CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

Today's Outline

- About this class.
- About me.
- About you.
- Modeling:
  - What is it?
  - Why should we care about this stuff?[1]
- A first model

[1] Besides, of course, the fact that you want to get a good grade

Class Overview

- As Snoop Dogg says, read the syllabus!
  - Posted on course Canvas page
  - It has many details about rules and regulations for the course
- Meeting Times: Monday-Wednesday 4:00PM-5:15PM
- Meeting Location: 1800 Engineering Hall

My Office Hours:

- Tuesdays: 9AM-10AM
- Fridays: 11AM-12PM
- By Appointment (last resort!)
  - My calendar is available at: `http://tinyurl.com/37upk9dp`
  - To make a meeting, please check my calendar, compare to your own free times, and suggest one or more times when we are both free

Course HomePage

- Canvas `https://canvas.wisc.edu/courses/397595`
- Lecture slides posted there. Posted as large modules, that may from time to time be updated.
- Announcements: You will be responsible for reading
- Organized into modules: One for each part of the course
- Homework assignments and submissions, grades, ...
- Piazza Q&A Forum:
  - Available through Canvas. Check that you have access. If not, find our class signup link at: `https://piazza.com/wisc/spring2024/sp24isye524001`
  - I prefer that you *do not* email me (or the TAs) questions about the course. Rather, I prefer you ask them on the Q&A forum.
  - Unless the question is of a personal nature, I will copy email questions to the Q&A forum and answer them there.

Teaching Assistants

- Mr. Eric Brandt
  - **Office Hours:** 2:30-3:30PM MW, 3146 ME Bldg.
  - email: `elbrandt@wisc.edu`
- Mr. Sanjai Pushpa
  - **Office Hours:** 2PM-3PM RF, 3146 ME Bldg.
  - email: `pushpa@wisc.edu`

Brother, can you spare a dime?

- There are ≈ 190 of you in the class
- We have 40 hours of TA effort/week
- And (hopefully) some grading help
- The TAs and I are here to help you learn, but please be respectful of our time

Prerequisites

- **Comfortable with Programming:** `Comp Sci200/300` You don't necessarily need to know Java, but you do need to be comfortable with a computer programming language. If you are not, then this course is not really for you.
- **Comfortable with Linear Algebra:** `Math 340 or equiv.` You need to know a little bit of Calculus, be familiar with Matrix notation, and have the mathematical sophistication necessary to not be intimidated by symbols like $\sum$, $\forall$ and $\in$.
- **Computer Sophistication:** We will be learning/installing/using software Julia/JuMP in the course. You will be responsible for ensuring you have working coding environment. You also will (probably) need to learn LaTeX to type the mathematical description models you build.

Course Details

- Learning is better if you participate.
    - I will call on you during class. (Gasp!)
- Even though this is a huge class, I really would like the class to be interactive.
    - We are likely to spend time working through models
    - If you have a laptop, you may wish to bring it to class, in case we have interactive sessions.

Coursework Details

- Assignments
  - Expect to be relatively time-consuming
  - Largish assignments assigned every couple weeks or so. (Six assignments planned)
- Exams:
  - Two in-class midterm exams. (February 28, April 8)
  - Final exam (May 10)

Homework: Roughly Bi-weekly

- Homework submit through Canvas.
- Homework will be graded on a "spot" basis: Usually one or two problems randomly chosen and graded. The rest checked for complete effort (but not correctness).
- You are responsible for assessing your progress on the problem sets
- Problem solutions will be posted soon after due.
- The lowest homework score will be dropped
  - Purpose: Accommodate *all* reasons for missed work
- Late homework will be penalized 10%: And cannot be accepted once the solutions are posted.

Grading

- 30% Regular Homework Sets
- 20% Midterm #1
- 20% Midterm #2
- 30% Final Exam

- Syllabus has grade cutoff percentages, but I typically grade on a curve. Lowering grade cutoffs as necessary.
- The median performer in the class should get an `AB` grade, but below the median will likely get a `B` or lower.

Course Texts

- None Required.

Recommended

- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. The book is available for free here: http://stanford.edu/~boyd/cvxbook/.
- H.P. Williams. *Model Building in Mathematical Programming*, 5th Edition. Wiley, 2013.
- R.L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

Cheater!!!!

Please don't cheat

- It will make me sad.
- And mad.

- Homework assignments can (and should!) be discussed together
- However, you must write up your answers independently!
- If too much "collaboration" on homework is noticed, I reserve the right to give all parties involved 0% on the assignment
- Using a LLM to do your homework is not an ethical use of that resource

Mathematical Topics

1. Linear and Network Programming ($\approx$ 9-10 lectures)
2. Convex Programming ($\approx$ 6 lectures)
3. Integer Programming ($\approx$ 8 lectures)
4. Stochastic Programming ($\approx$ 2 lectures)

Course Objectives

1. The ability to write down an algebraic formulation of an optimization model that captures the main decision elements of practical problems.
2. The ability to categorize optimization models, and understand the implications of modeling on algorithm performance
3. To understand the tradeoff between model accuracy and tractability and to consider the feasibility of alternative design solutions
4. The ability to explain, at a non-technical level, how optimization may be applied to decision problems.
5. To become familiar with the operation of state-of-the-art optimization software, including parameters that may significantly affect software performance
6. Use the Julia language and JuMP modeling package;
7. Have Fun (and work hard!)

Great Expectations

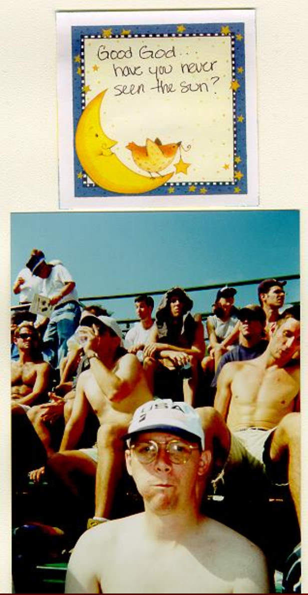
I am expected to...

- Teach
- Answer your questions
- Be at my office hours
- Give you feedback on how you are doing in a timely fashion

You are expected to...

- Learn
- Attend lectures and participate
- Do the problem sets
- Not be rude, if possible.
  - Sleeping, Talking, Showing up late
  - Cell Phones, Texting, Tweeting, Surfing
  - Leaving in the middle of lecture

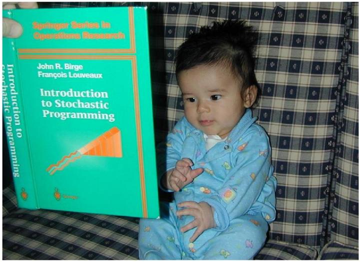
About me...



- B.S. (G.E.), UIUC, 1992.
- M.S., OR, GA Tech, 1994.
- Ph.D., GA Tech, 1998
- 1998-2000 : MCS, ANL
- 2000-2002 : Axioma, Inc.
- 2002-2007 : Lehigh University
- Research Area: Large-Scale Optimization
- (Current) Applications: Energy, Defense
- Married. One child, Jacob who is a junior in college. (Studying math)
- Hobbies: Golf, Human Pyramids, Integer Programming

Picture Time/About You

- I typically learn the names of all the students in my class.
- This may be too difficult for me this semester—I apologize!
- But I still would like to know about you—Please fill out the course interest survey on Canvas



Evil?!

- This is not an easy course
- Eric, Sanjai, and I really want you to learn the material and do well
- We will all have to work hard to make this happen
- With 190 students in the class, we cannot babysit you and provide a “hands-on” level of service. Please be professional.
- Please be patient with me—I have never taught a class this large before.

