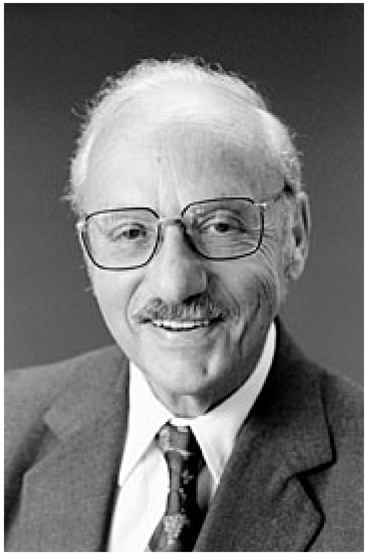
Optimization

Introduction and Overview

The Story of George Dantzig

Many of you will have heard the story of the student who arrived to class late, and copied down the homework problems off of the chalkboard. They were very hard problems. He couldn't solve them all. Dejected, he turned in the homework.

Later, the professor turned up at his home, very excited! "George, you have solved two unsolved problems!"



The Father of Linear Programming

This actually happened to George Dantzig, who went on to have a very long career, laying some of the foundations of operations research.

Operations Research

Operations Research really took off as its own field in the World War II era. The British military used it extensively in the war effort for things like radar location, supply chain, inventory, and logistics. After the war it spread into industrial and financial uses, and has continued to spread.

What else might you hear it called?

Operational Research (British)	Management Science
Optimization	Industrial Engineering
Decision Science	Applied Math

Optimization

The standard optimization problem looks like this:

Minimize (or maximize)

A goal function (the objective)

Subject to

A series of constraints that must be met

Optimization Examples

Maximize revenue from selling concert tickets by setting prices so that they all sell

Minimize inventory storage costs, but be able to ship all orders within 48 hours

Minimize transaction costs while still tracking the S&P 500 with 99% efficiency

Decide which packages to put on this truck, subject to weight and volume limits on what the truck can carry

Plan the NFL schedule. No team can have more than 3 home (or away) games in a row. The Giants and the Jets use the same stadium, and there's a concert in Miami's stadium on November 11.

DataScience@SMU

Linear Programming

Linear Programming

We'll start with linear programming and spend most of our time on that. It's the basic approach to mathematical modeling and optimization.

Later we'll talk about other cases:

- Integer programming
- Nonlinear programming
- Stochastic programming
- Network Flows

What makes a Linear Program?

Minimize (or maximize)

A linear goal function (the objective)

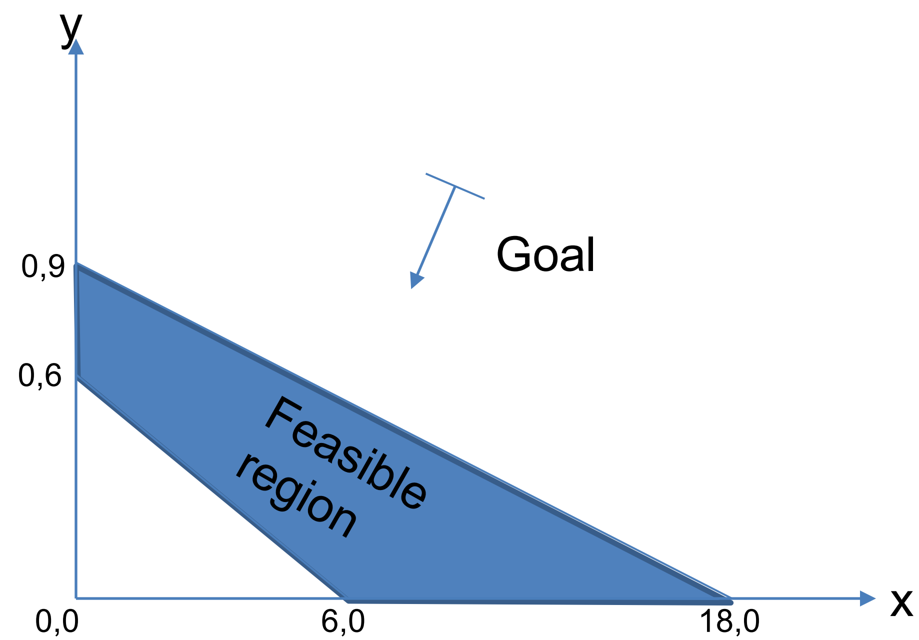
Subject to

A series of linear constraints that must be met

The Feasible Region

Let's look at a small, two-dimensional linear program:

Minimize	$2x + 5y$
Subject to	$x + y >= 6$
	$x + 2y <= 18$
	$x >= 0$
	$y >= 0$



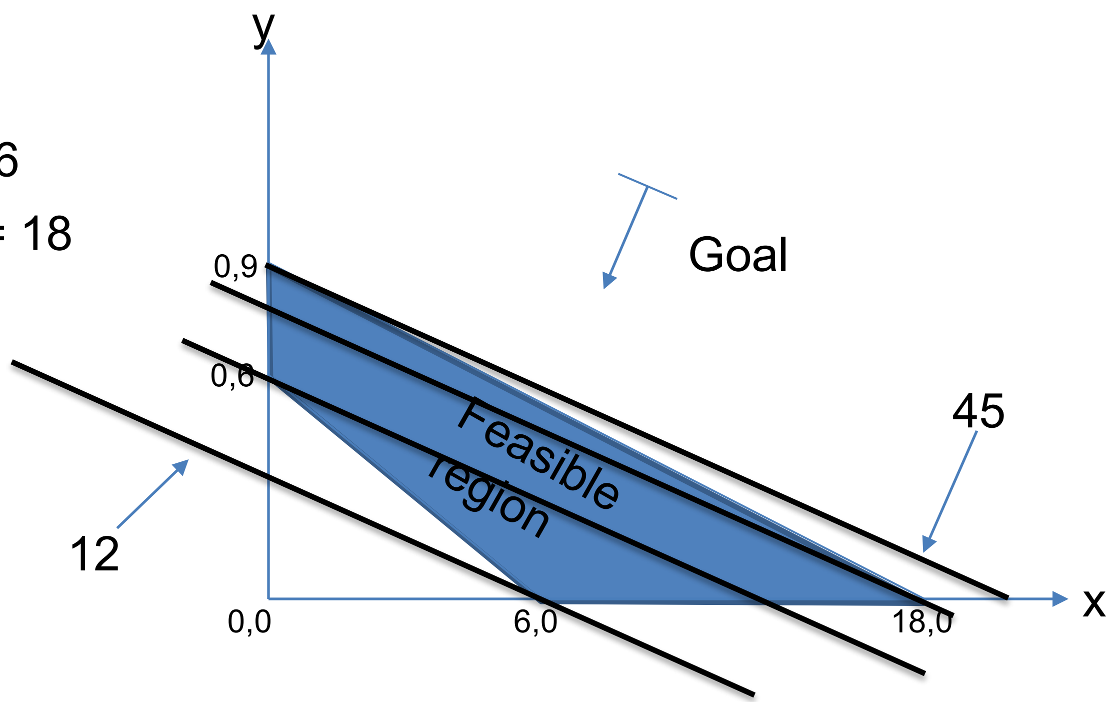
The point within the feasible region that minimizes the function is (6,0).

Which point would maximize the function?

The Feasible Region

Let's look at a small, two-dimensional linear program:

Minimize	2x + 5y
Subject to	x + y >= 6
	x + 2y <= 18
	x >= 0
	y >= 0



In this two dimensional case, we can draw the objective function at different values. Over the feasible region, the objective ranges from 12 to 45. Anything beyond that moves us out of the feasible region.

Linear Programming Geometry

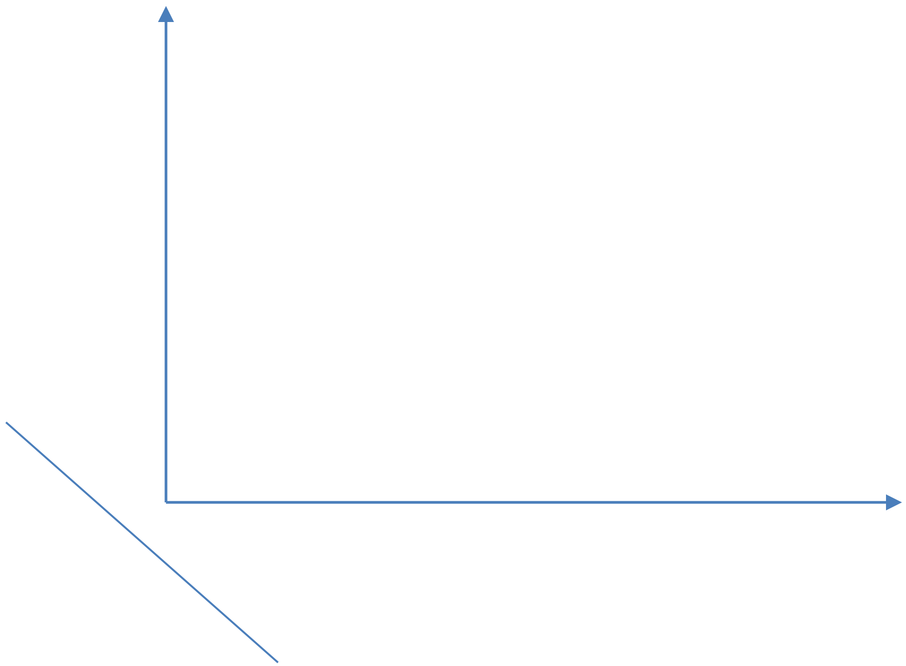
In our previous example, graphing the feasible region and the objective function at various values allowed us to find both the minimum and the maximum. That's an important way to understand what's happening with a particular model, but is difficult to do with problems in more than 3 dimensions.

There are a few special cases regarding the feasible region that we'll look at next.

