Extra Lecture - Linear Programming

Eric A. Autry

Course Announcements

Project 4 due Thursday.

Check grades on Sakai.

Practice Problems to be posted tonight.

- Solutions by Wednesday night.

Updated notes by Thursday.

Please take DukeHub Survey.

Profit Maximization

Say we have a company that makes two products.

- Gadgets: 300 demanded per day at a profit of \$1 each.
- Gears: 200 demanded per day at a profit of \$6 each.

Now, if the company could produce all 500 demanded gadgets and gears each day, it would make a profit of

$$300 \times \$1 + 200 \times \$6 = \$1500.$$

However, the company is only able to make a total of 400 products per day.

Question: how many gadgets and gears should the company produce each day to maximize profit?

Profit Maximization

We can encode this problem as a maximization problem with linear constraints:

Define variables x_1 and x_2 to be the number of gadgets and gears produced, respectively.

The goal is to maximize the **objective function**, the profit:

$$x_1 + 6x_2,$$

subject to the **constraints**:

$$\begin{aligned} x_1 &\leq 200, \\ x_2 &\leq 300, \\ x_1 + x_2 &\leq 400, \\ x_1, x_2 &\geq 0. \end{aligned}$$

Note: the objective function and constraints are linear in the variables.

Profit Maximization

The goal is to maximize the **objective function**, the profit:

$$x_1 + 6x_2,$$

subject to the **constraints**:

$$\begin{aligned} x_1 &\leq 200, \\ x_2 &\leq 300, \\ x_1 + x_2 &\leq 400, \\ x_1, x_2 &\geq 0. \end{aligned}$$

We can plot these constraints (each is a line in the x_1-x_2 plane).

The **feasible** solutions lie in a region bounded by these lines, which will be a convex polygon.

Meanwhile, contours of the objective function correspond to lines of fixed cost solutions:

$$x_1 + 6x_2 = c.$$

Profit Maximization

When we look at the different contours, we notice that they move up and to the right (parallel to each other) as we increase the profit c.

We want to increase profit as much as possible while still remaining in the feasible region.

So the optimal solution will be the last point the profit line sees before leaving the region, which will be a **vertex** of the convex polygon!

In general for linear programming, the optimal solution lies on a vertex of the feasible region.

Simplex (1947)

In general for linear programming, the optimal solution lies on a vertex of the feasible region.

Unless there is no optimal solution:

- If the LP is **infeasible**, where the constraints are too tight to be satisfied, such as: $x_1 \leq 1,\ x_1 \geq 2$.
- The constraints are too loose and give an **unbounded** feasible region, such as: $x_1, x_2 \geq 0$.

Idea: start with a vertex, compare its value to its neighbors, and move to the highest profit neighbor until we find a vertex with no better neighbors.

Why does this *local* test give us *global* optimality?

By geometry: all of the optimal vertex's neighbor vertices lie below a profit line, and so the rest of the polygon does as well.

Profit Maximization

What if the company decides to offer a new product, sprockets, which it will sell for a $13 profit.

But, sprockets and gears require some of the same packaging machinery to produce:

- Sprockets requiring $3\times$ the packaging labor as a gear,
- Together they cannot take more than 600 units of packaging labor.

$$\text{maximize:} \quad x_1 + 6x_2 + 13x_3$$

subject to:

$$\begin{aligned} x_1 &\leq 200, \\ x_2 &\leq 300, \\ x_1 + x_2 + x_3 &\leq 400, \\ x_2 + 3x_3 &\leq 600, \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

Now a 3D problem!

Production Planning

It turns out that the demand for sprockets is extremely seasonal.

The company analyst has obtained demand estimates for all months of the upcoming calendar year: $d_1, d_2, \ldots, d_{12}$.

They are very uneven, ranging from as low as 440 per month to 920 per month.

Current information about the company's workforce:

- There are 30 employees.
- Each produces 20 sprockets per month.
- Each has a \$2,000 per month salary.

Production Planning

Three ways to handle changing demand:

1. Overtime: we can pay workers 80% more than regular wages, but each worker is only allowed to work 30% extra time.
2. Hiring and Firing Workers: but this costs $320 and $400, respectively, for the paperwork per worker.
3. Storing Surplus: but this costs $8 a month per sprocket stored, we currently have no extra sprockets, and we do not want extra when the year ends.

How should we handle the changing demand of sprockets to minimize our overall cost?

This can be **formulated** as a linear program!

Production Planning

Step 1: Define the variables.

- $w_i =$ the number of workers during the ith month; $w_0 = 30$.
- $x_i =$ the number of sprockets made during the ith month.
- $o_i =$ the number of overtime sprockets in the ith month.
- $h_i =$ the number of workers hired in the ith month.
- $f_i =$ the number of workers fired in the ith month.
- $s_i =$ the number of sprockets stored at end of month i; $s_0 = 0$.

Note: since there are 6 variables for each month, that means that there are 72 total variables.

So our problem has 72 dimensions!

Production Planning

Step 2: Objective Function.

Recall that:

- Each worker gets paid \$2000.
- Hiring costs \$320.
- Firing costs \$400.
- Storage is \$8 per month.
- Overtime is 180% regular cost.

So we want to minimize the overall cost:

$$2000 \cdot \Sigma\, w_i + 320 \cdot \Sigma\, h_i + 400 \cdot \Sigma f_i + 8 \cdot \Sigma\, s_i + 180 \cdot \Sigma\, o_i$$

Production Planning

Step 3: Constraints.

