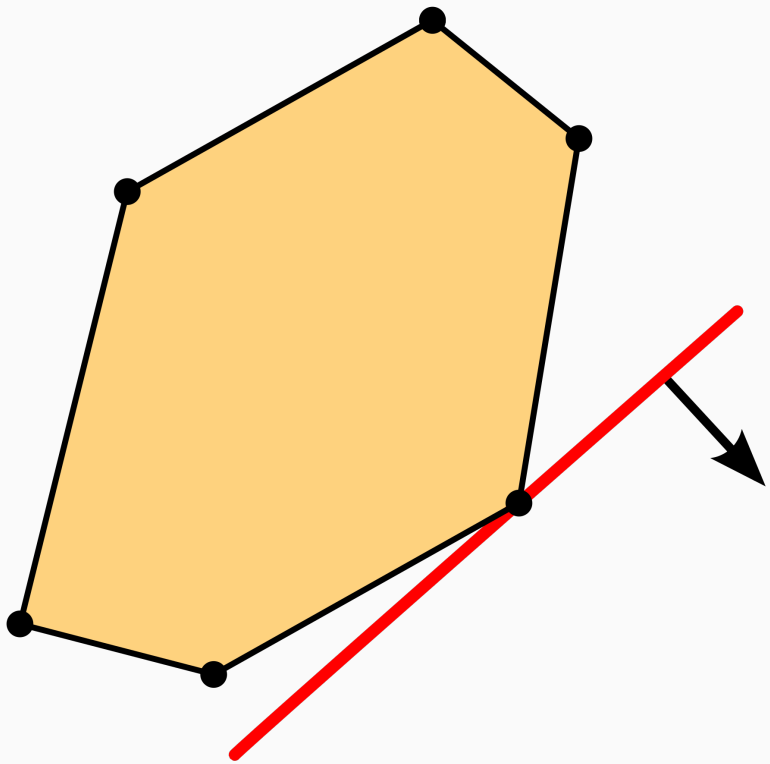
- Invented by Dantzig in 1947
- Simple, most common in practice
- Works extremely well on real-world data
- Exponential time in the worst case
- We will just see a tiny piece of this algorithm

How do we search through extreme points?



- From one extreme point, we can follow an edge to another
- Pros: local!
- Has a nice algebraic formulation
- But when do we know that we have the best solution?

Going through extreme points



- One option: keep track of which ones we've seen, stop once we've seen all of them
- This takes up lots and lots of space!
- Not very efficient
- No opportunities for heuristics:
  - even if we see the solution early, need to search through all of them

Key Lemma

Lemma 1

An extreme point is an optimal solution if every adjacent extreme point has a strictly worse objective value.

- That is to say: a local maximum is always a global maximum!
- Adjacent means connected by a line
- More formally: "adjacent" extreme points can be determined by loosening one constraint and tightening another
- Called a "pivot"

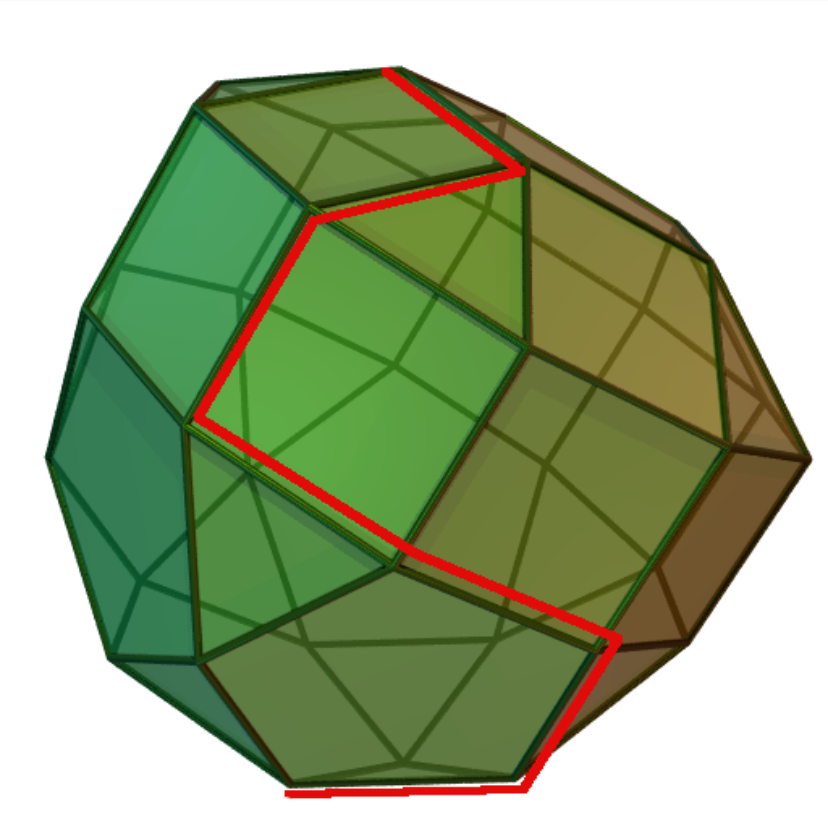
Simplex: searching through extreme points