Decision Problems

- In this course you will learn to model decision problems

Decision Problems

1. How many clerks do I need at my grocery store?
2. How much capacity should I add to my telecommunications network?
3. How should I design my new airplane wing?
4. Is my new airbag design safer than the previous design?
5. What types of aircraft should fly each flight in a schedule?
6. What stocks should I buy?

- Warning! Not all of these questions are best answered by *optimization* models

Models

"A model should be as simple as possible and yet no simpler"
–Albert Einstein

Why Model

1. To get an answer!
2. From building a model, we can gain insight.
3. We can "experiment" with a model.

Types of models

- Physical
  - Airline wing design
- Abstract
  - Statistical: Time Series, Regression, etc...
  - Simulation
  - Economic
  - **Optimization!**

Common Fallacies

1. How can anyone possibly get and be confident in the data that makes up the model?
    - We're not. The analyst must understand the reality of the process to deduce whether the model solution makes sense
2. It's "too abstract"
    - It's not. The analyst must be able to explain *why* the solution approach is proper
3. If we got an answer on the computer, it must be right!
    - It's not. All models are wrong. But some are useful. (George Box)

The upshot!

(Optimization) models can be an important tool in a decision-making process.

Components of an Optimization Model

1. Decision variables
   - Variables representing the unknown quantities
2. Constraints
   - Requirements that all solutions must satisfy, (expressed algebraically)
3. Objective
   - A quantity that you would like to make as small or as large as possible.

Variables:

- x: Pounds of barley to purchase
- y: Pounds of hops to purchase
- z: Gallons of beer made

Constraints:

- $y \leq 2$
- $x \leq 8y$
- $z = 0.4x + 0.9y$

Objective:

- $\max z$

Another View at Model Components

1. Inputs
   - Sets. Used typically for algebraic models.
     - e.g., P: Set of products, I: Set of locations
   - "Numbers". These are called parameters. The parameters may be indexed over sets.
     - e.g., u_p: The maximum amount of product p available
2. Decision Variables
   - "Numbers you are allowed to change". It is the goal of the optimization to find the "best" values of these controls (or decision variables). Decision variables can also be indexed over sets
     - e.g., z_i: Gallons of beer to ship to location i
3. Outputs
   - These may be optimal values of the decision variables, or a derived value, such as the objective function value
   - e.g.: $\sum_{i \in \text{Madison}} z_i$

Categories of Optimization Models

- Linear vs. Nonlinear?
  - Are the functional relationships between decision variables linear functions or nonlinear functions?
- Convex vs. Nonconvex?
  - Are the functional relationships convex?
- Discrete vs. Continuous?
  - Must the decision variables take only discrete values?
- Deterministic vs. Stochastic?
  - Is uncertainty in the model explicity considered?

The upshot

- These categorizations have a significant impact on the tractability of an instance
- You should be able to categorize problem instances

Solving

- Actually involves gathering and processing data: Turning your model into an instance.
  - Model : A structure containing (algebraic) relationships between entities.
  - Instance : A combination of data and model that can be solved. (i.e – it has "numbers")
  - Spreadsheet models "blur" this distinction, that's why I don't like them very much.
- (Algebraic) Modeling languages are better!
  - Have hooks to solvers
  - Many have hooks to spreadsheets and databases
- We will use JuMP, a package in the Julia programming language to build our models.
- We will (try to) teach some best modeling practices about abstraction—separating model from data

524—Gateway to other optimization courses (usually more advanced)

- Linear programming (CS 525)
- Nonlinear optimization (CS 726, 730)
- Convex analysis (CS 727)
- Integer Optimization (CS 728)
- Stochastic/Dynamic programming (CS 719, 723)

Selected applied topics:

- Machine learning (CS 760, 761, 762)
- Optimal control (ECE 719, 819, 821)
- Robot motion planning (ME 739, 780)

(Many of these are cross-listed across other departments.)

Top Brass example

Top Brass Trophy Company makes large championship trophies for youth athletic leagues. At the moment, they are planning production for fall sports: football and soccer.

Each football trophy has a wood base, an engraved plaque, a large brass football on top, and returns \$12 in profit.

Soccer trophies are similar except that a brass soccer ball is on top, and the unit profit is only \$9.

Since the football has an asymmetric shape, its base requires 4 board feet of wood; the soccer base requires only 2 board feet. There are 1000 brass footballs in stock, 1500 soccer balls, 1750 plaques, and 4800 board feet of wood.

What trophies should be produced from these supplies to maximize total profit assuming that all that are made can be sold?

football soccer both

Top Brass data

Recipe for building each trophy

	wood	plaques	footballs	soccer balls	profit
football	4 ft	1	1	0	\$12
soccer	2 ft	1	0	1	\$9

Quantity of each ingredient in stock

	wood	plaques	footballs	soccer balls
in stock	4800 ft	1750	1000	1500

Top Brass Model Components

1. **Decision variables**
   - f: number of football trophies built
   - s: number of soccer trophies built
2. **Constraints**
   - $4f + 2s \leq 4800$ (wood budget)
   - $f + s \leq 1750$ (plaque budget)
   - $0 \leq f \leq 1000$ (football budget)
   - $0 \leq s \leq 1500$ (soccer ball budget)
3. **Objective**
   - Maximize $12f + 9s$ (profit)

Top Brass model (optimization form)

$$\begin{aligned}
\underset{f,\,s}{\text{maximize}} \quad & 12f + 9s \\
\text{subject to:} \quad & 4f + 2s \leq 4800 \\
& f + s \leq 1750 \\
& 0 \leq f \leq 1000 \\
& 0 \leq s \leq 1500
\end{aligned}$$

- This is an instance of a *linear program* (LP), which is a common type of optimization model.
- We have decision variables and parameters.

Top Brass model (generic)

$$\begin{aligned} \underset{f,s}{\text{maximize}} \quad & c_1 f + c_2 s \\ \text{subject to:} \quad & a_{11} f + a_{12} s \leq b_1 \\ & a_{21} f + a_{22} s \leq b_2 \\ & \ell_1 \leq f \leq u_1 \\ & \ell_2 \leq s \leq u_2 \end{aligned}$$

- By changing the parameters, we create different *problem instance*.
- It's good practice to separate parameters (data) from the algebraic structure (model).
  - This makes it easy to try out different parameter settings, so see how they affect the optimal values of the decision variables.

Top Brass code (IJulia notebook)

```
using JuMP
m = Model()
@variable(m, 0 <= f <= 1000)      # football trophies
@variable(m, 0 <= s <= 1500)      # soccer trophies
@constraint(m, 4f + 2s <= 4800)   # total board feet of wood
@constraint(m, f + s <= 1750)     # total number of plaques
@objective(m, Max, 12f + 9s)      # maximize profit
```

Note: we did *not* separate the data from the model in this example! Next class, we will see how we can generalize the code.

Getting Started with Julia

1. Get up and running with Julia + IJulia + JuMP and jupyter notebooks, following the instructions posted on Canvas.
2. Work through tutorials for Julia and JuMP.
3. Load the file Top Brass.ipynb in IJulia and confirm that you can reproduce the results shown in class.
4. Try to obtain an academic license and install Gurobi, and link it to Julia, using the instructions in Top Brass.ipynb.
    1. Gurobi is a high-quality commercial code for linear and integer programming.
    2. You won't need Gurobi in class, but if you are solving large problems in the project, you may find it useful.
5. Experiment! Try changing the parameters and seeing if the solution still makes sense.