First, all variables must be nonnegative:

$$w_i,\ x_i,\ o_i,\ h_i,\ f_i,\ s_i \geq 0, \quad i = 1, 2, \ldots, 12.$$

The total sprockets made per month (x_i) is regular amount (20 per worker) plus overtime (o_i):

$$x_i = 20w_i + o_i, \quad i = 1, 2, \ldots, 12.$$

The number of workers can change at the start of each month:

$$w_i = w_{i-1} + h_i - f_i, \quad i = 1, 2, \ldots, 12.$$

The number of stored carpets is the amount stored from the previous month (s_{i-1}), plus the number we made (x_i), minus the number we sold (d_i):

$$s_i = s_{i-1} + x_i - d_i, \quad i = 1, 2, \ldots, 12.$$

Overtime is limited (30% of 20 sprockets per worker per month):

$$o_i \leq 6w_i, \quad i = 1, 2, \ldots, 12.$$

Mixed Integer Linear Programming (MILP)

Now we can solve using Simplex!

Problem #1: The optimal solution might be *fractional*.

- What if we are told to hire 10.5 workers in March?
- We could try rounding, and we might be safe.
- But in general, **rounding will not work**.

Linear Programs can be solved optimally and efficiently **unless** some of the variables are required to have integer values.

Such a constraint means that we are dealing with a **Mixed Integer Linear Program (MILP)**.

This is an NP complete problem!

A common solution technique is called **branch-and-bound**, where we try to iteratively eliminate portions of the feasible domain until only one point is left.

Production Planning

Problem #2:

- Some of our constraints are **equality** instead of **inequality**.
- Our problem is a **minimization** instead of **maximization**.

This isn't a problem at all. A general LP can:

1. Be a maximization or minimization problem.
2. Have equality and/or inequality constraints.
3. Have nonnegative or unrestricted variables.

All of these versions of LPs can be *reduced* to each other.

Variants of Linear Programming

All of these versions of LPs can be *reduced* to each other:

1. Maximization to minimization (and vice-versa): just multiply the coefficients of the objective function by -1.
2.
   - Inequality $\sum_{i=1}^{n} a_i x_i \leq b$: define new variable $s \geq 0$, and then
   $$\sum_{i=1}^{n} a_i x_i + s = b.$$
   - Equality $ax = b$: create constraints $ax \leq b$ and $ax \geq b$.
3. Unrestricted variable x:
   - Define two new nonnegative variables $x^+, x^- \geq 0$.
   - Then replace all instances of x with $(x^+ - x^-)$.

Bandwidth Allocation

Suppose we are managing a network with the bandwidth shown on the board.

We want to make three connections:

- A connection between Alice and Bob that will pay $3 per unit of bandwidth.
- A connection between Bob and Claire that will pay $2 per unit of bandwidth.
- A connection between Alice and Claire that will pay $4 per unit of bandwidth.

We require that no edge's bandwidth is exceeded, and that each connection will get a bandwidth of at least 2 units.

How should we route these connections to maximize the network's revenue?

Bandwidth Allocation

Note: each connection has a short path (using only two nodes) and a long path (going through the third node).

- Define the short-path bandwidth allocated to the connection between Alice and Bob as the variable x_{AB}.
- Define the long-path bandwidth allocated to the connection between Alice and Bob as the variable x'_{AB}.

This means that our objective function is to maximize the value of the revenue:

$$3(x_{AB} + x'_{AB}) + 2(x_{BC} + x'_{BC}) + 4(x_{AC} + x'_{AC}).$$

All bandwidths are nonnegative:

$$x_{AB}, x'_{AB}, x_{BC}, x'_{BC}, x_{AC}, x'_{AC} \geq 0.$$

All connections must be at least 2 units:

$$x_{AB} + x'_{AB} \geq 2, \quad x_{BC} + x'_{BC} \geq 2, \quad x_{AC} + x'_{AC} \geq 2.$$

Bandwidth Allocation

We now need to encode the network itself as constraints...

Note: the total bandwidth passing through an edge cannot exceed that edge's bandwidth!

$$
\begin{aligned}
x_{AB} + x'_{AB} + x_{AC} + x'_{AC} &\leq 12 \quad [\text{edge } (a, A)],\\
x_{AB} + x'_{AB} + x_{BC} + x'_{BC} &\leq 10 \quad [\text{edge } (b, B)],\\
x_{BC} + x'_{BC} + x_{AC} + x'_{AC} &\leq 8 \quad [\text{edge } (c, C)],\\
x_{AB} + x'_{BC} + x'_{AC} &\leq 6 \quad [\text{edge } (a, b)],\\
x_{BC} + x'_{AB} + x'_{AC} &\leq 13 \quad [\text{edge } (b, c)],\\
x_{AC} + x'_{AB} + x'_{BC} &\leq 11 \quad [\text{edge } (a, c)].
\end{aligned}
$$

We can now solve using Simplex to get the optimal connections:

$$x_{AB} = 0, \quad x'_{AB} = 7, \quad x_{BC} = x'_{BC} = 1.5, \quad x_{AC} = 0.5, \quad x'_{AC} = 4.5.$$

Network Flow

Problem: our formulation has a variable for every possible path between any two users.

The number of these paths can grow exponentially with the number of users.

But that means the dimensionality of our problem can grow exponentially!

A much more clever (and scalable) formulation is called a Network Flow problem, which can be solved efficiently using the Ford-Fulkerson algorithm.