- Start at some extreme point
- While there is an adjacent extreme point with the same or better objective function:
    - Go to that extreme point
- Return current extreme point

Does this work?

- By our lemma, if it finishes, the value it returns is correct.
- When might it not finish?
- First: need to find the initial extreme point
    - Significant area of research; usually easy in practice
- Can the algorithm loop infinitely?
    - Yes. Also significant area of research, can generally be avoided in practice.

Simplex Algorithm



- This is what simplex does:
- Greedily searches through points
- Does not keep track of previous points
- Very good at getting to the right place quickly in practice

Where to pivot?

- Simplex performance depends on what extreme point we go to next ("pivot rule")
- How can we choose?
- One option: greedily choose best objective function
    - Not bad, but not as good as you'd think
- 70 years of optimization have gotten us really effective rules
- Some work well for certain types of problems (i.e. network flows)

How fast is it?

- Classic result: there exists an LP with n variables and n constraints such that simplex can take $\Omega(2^n)$ time (Klee Minty 1972)
    - (But subexponential pivot rule by Hansen and Zwick in 2015!)
- Even if all constants are in $\{1, 2, 3, 4\}$
- Good news: bad cases are very very carefully crafted, extremely rare in practice

Using an LP Solver

LP Solver in this course

- GLPK: open source solver
- Can be called from C or as a standalone program
  - We'll be using as a standalone program
  - Arguably easier. (Downside: can't *program* the generation of the LP. Have to write it out by hand.)
- (Expensive) industrial programs may have better performance, especially for specific types of LPs

What does GLPK do

- Best effort to solve the problem (uses very optimized simplex, plus some other stuff)
- Gives you solution, tells you whether or not it's optimal.
  - Remember that simplex may know when it arrives at an optimal solution
  - (More advanced techniques can also be used)
- So far: basically solves everything I've tried instantly, optimally
- Full disclosure: I've used this program a few times but I don't know it in and out, especially corner cases

Formatting LP in this class

- We'll be using the CPLEX format
- Pretty much looks like writing the LP in text
- Note: all inequalities may be written as strict inequalities: you can write $<$ rather than $<=$. But $<=$ is always meant!

CPLEX LP format summary: objective function

- (Must) start with objective function
- write `maximize` or `minimize`
- Then just write the function! (Can name it if you want with name:)
- Example: `minimize obj:  - x1 + 2 x2 - 3.5 x3`

CPLEX LP format summary: Constraints

- Must write `subject to`
- Then, one constraint per line (again, can name)
- Must have one constant on right side of equation

```
Subject To
one:  y1 + 3 a1 - a2 - b >= 1.5
y2 + 2 a3 + 2 a4 - b >= -1.5
two :  y4 + 3 a1 + 4 a5 - b <= +1
.20y5 + 5 a2 - b = 0
```

CPLEX LP format summary: bounding variables

- Special section to give bounds on individual variables
- Useful! (and optional; variables are positive by default)
- Write `bounds` then a sequence of bounds (one per variable)
- `+inf` and `-inf` for infinity; `free` for unbounded variable

```
 Bounds
-inf <= a1 <= 100
-100 <= a2
b <= 100
x2 = +123.456
x3 free
```

CPLEX LP format summary: finishing it up

- Starting next week: another section for specifying integer variables
- Don't need that section for now!
- Then write `end` keyword

Running the LP Solver

- `glpsol --cpxlp [LP file] -o [desired output file]`
- `glpsol --cpxlp mylp.lp -o mylp.out`
- Outputs solution to output file (text format!! Despite the extension)
- Also outputs a bunch of information to the command line
- Let's look at an example!