Julia Tutorials

- Noteworthy differences between Julia and other languages:
  `https://docs.julialang.org/en/v1/manual/noteworthy-differences/`
- Useful tutorial:
  `https://learnxinyminutes.com/docs/julia/`
- Official Julia documentation:
  `https://docs.julialang.org/en/v1/`
- JuMP: `https://jump.dev/JuMP.jl/stable/`
- More resources on course Canvas Page

Assignment #0

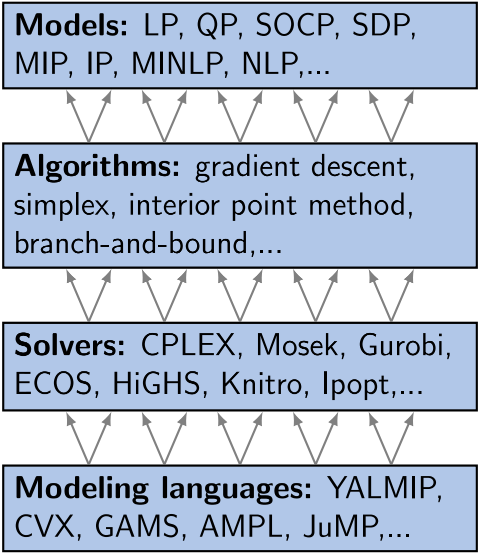
- Get and install Julia and JuMP in your working environment.
- Please post any installation issues on Piazza, so your students colleagues (and the TAs and I) can try to help.
- We have posted some introductory Julia notebook tutorials. You should go through these and following the instructions for turning in JuliaTutorialExercises.ipynb
- You should also ensure that you can correctly print out the notebook as PDF
  - jupyter nbconvert –to pdf notebook.ipynb
  - You may need to install pandoc and/or xetex
- First homework coming soon, so please try to get things working.

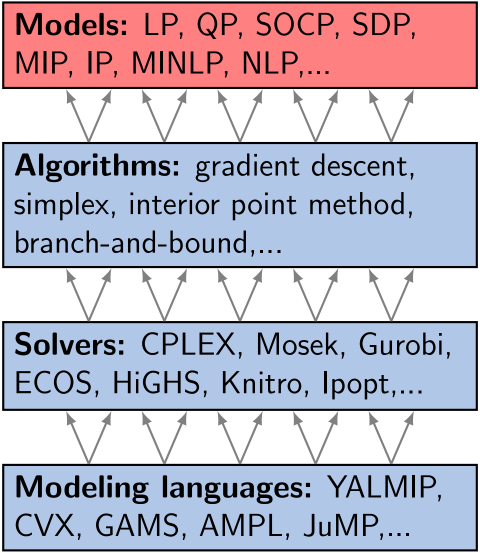
CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024


Models: LP, QP, SOCP, SDP,
MIP, IP, MINLP, NLP,...
Algorithms: gradient descent,
simplex, interior point method,
branch-and-bound,...
Solvers: CPLEX, Mosek, Gurobi,
ECOS, HiGHS, Knitro, Ipopt,...
Modeling languages: YALMIP,
CVX, GAMS, AMPL, JuMP,...



Optimization models can be categorized based on:

- types of variables
- types of constraints
- type of objective

Example: every linear program (LP) has:

- continuous variables
- linear constraints
- a linear objective

We will learn about many other types of models.

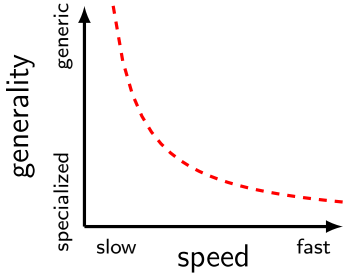
Models: LP, QP, SOCP, SDP, MIP, IP, MINLP, NLP,...

Algorithms: gradient descent, simplex, interior point method, branch-and-bound,...

Solvers: CPLEX, Mosek, Gurobi, ECOS, HiGHS, Knitro, Ipopt,...

Modeling languages: YALMIP, CVX, GAMS, AMPL, JuMP,...

Numerical (usually iterative) procedures that can solve instances of optimization models. More specialized algorithms are usually faster.



Models: LP, QP, SOCP, SDP, MIP, IP, MINLP, NLP,...

Algorithms: gradient descent, simplex, interior point method, branch-and-bound,...

Solvers: CPLEX, Mosek, Gurobi, ECOS, HiGHS, Knitro, Ipopt,...

Modeling languages: YALMIP, CVX, GAMS, AMPL, JuMP,...

Solvers are *implementations* of algorithms. Sometimes they can be quite clever!

- typically implemented in C/C++ or Fortran
- may use sophisticated error-checking, complex heuristics etc.

Availability varies:

- some are open-source
- some are commercial

Models: LP, QP, SOCP, SDP, MIP, IP, MINLP, NLP,...

Algorithms: gradient descent, simplex, interior point method, branch-and-bound,...

Solvers: CPLEX, Mosek, Gurobi, ECOS, HiGHS, Knitro, Ipopt,...

Modeling languages: YALMIP, CVX, GAMS, AMPL, JuMP,...

Modeling languages provide a way to interface with many different solvers using a common language.

- Can be a self-contained language (GAMS, AMPL)
- Some are implemented in other languages (JuMP in Julia, CVX in Matlab, Pyomo in Python)

Again, availability varies:

- some are open-source
- some are commercial

Solvers in JuMP

Before solving a model, you must specify a solver.
You can do this when you declare the model:

```
using JuMP, HiGHS, ECOS, SCS
m = Model(HiGHS.Optimizer)
m = Model(ECOS.Optimizer)
m = Model(SCS.Optimizer)
```

You can also declare a blank model and specify the solver later.

```
m = Model()
set_optimizer(m, HiGHS.Optimizer)
optimize!(m)
set_optimizer(m, ECOS.Optimizer)
optimize!(m)
```

Solvers in JuMP

Before using a solver, you must include the appropriate package:

```
using JuMP, HiGHS
```

Every solver must be installed before it can be used:

```
Pkg.add("HiGHS")
```

Some things to know:

- Installing a package may take a minute or two, but it only has to be done once.
- The first time you use a package after you install or update it, Julia will precompile it. This will take an extra 5–30 sec.
- Keep all your packages up-to-date using `Pkg.update()`

Solvers in JuMP

Top Brass.ipynb

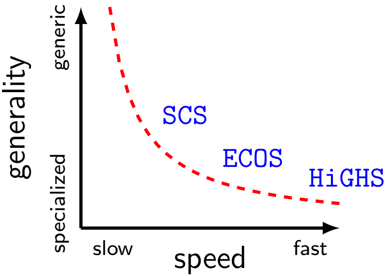
- Try `HiGHS`, `ECOS`, `SCS` solvers. Is the answer the same?
- Compare solvers using the `@time` macro
- What happens if an unsuitable solver is used?

Speed vs Generality

We will see later in the class that these models are nested:

$$\text{LP} \subseteq \text{SOCP} \subseteq \text{SDP}$$

SCS (an SDP solver) is relatively slow at solving LPs because it solves them by first converting them to an SDP!



Writing modular code

It is good practice to separate the *data* from the *model*.

See the code in later cells of Top Brass.ipynb

- Use *arrays* to index over sets
  - `sports = [:football, :soccer]`
- Use *dictionaries* to make the code more modular
  - `wood = Dict( :football => 4, :soccer => 2)`
- Use *expressions* to make the code more readable
  - `@expression(m1, tot_plaques, sum(trophies[i] * plaques[i] for i in sports) )`
- Try adding a new type of trophy!