Infeasible

A problem may have no feasible region, for example:

Minimize	$3x + 4y$
Subject to	$x + y <= -1$
	$x >= 0$
	$y >= 0$

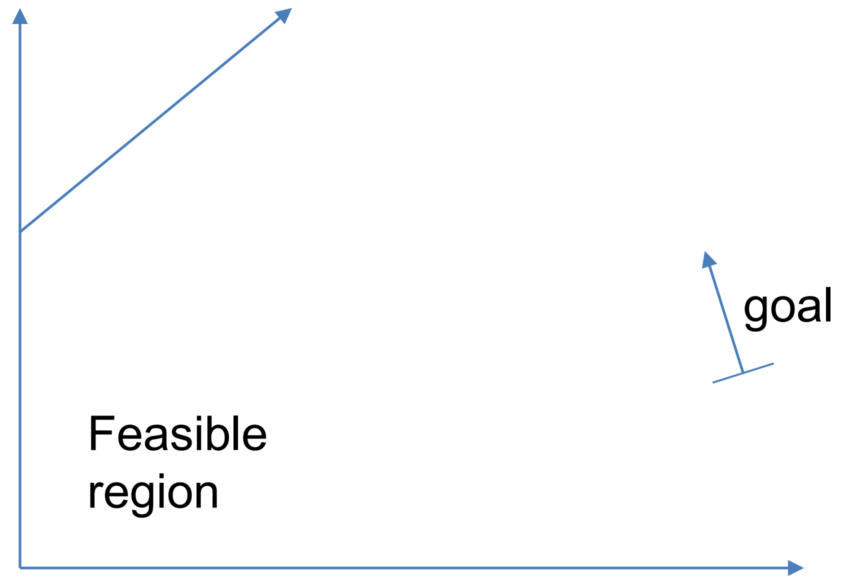


Such problems have no solution. That in itself can sometimes tell you something. Often it means that you've created an incorrect model.

Unbounded

Some problems may have an unbounded feasible region, and so have an infinite objective value.

Minimize	x – 2y
Subject to	-x + y <= 10
	x >= 0
	y >= 0



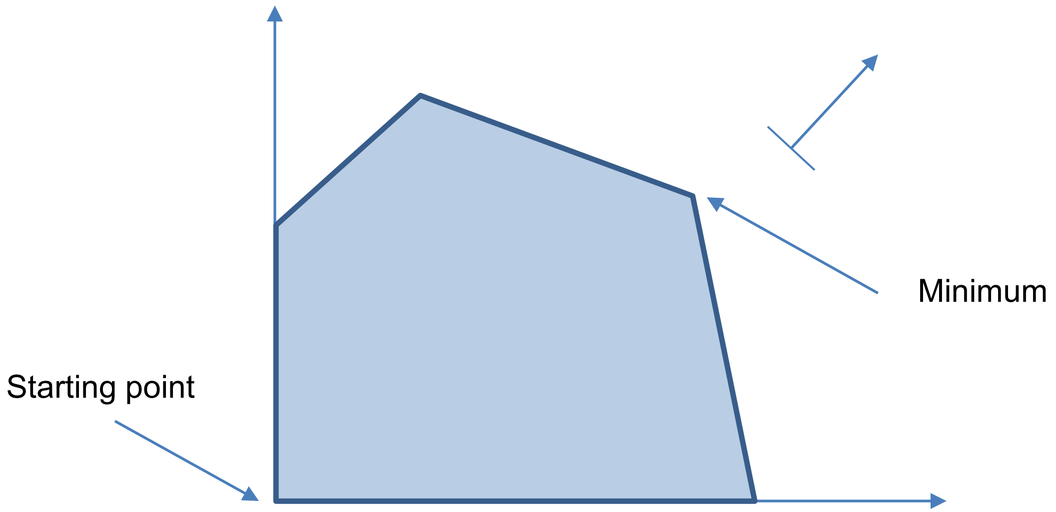
Unbounded problems can be interesting, but the typical model is both bounded and feasible. We'll focus on those problems.

DataScience@SMU

The Simplex Method

Solution Methods

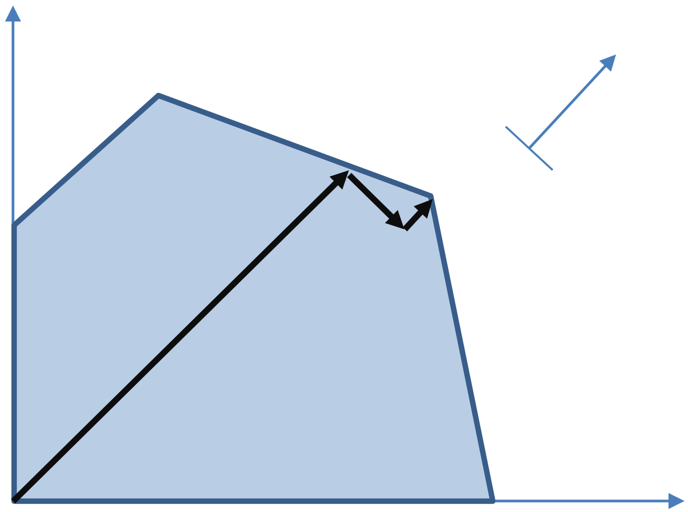
Imagine that we have the following feasible region and objective minimum direction:



We want a process that moves us from the starting point (zero) to the minimum.