GLPK Example Usage

Example 1: Diet

- You need to eat 46 grams of protein and 130 grams of carbs every day
- 100g Peanuts: 25.8g of protein, 16.1g carbs, $1.61
- 100g Rice: 2.5g protein, 28.7g carbs, $.79
- 100g Chicken: 13.5g protein, 0g carbs, $.70

What is the cheapest way you can hit your diet goals? First, let's formulate the LP together on the board

GLPK Example

Now let's make the file

- Start with objective function
- Then `subject to`, then constraints
- Finally, `bounds` followed by bounds
- Then `end`

Conclusion/Summary for Linear Programming

What Takeaways do I want?

- What is an LP?
- How to take a problem and phrase it as a linear program?
- Use GLPK to solve problems
- Basics of: how does the simplex algorithm work?

Brear in mind: we can solve some neat problems now. Next week we'll get to MIPs where the real magic lies.

More LP Practice

Another practice problem

- Very LP-y
- We'll prove correctness (fairly boring in this case—practice for more interesting LPs)
- And we'll run it through the solver to get the optimal solution

Allocating farm resources

Walnut Orchard has two farms that grow wheat and corn. Because of differing soil conditions, there are differences in the yields and costs of growing crops on the two farms. The yields and costs are

	Farm 1	Farm 2
Corn yield/acre	500 bushels	650 bushels
Cost/acre of corn	\$100	\$120
Wheat yield/acre	400 bushels	350 bushels
Cost/acre of wheat	\$90	\$80

Each farm has 100 acres available for cultivation; 11,000 bushels of corn and 7000 bushels of wheat must be grown. Goal: minimize cost.

Walnut Orchard Solution

Hopefully we came up with something like:

- c_1, c_2, w_1, w_2 are number of acres for corn and wheat on farm 1 and 2 respectively
- Objective: minimize $100c_1 + 120c_2 + 90w_1 + 80w_2$
- Constraints:
  - $c_1 + w_1 \leq 100 \quad c_2 + w_2 \leq 100$
  - $500c_1 + 650c_2 \geq 11000$
  - $400w_1 + 350w_2 \geq 7000$
  - $c_1, c_2, w_1, w_2 \geq 0$

Proving Correctness

As usual, we'll split into two claims.

- Let's say there's an LP solution c_1, c_2, w_1, and w_2 with objective value C. Then there exists a way for Walnut Orchard to allocate its resources with total cost C.

- Let's say there exists a way for Walnut Orchard to grow wheat and corn that has total cost C. Then there exists an LP solution with objective value C.

First claim

Let's say there's an LP solution c_1, c_2, w_1, and w_2 with objective value C. Then there exists a way for Walnut Orchard to allocate its resources with total cost C.

Proof: Let's say that Walnut Orchard grows c_1 acres of corn on farm 1 (resp. c_2, w_1, w_2). Let's verify that all the constraints are satisfied:

- Each farm has 100 acres available for cultivation

The total number of acres we use for farm 1 is $c_1 + w_1$. Since $c_1 + w_1 \leq 100$ this constraint is met for farm 1.

The total number of acres we use for farm 2 is $c_2 + w_2$. Since $c_2 + w_2 \leq 100$ this constraint is met for farm 2.

First claim

Let's say there's an LP solution c_1, c_2, w_1, and w_2 with objective value C. Then there exists a way for Walnut Orchard to allocate its resources with total cost C.

Proof: Let's say that Walnut Orchard grows c_1 acres of corn on farm 1 (resp. c_2, w_1, w_2). Let's verify that all the constraints are satisfied:

- 7000 bushels of wheat must be grown

We grow w_1 acres of wheat on farm 1, leading to $400w_1$ bushels.
We grow w_2 acres of wheat on farm 2, leading to $350w_2$ bushels.

Since we required $400w_1 + 350w_2 \geq 7000$ the constraint is satisfied.

First claim

Let's say there's an LP solution c_1, c_2, w_1, and w_2 with objective value C. Then there exists a way for Walnut Orchard to allocate its resources with total cost C.

Proof: Let's say that Walnut Orchard grows c_1 acres of corn on farm 1 (resp. c_2, w_1, w_2). Let's verify that all the constraints are satisfied:

- 11000 bushels of corn must be grown

We grow c_1 acres of corn on farm 1, leading to $500c_1$ bushels. We grow c_2 acres of corn on farm 2, leading to $650c_2$ bushels.

Since we required $500c_1 + 650c_2 \geq 11000$ the constraint is satisfied.