Comparison: GAMS (1)

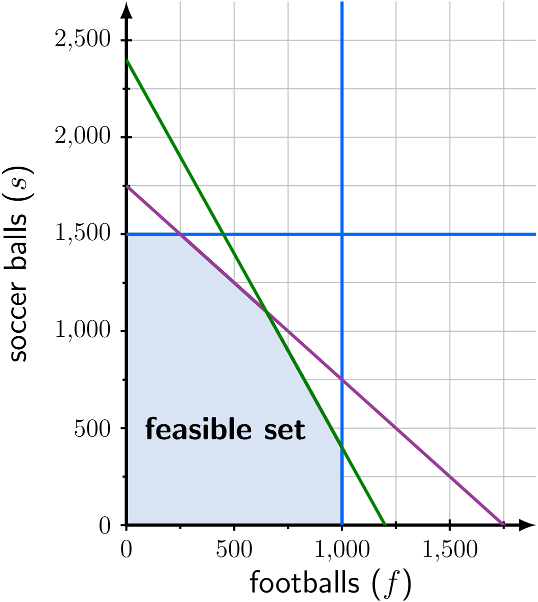
JuMP and GAMS are structurally very similar

```
* TOP BRASS PROBLEM
set I/football, soccer/;
free variable profit "total profit";
positive variables x(I) "trophies";
* DATA section
parameters
         profit(I)   / "football" 12 , "soccer" 9 /
         wood(I)     / "football"  4 , "soccer" 2 /
         plaques(I)  / "football"  1 , "soccer" 1 /;
scalar
         quant_plaques  /1750/
         quant_wood     /4800/
         quant_football /1000/
         quant_soccer   /1500/;
* MODEL section
equations
obj    "max total profit"
foot   "bound on the number of brass footballs used"
socc   "bound on the number of brass soccer balls used",
plaq   "bound on the number of plaques to be used",
wdeq   "bound on the amount of wood to be used";
```

Comparison: GAMS (2)

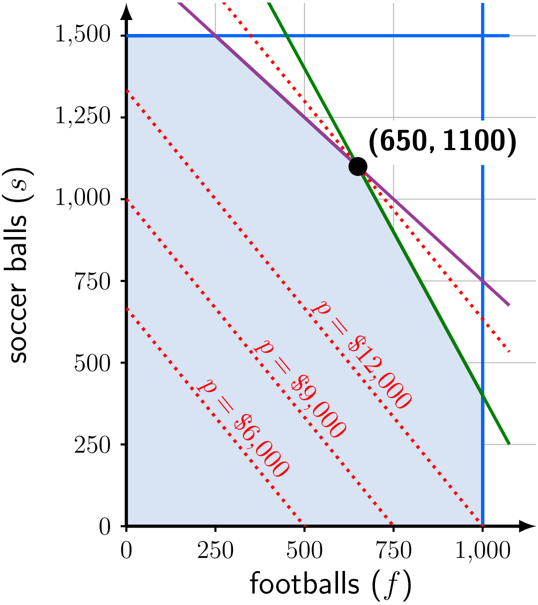
JuMP and GAMS are structurally very similar

```
* CONSTRAINTS
obj..
total_profit =e= sum(I, profit(I)*x(I));

foot..
I("football") =l= quant_football;

socc..
I("soccer") =l= quant_soccer;

plaq..
sum(I,plaques(I)*x(I)) =l= quant_plaques;

wdeq..
sum(I,wood(I)*x(I)) =l= quant_wood;

model topbrass /all/;

* SOLVE
solve topbrass using lp maximizing profit;
```



$$
\begin{aligned}
\max_{f,s} \quad & 12f + 9s \\
\text{s.t.} \quad & 4f + 2s \le 4800 \\
& f + s \le 1750 \\
& 0 \le f \le 1000 \\
& 0 \le s \le 1500
\end{aligned}
$$

Each point (f, s) is a possible decision.



$$\max_{f,\,s} \quad 12f + 9s$$

$$\text{s.t.} \quad 4f + 2s \le 4800$$
$$f + s \le 1750$$
$$0 \le f \le 1000$$
$$0 \le s \le 1500$$

Which feasible point has the max profit?

$$p = 12f + 9s$$

Graphically Solving LP's

- Not for production use, but gives insight into what the algorithm for solving the problem is doing

Identify Feasible Region

- Graph each constraint as an inequality
- Note which side is feasible
- Identify the feasible region: The set of all feasible solutions
- Remember to include nonnegativity!

Graphically Solving LPs

"Move" Objective

- Draw parallel "isoprofit" lines. (All points on each line give the same value of the objective function)
- These are points that are orthogonal to the objective function vector
- Optimal point(s) will be on the highest isoprofit line that touches the feasible region

Observations

Let's Think About Geometry

If there exists an optimal solution to a LP instance, then there exists an optimal solution that exists at an extreme point of the feasible region.

The Simplex Method

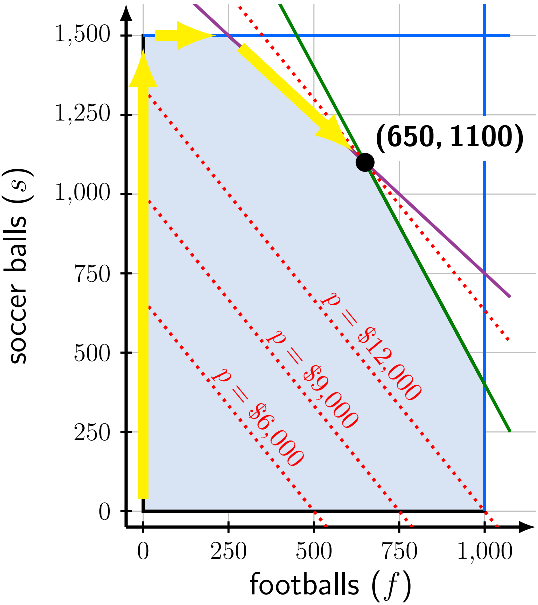
0. Start from an extreme point.
1. Find an improving direction d. If none exists, STOP. The extreme point is an optimal solution.
2. Move along d until you hit a new extreme point. Go to 1.

The Simplex Method

- The simplex method is a systematic way in which to do the algebra necessary to do steps 0, 1, and 2.

Some definitions and facts:

- An inequality $a^T x \leq b$ is *binding* at x if $a^T x = b$.
- An extreme point is the intersection of at least n inequalities in $\mathbb{R}^n$.
- *Basis* : The indices of the n inequalities that are "binding" at an extreme point solution. (The solution itself is sometimes called a basic feasible solution).



$$\max_{f,\, s} \quad 12f + 9s$$
$$\text{s.t.} \quad 4f + 2s \le 4800$$
$$f + s \le 1750$$
$$0 \le f \le 1000$$
$$0 \le s \le 1500$$

- Walk $(0,0) \to (0,1500)$
- Walk $(0,1500) \to (250,1500)$
- Walk $(250,1500) \to (650,1100)$

Next class...

- Get Julia and JuMP Running in your environment
- Be sure you can print PDF of notebooks (Homework 0)
- Julia/IJulia/JuMP tutorial

Next Time

- Review of matrix math
- Geometry of general linear programs
- LP standard forms and transformations

CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

Matrix basics

A matrix is an array of numbers. $A \in \mathbb{R}^{m \times n}$ means that:

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \quad (m \text{ rows and } n \text{ columns})$$

Two matrices can be multiplied if inner dimensions agree:

$$\underset{(m\times p)}{C} = \underset{(m\times n)}{A}\,\underset{(n\times p)}{B} \quad \text{where} \quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 8 & 9 \end{bmatrix} = \begin{bmatrix} 1\cdot 4 + 2\cdot 8 & 1\cdot 3 + 2\cdot 9 \\ 3\cdot 4 + 4\cdot 8 & 3\cdot 3 + 4\cdot 9 \\ 5\cdot 4 + 6\cdot 8 & 5\cdot 3 + 6\cdot 9 \end{bmatrix} = \begin{bmatrix} 20 & 21 \\ 44 & 45 \\ 68 & 69 \end{bmatrix}$$

Matrix basics

A matrix is an array of numbers. $A \in \mathbb{R}^{m \times n}$ means that:

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \quad (m \text{ rows and } n \text{ columns})$$

Two matrices can be multiplied if inner dimensions agree:

$$\underset{(m\times p)}{C} = \underset{(m\times n)}{A}\,\underset{(n\times p)}{B} \quad \text{where} \quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 8 & 9 \end{bmatrix} = \begin{bmatrix} 1\cdot 4 + 2\cdot 8 & 1\cdot 3 + 2\cdot 9 \\ 3\cdot 4 + 4\cdot 8 & 3\cdot 3 + 4\cdot 9 \\ 5\cdot 4 + 6\cdot 8 & 5\cdot 3 + 6\cdot 9 \end{bmatrix} = \begin{bmatrix} 20 & 21 \\ 44 & 45 \\ 68 & 69 \end{bmatrix}$$

Matrix basics

Transpose: The transpose operator A^T swaps rows and columns. If $A \in \mathbb{R}^{m\times n}$ then $A^\mathsf{T} \in \mathbb{R}^{n\times m}$ and $(A^\mathsf{T})_{ij} = A_{ji}$.