Interior vs Exterior Point Methods

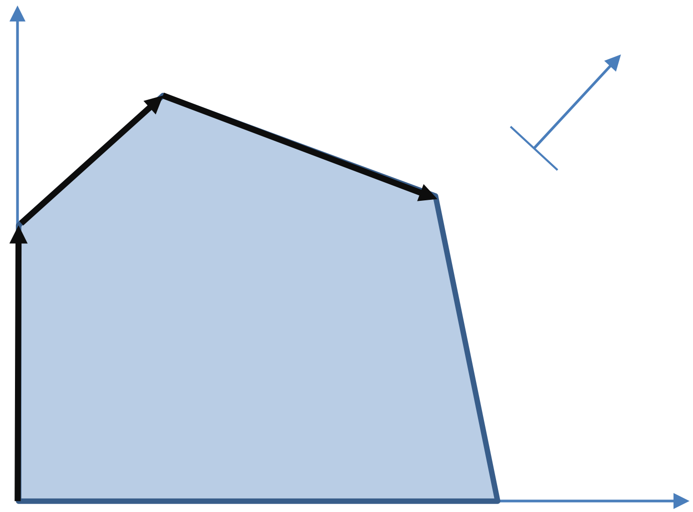
We might move through the feasible region in the objective direction, like so:



An interior point method might follow the arrows above to find the minimum.

Interior vs Exterior Point Methods

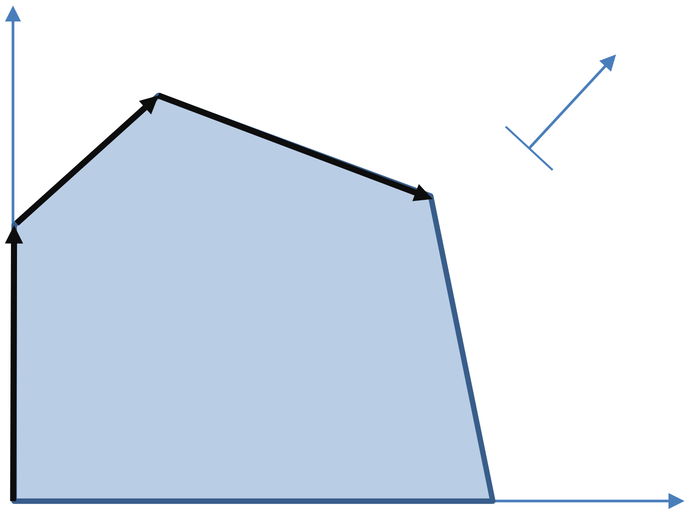
We could also move around the edges from vertex to vertex:



An exterior point method might follow the arrows above to find the minimum. The standard exterior point method, invented by George Dantzig, is the simplex method.

Extreme Points

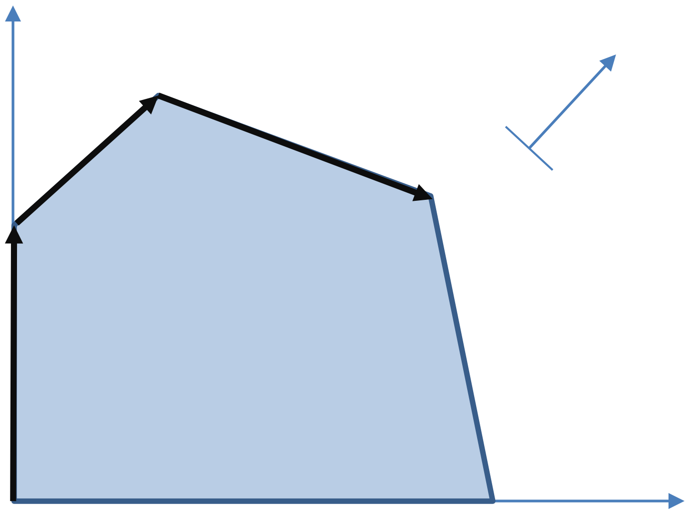
What if the minimum is not on the outside of the feasible region?



It can be proven that when an optimal solution exists for a linear program, then an optimal extreme point (vertex) also exists. Thus exterior point methods (like the simplex method) can be used to find the optima.

The Simplex Method

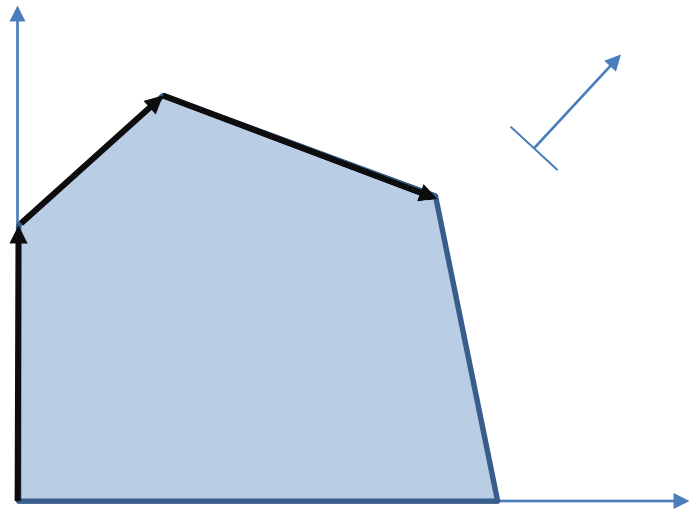
The Simplex Method is a global search using local decisions.



At each point, we want to choose a direction that would improve the objective. There can be more than one choice. Notice in this case we could have circled around the outside of the polygon in the other direction.