First claim

Let's say there's an LP solution c_1, c_2, w_1, and w_2 with objective value C. Then there exists a way for Walnut Orchard to allocate its resources with total cost C.

Proof: Let's say that Walnut Orchard grows c_1 acres of corn on farm 1 (resp. c_2, w_1, w_2). Let's check the cost:

Cost for Walnut Orchard is $100c_1 + 120c_2 + 90w_1 + 80w_2$. This is exactly equal to C.

Second claim

Let's say there exists a way for Walnut Orchard to grow wheat and corn that has total cost C. Then there exists an LP solution with objective value C.

Proof: Define c_1 to be the number of acres of corn grown by Walnut Orchards on farm 1 while achieving cost C (resp. c_2, w_1, w_2). Let's check the cost:

Cost for Walnut Orchard is $100c_1 + 120c_2 + 90w_1 + 80w_2$. This is exactly equal to the objective value of the LP.

Second claim

Let's say there exists a way for Walnut Orchard to grow wheat and corn that has total cost C. Then there exists an LP solution with objective value C.

Proof: Define c_1 to be the number of acres of corn grown by Walnut Orchards on farm 1 while achieving cost C (resp. c_2, w_1, w_2). Let's make sure all constraints are met:

- Must have $c_1 + w_1 \leq 100$ and $c_2 + w_2 \leq 100$.

Each farm has 100 acres available for cultivation. By definition, the total number of acres we use for farm 1 is $c_1 + w_1$; this must be at most 100. The same argument works for farm 2.

Second claim

Let's say there exists a way for Walnut Orchard to grow wheat and corn that has total cost C. Then there exists an LP solution with objective value C.

Proof: Define c_1 to be the number of acres of corn grown by Walnut Orchards on farm 1 while achieving cost C (resp. c_2, w_1, w_2). Let's make sure all constraints are met:

- Must have $400w_1 + 350w_2 \geq 7000$.

We grow w_1 acres of wheat on farm 1, leading to $400w_1$ bushels. We grow w_2 acres of wheat on farm 2, leading to $350w_2$ bushels. 7000 bushels of wheat must be grown in total on the farm, so the equation must be satisfied.

Second claim

Let's say there exists a way for Walnut Orchard to grow wheat and corn that has total cost C. Then there exists an LP solution with objective value C.

Proof: Define c_1 to be the number of acres of corn grown by Walnut Orchards on farm 1 while achieving cost C (resp. c_2, w_1, w_2). Let's make sure all constraints are met:

- Must have $500c_1 + 650c_2 \geq 11000$.

We grow c_1 acres of corn on farm 1, leading to $500c_1$ bushels. We grow c_2 acres of corn on farm 2, leading to $650c_2$ bushels. 11000 bushels of wheat must be grown in total on the farm, so the equation must be satisfied.

Let's solve the LP using our solver!

One more LP

- Let's solve a difficult optimization problem with our LP solver
- Idea: a middle school closed. Students from 6 different areas need to be assigned to the three other existing middle schools in the area. How can we do that?
- Need to assign all students; make sure students are reasonably well-balanced; minimized cost of transportation
- One problem: will wind up with fractional number of students assigned. How can we resolve this?
  - Round!
  - Does make our solution not optimal. But we'll discuss: by how much?

Setting up school problem

- Three schools with capacities 900, 1100, 1000
- Grades 6, 7, and 8
- Each grade assigned to a school must consist of between 30% and 36% of the school's total assignment. (Can't give one school all eighth graders.)
- Let's look at numbers in terms of what students are from each area, and how much it costs to get students from an area to a school.

School Problem Numbers

Area	6th	7th	8th	School 1	School 2	School 3
1	144	171	135	$300	0	$700
2	222	168	210	-	$400	$500
3	165	176	209	$600	$300	$200
4	98	140	112	$200	$500	-
5	195	170	135	0	-	$400
6	153	126	171	$500	$300	0

- Three schools with capacities 900, 1100, 1000
- Each grade assigned to a school must consist of between 30% and 36% of the school's total assignment. (Can't give one school all eighth graders.)
- minimize total cost

Our solution is fractional!

- What can we do?
- Round up or down; make sure constraints are still met
- How much can this affect our cost?
- Each school will probably end up with ≈ 1 student away from optimal. Unlikely to be more than \$1000 or so off.
- When is this strategy not a good idea?
    - When rounding changes the solution by a larger amount