- $(A^\mathsf{T})^\mathsf{T} = A$
- $(AB)^\mathsf{T} = B^\mathsf{T} A^\mathsf{T}$

A vector is a column matrix. We write $x \in \mathbb{R}^n$ to mean that:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad \text{(a vector } x \in \mathbb{R}^n \text{ is an } n \times 1 \text{ matrix)}$$

The transpose of a column vector is a row vector:

$$x^\mathsf{T} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \qquad \text{(i.e. a } 1 \times n \text{ matrix)}$$

Matrix basics

Two vectors $x, y \in \mathbb{R}^n$ can be multiplied together in two ways. Both are valid matrix multiplications:

- **inner product**: produces a scalar.

$$x^{\mathsf{T}} y = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = x_1 y_1 + \cdots + x_n y_n$$

  Also called "dot product". Often written $x \cdot y$ or $\langle x, y \rangle$.

- **outer product**: produces an $n \times n$ matrix.

$$x y^{\mathsf{T}} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & \dots & x_1 y_n \\ \vdots & \ddots & \vdots \\ x_n y_1 & \dots & x_n y_n \end{bmatrix}$$

Matrix basics

- Matrices and vectors can be stacked and combined to form bigger matrices as long as the dimensions agree. e.g. If $x_1, \ldots, x_m \in \mathbb{R}^n$, then $X = \begin{bmatrix} x_1 & x_2 & \ldots & x_m \end{bmatrix} \in \mathbb{R}^{m \times n}$.

- Matrices can also be concatenated in blocks. For example:

$$Y = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

if A, C have same number of columns, A, B have same number of rows, etc.

- Matrix multiplication also works with block matrices!

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} AP + BQ \\ CP + DQ \end{bmatrix}$$

as long as A has as many columns as P has rows, etc.

Linear and affine functions

- A function $f(x_1, \ldots, x_m)$ is *linear* in the variables $x_1, \ldots, x_m$ if there exist constants $a_1, \ldots, a_m$ such that

$$f(x_1, \ldots, x_m) = a_1 x_1 + \cdots + a_m x_m = a^{\mathsf{T}} x$$

- A function $f(x_1, \ldots, x_m)$ is *affine* in the variables $x_1, \ldots, x_m$ if there exist constants $b, a_1, \ldots, a_m$ such that

$$f(x_1, \ldots, x_m) = a_0 + a_1 x_1 + \cdots + a_m x_m = a^{\mathsf{T}} x + b$$

Examples:

1. $3x - y$ is linear in (x, y).
2. $-6x + 7y - 1$ is affine in (x, y).
3. $x^2 + y^2$ is not linear or affine.

- N.B.: Some texts use linear and affine interchangeably

Linear and affine functions

Several linear or affine functions can be combined:

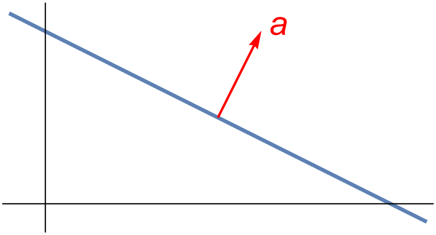
$$\begin{matrix} a_{11}x_1 + \cdots + a_{1n}x_n + b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n + b_2 \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n + b_m \end{matrix} \implies \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

which can be written simply as $Ax + b$. Same definitions apply:

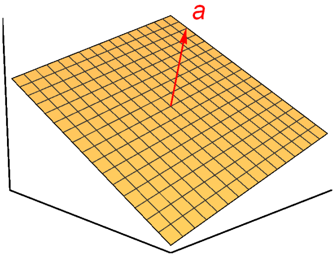
- A vector-valued function $F(x)$ is *linear* in x if there exists a constant matrix A such that $F(x) = Ax$.
- A vector-valued function $F(x)$ is *affine* in x if there exists a constant matrix A and vector b such that $F(x) = Ax + b$.

Geometry of affine equations

- The set of points $x \in \mathbb{R}^n$ that satisfies a linear equation $a_1x_1 + \cdots + a_nx_n = 0$ (or $a^\mathsf{T}x = 0$) is called a *hyperplane*. The vector a is *normal* to the hyperplane.
- If the right-hand side is nonzero: $a^\mathsf{T}x = b$, the solution set is called an *affine hyperplane*, (it's a shifted hyperplane).



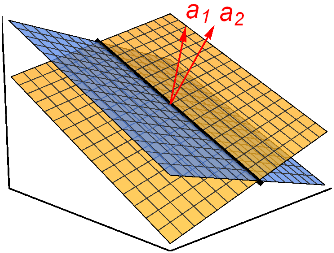
Affine hyperplane in 2D



Affine hyperplane in 3D

Geometry of affine equations

- The set of points $x \in \mathbb{R}^n$ satisfying many linear equations $a_{i1}x_1 + \cdots + a_{in}x_n = 0$ for $i = 1, \ldots, m$ (or $Ax = 0$) is called a *subspace* (the intersection of many hyperplanes).
- If the right-hand side is nonzero: $Ax = b$, the solution set is called an *affine subspace*, (it's a shifted subspace).



Intersections of affine hyperplanes are affine subspaces.

Geometry of affine equations

The *dimension* of a subspace is the number of independent directions it contains: the *size of the largest set of linearly independent vectors* in the subspace.

A line has dimension 1, a plane has dimension 2, and so on.

Hyperplanes are subspaces!

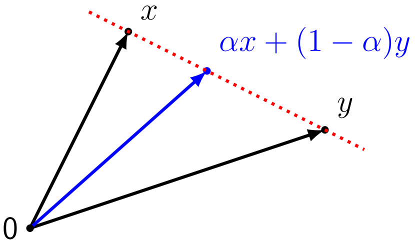
- A hyperplane in $\mathbb{R}^n$ is a subspace of dimension $n-1$.
- The intersection of k hyperplanes has dimension at least $n-k$ ("at least" because of potential redundancy).

Affine combinations

If $x, y \in \mathbb{R}^n$, then the combination

$$w = \alpha x + (1-\alpha)y \quad \text{for some } \alpha \in \mathbb{R}$$

is called an *affine combination*.



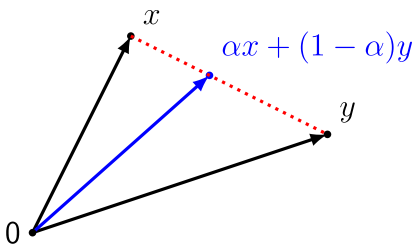
If $Ax = b$ and $Ay = b$, then $Aw = b$. So affine combinations of points in an (affine) subspace also belong to the subspace.