We move from point to point until we reach a spot where any direction we choose would result in the objective getting worse. At that point we stop, having found the maxima.

How To Solve Problems with The Simplex Method



The whole procedure is numerically quite complicated, and beyond the scope of this course. It is possible to solve large problems with thousands or millions of dimensions.

The process requires calculating possible directions to move from each vertex (including making sure that the move is an improvement in the direction of the objective), calculating the distance to be moved and the location of the next vertex.

How To Solve Problems with The Simplex Method In Practice

Industrial strength solver software is readily available. It can be embedded within programs that query data, set up models, solve, adjust, resolve, and return solutions.

Beyond learning the individual steps of the Simplex Method, almost no one should attempt to implement the Simplex Method in software, because it has already been done so well by so many people.

- FICO Xpress (my favorite)
- Gurobi
- IBM CPLEX
- COIN-OR (open source)

All of these products have implementations of the Simplex Method, interior point methods, and other algorithms that we will see. Even MS Excel has a basic LP solver.

DataScience@SMU

Linear Programming

Basic examples

The Knapsack Problem

Maximize

$$3x_1 + 2x_2 + 5x_3 \qquad + 6x_5 + 2x_6$$

Subject To

$$4x_1 + 6x_2 + 4x_3 + 3x_4 + 4x_5 + 5x_6 <= 60$$

All x >= 0

The goal of this problem is to select the most valuable items (x_1, x_2, etc) to fit within a knapsack of a fixed volume. In this case, there's a constraint that limits the volume selected to 60, and gives each item's size. The value of an item is expressed by the objective function.

We can examine possible solutions using each variable and see that, for the optimal solution, we should choose the variable with the highest value per volume. In this case, x_5 is the variable that gives the highest ratio, and so a solution of $x_5 = 15$ gives us the optimal objective of 90.

The Knapsack Problem - Variations

Maximize

$$3x_1 + 2x_2 + 5x_3 \qquad + 6x_5 + 2x_6$$

Subject To

$$4x_1 + 6x_2 + 4x_3 + 3x_4 + 4x_5 + 5x_6 <= 60$$

All x >= 0

- How would the model change if we were required to pack at least 2 of x_4?
- How would the model change if there were only 10 of item x_5 available to pack?
- How would the solution change if the item sizes were not factors of 60?

Mix Problems

A pet food supplier produces dry cat food. The cat food consists of three main ingredients: chicken, tuna, and bone meal. These ingredients contain three nutrients: protein, calcium, and fat. The ingredients must be mixed to achieve a certain level of each nutrient at the lowest total cost. Here are the nutrient contents per pound of each ingredient:

Nutrient	Chicken	Tuna	Bone Meal
Protein	25	15	25
Calcium	15	30	20
Fat	5	12	8

Prices per pound are \$3.00 for chicken, \$2.40 for tuna, and \$3.60 for bone meal.

Protein needs to be within 18-22 per pound, fat within 6-12, and calcium at least 20.

Mix Problems

Minimize

cost = \$3.00 chicken + \$2.40 tuna + \$3.60 bone meal

Subject To

18 <= 25 chicken + 15 tuna + 25 bone <= 22

20 <= 15 chicken + 30 tuna + 20 bone

6 <= 5 chicken + 12 tuna + 8 bone <= 12

all ingredients >= 0

Distribution Problems

A company has two factories which make its products, Factory A and B.

The company has 6 warehouses (1-6) across the country which hold the factory output for distribution.

Each combination of factory and warehouse has a cost to ship items from the factory to the warehouse, c_{A1} for example.

Select which factory will supply each warehouse, minimizing cost.

Distribution Problems

Minimize

$$c_{A1}A1 + c_{A2}A2 + c_{A3}A3 + c_{A4}A4 + c_{A5}A5 + c_{A6}A6 +$$
$$c_{B1}B1 + c_{B2}B2 + c_{B3}B3 + c_{B4}B4 + c_{B5}B5 + c_{B6}B6$$

Subject To

$$A1 + B1 = 1$$
$$A2 + B2 = 1$$
$$A3 + B3 = 1$$
$$A4 + B4 = 1$$
$$A5 + B5 = 1$$
$$A6 + B6 = 1$$

Distribution Problems - Variations

Minimize

$$c_{A1}A1 + c_{A2}A2 + c_{A3}A3 + c_{A4}A4 + c_{A5}A5 + c_{A6}A6 +$$
$$c_{B1}B1 + c_{B2}B2 + c_{B3}B3 + c_{B4}B4 + c_{B5}B5 + c_{B6}B6$$

Subject To

$$Ai + Bi = 1 \quad \text{(for all } i=1...6)$$

- How would the model change if each factory had to supply exactly three warehouses?
- Suppose each warehouse had a demand, Di, and each factory produced a supply, Si. How would you model the required demand and limited supply?

Inventory Problems

A company wants to plan its quarterly production of two items A and B for 2020. Demand is forecast as:

	Q1	Q2	Q3	Q4
Item A	40	50	60	40
Item B	60	60	70	60

At the beginning of Q1, the plan is that the company will have 10 of item A and 15 of item B on hand. They also want to end 2020 with 10 of each item on hand.

The holding costs of items A and B are $10 and $8 for any inventory not sold at the end of each quarter. The maximum amount of each item that can be produced each month is 50 item A and 65 item B.

Create a production plan that meets demand while minimizing inventory costs.

Inventory Problems

Minimize

$$10\ Q1A + 10\ Q2A + 10\ Q3A + 10\ Q4A +$$
$$8\ Q1B + 8\ Q2B + 8\ Q3B + 8\ Q4B$$

Subject To

$10 + P1A – 40 = Q1A$

$Q1A + P2A – 50 = Q2A$

$Q2A + P3A – 60 = Q3A$

$Q3A + P4A – 40 = 10$

$15 + P1B – 60 = Q1B$

$Q1B + P2B – 60 = Q2B$

$Q2B + P3B – 70 = Q3B$

$Q3B + P4B – 60 = 10$

All P and Q variables >= 0

PiA <= 50

PiB <= 65

Manufacturing Problems

A company manufactures tables and chairs. Each table generates $100 profit and each chair $60 profit.
Tables require 2 units of wood and 1 hour of labor.

Chairs require 1 unit of wood and 2 hours of labor.

Each week, the company has 59 units of wood and 40 hours of labor available. How many tables and chairs should it manufacture each week?

Manufacturing Problems

Maximize

$$100\,T + 60\,C$$

Subject To

$$2\,T + C <= 59$$

$$T + 2C <= 40$$

$$T, C >= 0$$

Manufacturing Problems - Variations

Maximize

$$100\,T + 60\,C$$

Subject To

$$2\,T + C <= 59$$

$$T + 2C <= 40$$

$$T, C >= 0$$

- If the company could get three more units of wood each week, how much would that be worth?
- If the company could only get one more unit of wood each week, how much would that be worth?

DataScience@SMU

Integer Programming

What's a Fractional Table?

Maximize

$$100\,T + 60\,C$$

Subject To

$$2\,T + C <= 60$$

$$T + 2C <= 40$$

$$T, C >= 0 \qquad\qquad T = 80/3,\ C=20/3$$

As we saw in the previous example, we have a solution that indicates a fractional value for the number of tables and chairs to produce. Often we will have variables and decisions that only make sense as discrete, integer variables (including binary yes/no decision variables.) Sometimes you get lucky and the optimal solution to the linear program is already integer, but you can't count on it.

Graphical Representation

<Here I'd like to use the magic whiteboard>

Model Divergence

The linear version of an integer problem is called the LP relaxation.

We can see how the difference between the linear solution space and the integer solution space causes problems by generating linear solutions to an integer problem.

There are two main approaches to address this issue:

- Cutting
- Branching

Cutting

<Return to magic whiteboard>

Cuts

Cuts reduce the linear solution space without reducing the integer solution space. When completely successful, cuts will make it so that all extreme points are also integer points, and thus the LP relaxation solution will be integer.

Some cuts are easy to generate, as we've seen from the 2-dimensional example. However, some cuts are very computationally difficult to find.

For difficult integer problems, cuts are "part of this nutritious breakfast!"

Branching

<Return to magic whiteboard>

Branching

Branching is similar to cutting in its effects. It will remove sections of the solution space which contain no integer solutions (or integer solutions which have been shown to be non-optimal.)

It is said that if you can branch, you can cut. We've seen in our example that some of the branches we used were the same as some of the cuts we made.

Branching Ideas

Besides branching on variables, it is also possible to branch on constraints. One example might be a warehouse site selection problem. If your variables are binary variables that indicate a yes/no decision on a particular warehouse site, you might branch with

$$\sum x <= 2 \qquad \text{and} \qquad \sum x >= 3$$

If the LP relaxation returned a result that indicated that the optimal (linear) solution were to open 2.5 warehouses, branching in this way can help resolve that issue.

It is generally helpful to branch on the most "important" variables first. Commercial software tools will automatically select branching variables for you, but it can sometimes be advantageous to override their selection.

Solving Integer Programs

All modern linear programming products contain integer programming capabilities (sometimes at an extra price.) They will generally have the following functionality:

- Automatic cut generation
- Branch and bound
- Heuristic solution generation for good bounds, including rounding and neighborhood searches

One issue of concern is that an integer program can be much harder to solve to optimality than a linear program. In these cases problem size, model “tightness”, and modelling decisions can come into play. Finding a good solution is often enough.

DataScience@SMU

Further Variations

Other Ideas

Linear and Integer Programming are the foundation of optimization, and have been studied extensively. They are in use today across a multitude of industries, rightly so. Given their speed and efficiency, they can be very valuable.

Further variations can come into play. Let's take a brief look at these ideas:

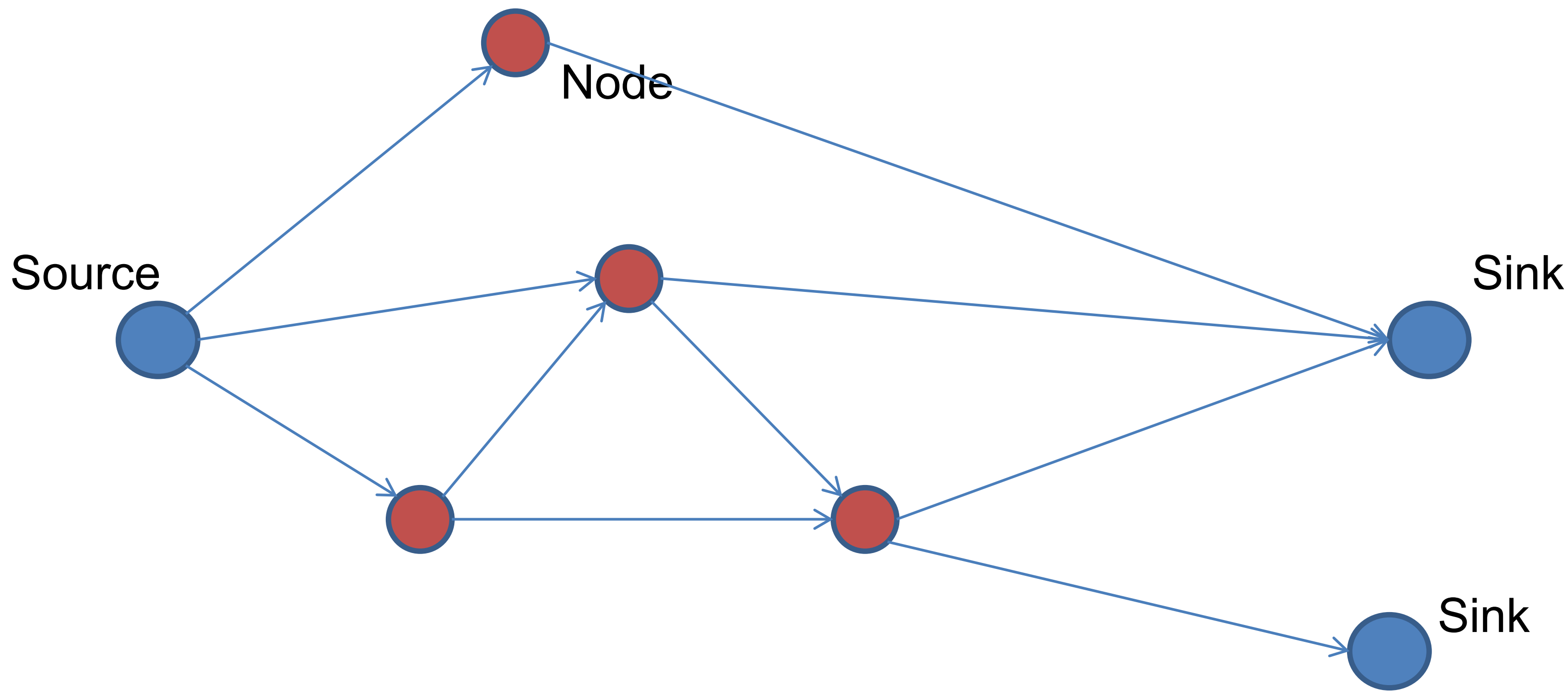
- Network Flows: specialized LPs which model a network
- Stochastic Programming: LPs which contain non-deterministic functions
- Nonlinear Programming: Objective function or constraints are no longer required to be linear

DataScience@SMU

Network Flows

Network Flows

Network Flows are specialized versions of linear programs. Some of the assumptions allow for the problems to solve more quickly.



Network Flows

Common Network Flow problems:

- Move X units from the source to the sink at lowest cost
- Move the maximum units from the source to the sink

Arc variables can have both a cost (the objective coefficient) and a capacity (the upper bound.) If resources are required to move across a given arc, then a non-zero lower bound can be used.

If you can assume that there are either no cycles (loops) that can be made, the problem becomes even easier. In cases where cycles are allowed, they typically have to either have a capacity limit (upper bound) or else they have to have a positive cost. Negative cost cycles usually mean an unbounded problem.

Network Flows

Network flow constraints:

Source:

$\sum x_{out} = s$ or $\sum x_{out} <= s$

Sink:

$\sum x_{in} = s$ or $\sum x_{in} <= s$

Node:

$\sum x_{in} - \sum x_{out} = 0$

Network Flow models have the attribute that all of the coefficients in the matrix are either 0, 1, or -1. That allows for other numerical advantages in solving.

DataScience@SMU

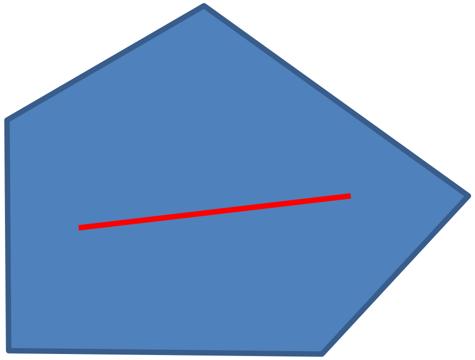
Nonlinear Programming

Abandon all hope,

Ye who enter here

Convex

One of the assumptions underlying linear programming is that the solution space is convex.