Convex combinations

If $x, y \in \mathbb{R}^n$, then the combination

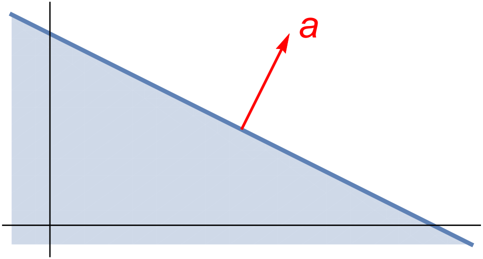
$$w = \alpha x + (1-\alpha)y \quad \text{for some } 0 \leq \alpha \leq 1$$

is called a *convex combination* (for reasons we will learn later). It's the line segment that connects x and y.



Geometry of affine inequalities

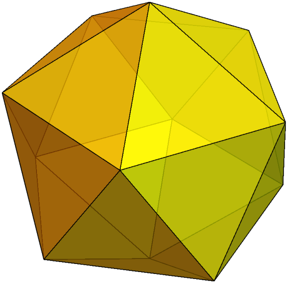
- The set of points $x \in \mathbb{R}^n$ that satisfies a linear inequality $a_1x_1 + \cdots + a_nx_n \leq b$ (or $a^\mathsf{T}x \leq b$) is called a *halfspace*. The vector a is *normal* to the halfspace and b shifts it.
- Define $w = \alpha x + (1-\alpha)y$ where $0 \leq \alpha \leq 1$. If $a^\mathsf{T}x \leq b$ and $a^\mathsf{T}y \leq b$, then $a^\mathsf{T}w \leq b$.



Halfspace

Geometry of affine inequalities

- The set of points $x \in \mathbb{R}^n$ satisfying many linear inequalities $a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i$ for $i = 1, \ldots, m$ (or $Ax \leq b$) is called a *polyhedron* (the intersection of many halfspaces). Some sources use the term *polytope* instead.
- As before: let $w = \alpha x + (1 - \alpha)y$ where $0 \leq \alpha \leq 1$. If $Ax \leq b$ and $Ay \leq b$, then $Aw \leq b$.



Intersections of halfspaces are polyhedra.

The linear program

A linear program is an optimization model with:

- real-valued variables ($x \in \mathbb{R}^n$)
- affine objective function ($c^\top x + d$), can be min or max.
- constraints may be:
  - affine equations ($Ax = b$)
  - affine inequalities ($Ax \leq b$ or $Ax \geq b$)
  - combinations of the above
- individual variables may have:
  - box constraints ($p_i \leq x_i$, or $x_i \leq q_i$, or $p_i \leq x_i \leq q_i$, where p_i and q_i are parameters, not variables)
  - no constraints (x_i is unconstrained)

There are many equivalent ways to express the same LP

Standard form

- Every LP can be put in the form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{maximize}} \quad & c^\mathsf{T} x \\ \text{subject to:} \quad & Ax \le b \\ & x \ge 0 \end{aligned}$$

- We'll call this the *standard form* of a LP.
- (Unfortunately, there are multiple definitions of "standard form" but let's use this one for purposes of this class.)

Back to Top Brass

$$\begin{aligned} \max_{f,s} \quad & 12f + 9s \\ \text{s.t.} \quad & 4f + 2s \leq 4800 \\ & f + s \leq 1750 \\ & 0 \leq f \leq 1000 \\ & 0 \leq s \leq 1500 \end{aligned}$$

$$\Longrightarrow$$

$$\begin{aligned} \max_{f,s} \quad & \begin{bmatrix} 12 \\ 9 \end{bmatrix}^\top \begin{bmatrix} f \\ s \end{bmatrix} \\ \text{s.t.} \quad & \begin{bmatrix} 4 & 2 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f \\ s \end{bmatrix} \leq \begin{bmatrix} 4800 \\ 1750 \\ 1000 \\ 1500 \end{bmatrix} \\ & \begin{bmatrix} f \\ s \end{bmatrix} \geq 0 \end{aligned}$$

This is in standard form, with:

$$A = \begin{bmatrix} 4 & 2 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 4800 \\ 1750 \\ 1000 \\ 1500 \end{bmatrix}, \quad c = \begin{bmatrix} 12 \\ 9 \end{bmatrix}, \quad x = \begin{bmatrix} f \\ s \end{bmatrix}$$

Transformation tricks

1. converting min to max or vice versa (take the negative):
$$\min_x f(x) = -\max_x (-f(x))$$

2. reversing inequalities (flip the sign):
$$Ax \leq b \quad \iff \quad (-A)x \geq (-b)$$

3. equalities to inequalities (double up):
$$f(x) = 0 \quad \iff \quad f(x) \geq 0 \quad \text{and} \quad f(x) \leq 0$$

4. inequalities to equalities (add slack):
$$f(x) \leq 0 \quad \iff \quad f(x) + s = 0 \quad \text{and} \quad s \geq 0$$

Transformation tricks

5. unbounded to bounded (add difference):

$$x \in \mathbb{R} \quad \Longleftrightarrow \quad u \geq 0, \quad v \geq 0, \quad \text{and} \quad x = u - v$$

6. bounded to unbounded (convert to inequality):

$$p \leq x \leq q \quad \Longleftrightarrow \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} x \leq \begin{bmatrix} q \\ -p \end{bmatrix}$$

7. bounded to nonnegative (shift the variable)

$$p \leq x \leq q \quad \Longleftrightarrow \quad 0 \leq (x - p) \quad \text{and} \quad (x - p) \leq (q - p)$$

More complicated example

Convert the following LP to standard form:

$$
\begin{aligned}
\underset{p,q}{\text{minimize}} \quad & p + q \\
\text{subject to:} \quad & 5p - 3q = 7 \\
& 2p + q \geq 2 \\
& 1 \leq q \leq 4
\end{aligned}
$$

notebook: Standard Form.ipynb

Example

Equivalent LP (standard form):

$$
\begin{aligned}
\underset{u,v,w}{\text{maximize}} \quad & -u + v - w \\
\text{subject to:} \quad & -5u + 5v + 3w \leq -10 \\
& 5u - 5v - 3w \leq 10 \\
& -2u + 2v - w \leq -1 \\
& w \leq 3 \\
& u, v, w \geq 0
\end{aligned}
$$

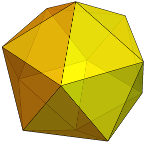
where: $p := u - v, \quad q := w + 1$

and: (original cost) $= -$(new cost) $+ 1$

LPs and polyhedra

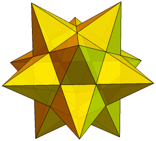
Linear programs have polyhedral feasible sets:

$$\{x \mid Ax \le b\} \qquad \Longrightarrow$$



Can every polyhedron be expressed as $Ax \le b$?

Not this one...

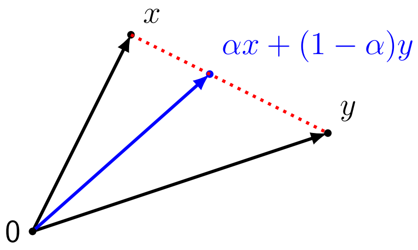


LPs and polyhedra

If $x, y \in \mathbb{R}^n$, then the linear combination

$$w = \alpha x + (1-\alpha)y \quad \text{for some } 0 \leq \alpha \leq 1$$

is called a *convex combination*. As we vary α, it traces out the line segment that connects x and y.



LPs and polyhedra

Note that when we say that $c \leq d$ where c and d are two *vectors* of the same dimension, we mean that *every component of c is less than or equal to the corresponding component of d*.

We can have vectors for which neither $c \leq d$ nor $c \geq d$ is true!

> If $Ax \leq b$ and $Ay \leq b$, and w is a convex combination of x and y, then $Aw \leq b$.

Proof: Suppose $w = \alpha x + (1-\alpha)y$.