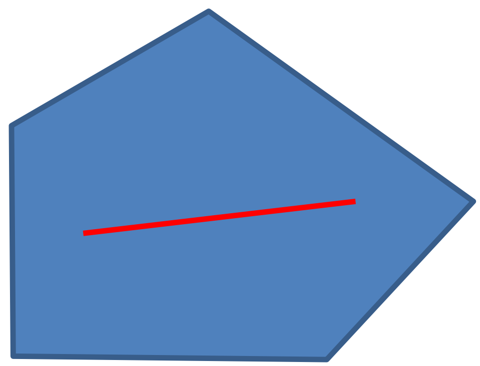


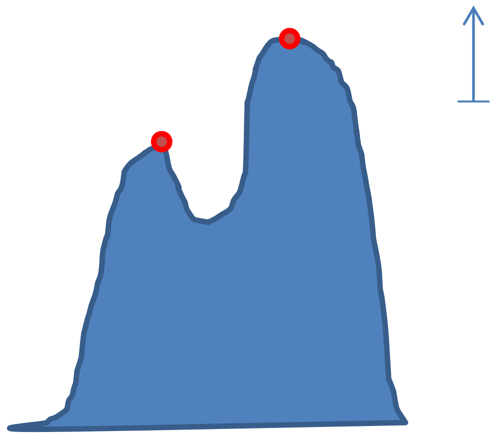
Convexity here means that if you choose any two feasible points in the solution space, the line connecting those two points will also be entirely within the solution space.

The implication is that if we find a point where any movement results in a worse solution, then that point is the global optimal solution.

Concave

Solution spaces that are not convex are called concave, and that presents a problem.





In the second solution space, we cannot connect the two points without leaving the solution space. Also, if the objective is as shown, the lower point on the left is a local optima, but not the global. In this case, any direction we move will make the solution worse, but we are not at the global optimal solution.

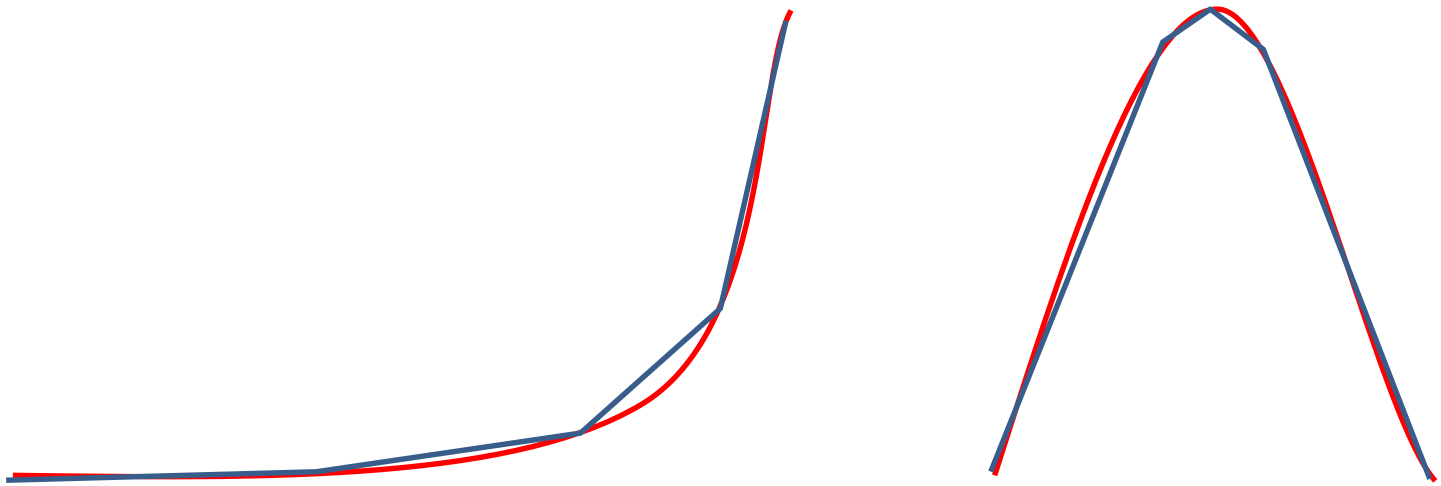
Nonlinear Optimization

Nonlinear optimization is often a struggle to see how many of the following assumptions you can make. The more of these you can use, the better, and often it is worth changing your model.

- If not linear, quadratic
- If not quadratic, convex
- Quadratic objective with linear constraints
- Quadratic objective with quadratic constraints
- Objective function has a derivative.

There are good quadratic solvers available. If derivatives are available, there are numerical methods to search through a convex space in the direction of the fastest improvement.

Nonlinear Integer Optimization



If your nonlinear program happens to be an integer program, you can often replace your nonlinear function with a piecewise linear function.

DataScience@SMU

Stochastic Programming

Stochastic Programming

For all of our previous examples, we had deterministic functions we optimized against. Often that doesn't hold.

The famous textbook example of stochastic programming is the newspaper problem. We have to decide today how many newspapers to print for tomorrow. How many we can sell tomorrow is unknown, although we have some forecasts.

Each paper we print costs X cents. We sell papers for Y cents.

Unsold papers are worthless. But we also don't want to print too few and pass up the opportunity to sell more. How do we maximize our profits?

These problems are similar to hotel rooms, airplane seats, and other problems where the production decision has to be made ahead of time.

Stochastic Programming

How do we maximize our profits? Key to answering this question is in understanding it. We won't know our actual profits until the day is over, when it is too late to do anything.

What we might be asking could be:

- How do we maximize our expected profits?
- How do we maximize our minimum profits?

Or some further variation. We can then adjust our objective function to align with our real-world objectives.

DataScience@SMU