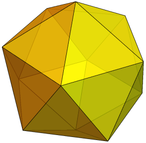
$$\begin{aligned} Aw &= A\left(\alpha x + (1-\alpha)y\right) \\ &= \alpha Ax + (1-\alpha)Ay \\ &\leq \alpha b + (1-\alpha)b = b \end{aligned}$$

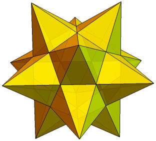
Therefore, $Aw \leq b$, which is what we were trying to prove.

LPs and polyhedra

The previous result implies that every polyhedron describable as $Ax \leq b$ must contain all convex combinations of its points.

- Such polyhedra are called *convex*.
- Informal definition: if you were to "shrink-wrap" it, the entire polyhedron would be covered with no extra space.

Convex: 

Not convex: 

Goes the other way too: every convex polyhedron can be represented as $Ax \leq b$ for appropriately chosen A and b.

Next...

- General modeling
- Cases of LP
- Start working on homework 1!

CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

You Deserve a Break Today

- We're hungry! —Let's determine how many of the following items to eat in order to meet our daily nutritional requirements.

Mmmmmmmmmmm.

- QP: Quarter Pounder
- MD: McLean Deluxe
- BM: Big Mac
- FF: Filet-O-Fish
- MC: McGrilled Chicken
- FR: Small Fries
- SM: Sausage McMuffin
- 1M: 1% Milk
- OJ: Orange Juice

Nutrients

- Prot: Protein
- VitA: Vitamin A
- VitC: Vitamin C
- Calc: Calcium
- Iron: Iron
- Cals: Calories
- Carb: Carbohydrates

Data

	QP	MD	BM	FF	MC	FR	SM	1M	OJ	Req'd
Cost	1.84	2.19	1.84	1.44	2.29	0.77	1.29	0.6	0.72	
Prot	28	24	25	14	31	3	15	9	1	55
VitA	15	15	6	2	8	0	4	10	2	100
VitC	6	10	2	0	15	15	0	4	120	100
Calc	30	20	25	15	15	0	20	30	2	100
Iron	20	20	20	10	8	2	15	0	2	100
Cals	510	370	500	370	400	220	345	110	80	2000
Carb	34	33	42	38	42	26	27	12	20	350

Elements of an Optimization

Variables

- What are we trying to decide?
- How many of each item to eat.
- Let x_j: Be the number of item j to eat.
  - (e.g. x_{QP}: Number of quarter pounders).

Objective

- Let's minimize our cost
- But how much does a daily menu cost?

Costing

- So if I bought my regular lunch: 3 quarter pounders, 2 small fries, and a 1% milk, my cost would be

$$3(1.84) + 2(1.44) + 1(0.6) = \$9.00$$

- A general expression for my cost as a function of my decision on what to buy is

$$\begin{aligned} 1.84x_{QP} + 2.19x_{MD} + 1.84x_{BM} + 1.44x_{FF} + 2.29x_{MC} \\ + 0.77x_{FR} + 1.29x_{SM} + 0.6x_{1M} + 0.72x_{OJ} \end{aligned}$$

- This is our linear objective function

Nag, Nag, Nag :-)

- My wife tells me that I need to get 100% of my daily nutritional requirements from eating at McGreasy's
- A general expression for the daily amount of Vitamin A that I get by eating at McGreasy's is[2]

$$15x_{QP} + 15x_{MD} + 6x_{BM} + 2x_{FF} + 8x_{MC} + 4x_{SM} + 10x_{1M} + 2x_{OJ}$$

- In general I need that

$$15x_{QP} + 15x_{MD} + 6x_{BM} + 2x_{FF} + 8x_{MC} + 4x_{SM} + 10x_{1M} + 2x_{OJ} \geq 100$$

- You can write similar constraints for each nutrient:

[2] I could eat 50 Filet-O-Fish to get my Vitamin A requirements!

The Final Model (1 of 3)

minimize

$$1.84x_{QP} + 2.19x_{MD} + 1.84x_{BM} + 1.44x_{FF} + 2.29x_{MC} + 0.77x_{FR} + 1.29x_{SM} + 0.6x_{1M} + 0.72x_{OJ}$$

subject to

Protein: $$28x_{QP} + 24x_{MD} + 25x_{BM} + 14x_{FF} + 31x_{MC} + 3x_{FR} + 15x_{SM} + 9x_{1M} + x_{OJ} \geq 55$$

Vitamin A: $$15x_{QP} + 15x_{MD} + 6x_{BM} + 2x_{FF} + 8x_{MC} + 4x_{SM} + 10x_{1M} + 2x_{OJ} \geq 100$$

Final McGreasy's Model (2 of 3)

Vitamin C: $$6x_{QP} + 10x_{MD} + 2x_{BM} + 15x_{MC} + 15x_{FR} + 4x_{1M} + 120x_{OJ} \geq 100$$

Calcium: $$30x_{QP} + 20x_{MD} + 25x_{BM} + 15x_{FF} + 15x_{MC} + 20x_{SM} + 30x_{1M} + 2x_{OJ} \geq 100$$

Iron: $$20x_{QP} + 20x_{MD} + 20x_{BM} + 10x_{FF} + 8x_{MC} + 2x_{FR} + 15x_{SM} + 2x_{OJ} \geq 100$$

Final McGreasy's Model (3 of 3)

Calories: $$\begin{aligned} 510x_{QP} + 370x_{MD} + 500x_{BM} + 370x_{FF} + 400x_{MC} \\ + 220x_{FR} + 345x_{SM} + 110x_{1M} + 80x_{OJ} \geq 2000 \end{aligned}$$

Carbs: $$\begin{aligned} 34x_{QP} + 35x_{MD} + 42x_{BM} + 38x_{FF} + 42x_{MC} + 26x_{FR} \\ + 27x_{SM} + 12x_{1M} + 20x_{OJ} \geq 350 \end{aligned}$$

$$x_{QP}, x_{MD}, x_{BM}, x_{FF}, x_{MC}, x_{FR}, x_{SM}, x_{1M}, x_{OJ} \geq 0$$

Check Out The Notebook

McDonaldsDiet.ipynb

- Use of `Dict(zip(indexList,ValuesList))` to create indexed parameters
- Use of `NamedArrays` package to allow array to be indexed by element names, not by number
- `(m, [i in nutrients], sum(A_NA[i,j]*x[j] for j in foods) >= required[i])` creates one constraint for every element in `nutrients`

The Sets View—A General Model

Sets

- F: Set of possible foods
- N: Set of nutrional requirements

Parameters

- c_j: Per unit cost of item $j \in F$
- ℓ_i: Lower Bound on amount of nutrient $i \in N$
- u_i: Upper Bound on amount of nutrient $i \in N$
- a_{ij}: Amount of nutrient $i \in N$ in food $j \in F$

The Diet Problem

$$\min \sum_{j \in F} c_j x_j$$

$$\begin{aligned} \ell_i \le \sum_{j \in F} a_{ij} x_j &\le u_i \qquad \forall i \in N \\ x_j &\ge 0 \qquad \forall j \in F \end{aligned}$$

$$\min_{x \in \mathbb{R}_+^{|F|}} \{c^T x \mid \ell \le Ax \le u\}$$

Check Out The Notebook

McDonaldsDiet-CSV.ipynb

- Uses julia `DataFrames`, like Pandas in python, or R Data Data Frames
- Uses julia `CSV` to read the CSV file into a dataframe
- Extract sets and parameters from dataframe, put into Dictionaries
- Extract 'A' matrix from dataframe, then put it into a `NamedArray`
- Code solving model is exact same: If `mcdonalds.csv` had 10,000 rows and columns, it would just solve a bigger problem!

Recall the Simplex Algorithm

The Simplex Method

0. Start from an extreme point.
1. Find an improving direction d. If none exists, STOP. The extreme point is an optimal solution.
2. Move along d until you hit a new extreme point. Go to 1.

Simplex Method – What can go wrong?

Simplex Method: Step 2

Move along d until you hit a new extreme point.

- What if we don't hit an extreme point?

$$\max x_1 + x_2$$

$$\begin{array}{lrcl} \text{s.t.} & x_1 + 2x_2 & \geq & 1 \\ & x_1, x_2 & \geq & 0 \end{array}$$

- Usually this means you forgot some constraints. Maybe your variable bounds?
- N.B.: Just because the region is unbounded doesn't mean that the LP is unbounded.

I Will Glady Pay You Tuesday...

- I really like hamburgers.
- Let's suppose in the diet problem, I decide to maximize the number of hamburgers I eat
- Let $B \subset F$

$$B = \{\text{QP, MD, BM}\}$$

- My new objective is to

$$\max \sum_{j \in B} x_b$$

- McDonaldsDiet-LPCases.ipynb

Mmmmmmmmmm. Beef

- Always check the Model status in the solution report

```
Model   status       : Unbounded
Simplex   iterations: 3
Objective value      :  5.0000000000e+01
HiGHS run time       :          0.00
Maximum Number Hamburgers 0.50:
Eat 0.50 of menu item :BM
```

- The Model status is unbounded!

```
stat = termination_status(m)
if stat != MOI.OPTIMAL
println("Solver did not find optimal solution, status:
", stat)
end
```

Simplex Method – What can go wrong?

Simplex Method: Step 0

Start from an extreme point

- What if there *are* no extreme points?
  - This (usually) means that the feasible region is empty.
  - The instance is infeasible.
  - $P = \{x \in \mathbb{R}^2 : x_1 + x_2 \leq 1, x_1 + x_2 \geq 2\}$
- How will we know if an instance is infeasible?
  - "Big-M", "Two-Phase"?
  - The solver will tell us!

Warning!

- It may be hard to "blame" one constraint for being infeasible.
- When building models for the real world determining what is "causing" the infeasibility may be tough.
- Whose "fault" is this?

$$x_1 - x_2 \geq 1, x_2 - x_3 \geq 1, -x_1 + x_3 \geq 1$$

My Wife Loves Me!

- In the interest of extending my life, Helen has requested that I obey the following constraints:

1. Don't eat more than 3 sandwiches per day

$$x_{QP} + x_{MD} + x_{BM} + x_{FF} + x_{MC} + x_{SM} \leq 3$$

2. Don't drink too much: $x_{1M} + x_{OJ} \leq 3$
3. Only two french fries per day: $x_{FF} \leq 2$

- But with these constraints, the problem is infeasible!

```
{Model   status      : Infeasible
Simplex   iterations: 5
Objective value     :  2.4751250000e+01}
```

Handling Infeasibility

Our First Trick

- Introduce slack/surplus variables and try to minimize the slack/surplus.

- Suppose I think that the "too much drinking" constraint is the one causing the problem to be infeasible
- New decision variable s: Number of extra drinks (over three) that I must drink in order to get a feasible solution

$$x_{1M} + x_{OJ} - s \leq 3, s \geq 0$$

- New Objective: $\min s$

- Be sure to go through McDonaldsDiet-LPCases.ipynb

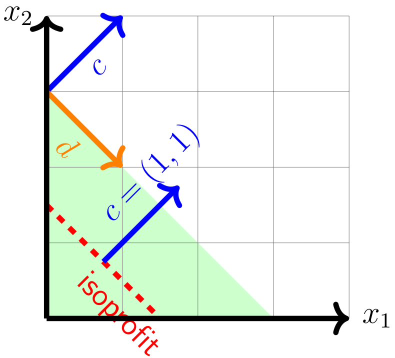
Multiple Optimal Solutions

- What if c is orthogonal to an "improving" direction d? (Rate of change 0)

maximize

$$x_1 + x_2$$

$$\begin{aligned} x_1 + x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

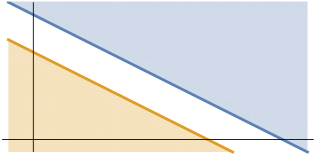


- We get an infinite number of optimal solutions.
- Every point that is a convex combination of the extreme points of the optimal face is also optimal

Solutions of an LP

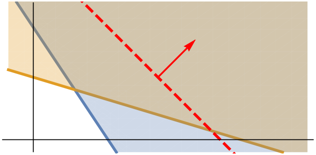
There are exactly three possible cases:

1. Model is *infeasible*: there is no x that satisfies all the constraints. (is the model correct?)



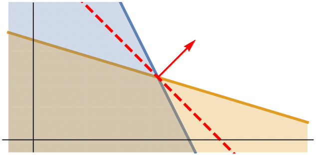
infeasible

2. Model is feasible, but *unbounded*: the cost function can be arbitrarily improved. (forgot a constraint?)



unbounded

3. Model has a solution which occurs *on the boundary* of the set. (there may be many solutions!)



CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

Blending Constraints

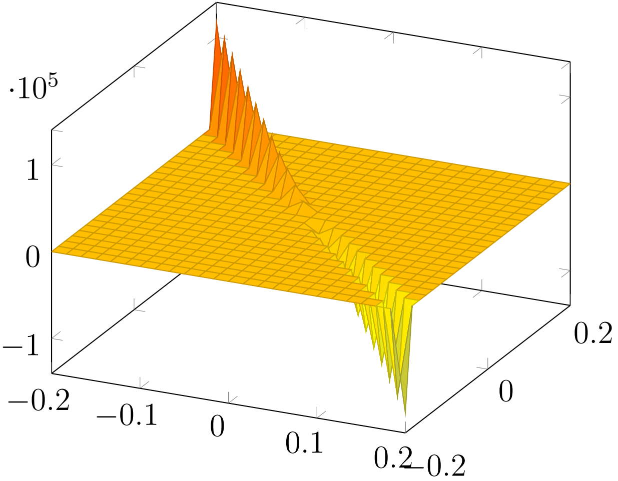
- Back to McGreasy's — Imagine that Helen has "relaxed" her constraint on my hamburger intake.
- Now, I can eat as many hambugers as I want, with two new requirements:
  1. We now have some maximum amount of every nutrient (say three times the minimum requirement)
  2. Keep my calories to a specified percentage of my vitamin intake:

$$\frac{\sum_{j \in F} a_{Cals,f} x_f}{\sum_{j \in F} a_{VitC,f} x_f} \leq \rho$$

$$\frac{\sum_{j \in F} a_{Cals,f} x_f}{\sum_{j \in F} a_{VitA,f} x_f} \leq \rho$$

- Is this a linear constraint?

NO! : $\frac{2x_1+x_2}{x_1+x_2}$



Solving with HiGHS

```
Constraints of type
MathOptInterface.ScalarNonlinearFunction-in-MathOptInterface.
are not supported by the solver.

If you expected the solver to support your problem, you may
have an error in your formulation. Otherwise,
consider using a different solver.

The list of available solvers, along with the problem
types they support, is available at
https://jump.dev/JuMP.jl/stable/installation/#Supported-solve
```

Making the Nonlinear Into Linear

- By doing some algebra, we can write the set of points satisfying this (nonlinear) inequality as a linear inequality...
- Multiply both sides of the inequality by $\sum_{j \in F} a_{VitC,f} x_f$
- What (very important) assumption did I just make?
- $\sum_{j \in F} a_{VitC,f} x_f > 0$ in any feasible solution!
- The moral of the story...
  - Not everything that looks nonlinear is nonlinear
- This is called a "blending" constraint.

Making Alloy

- We would like to make an amount d of a specific alloy
- There is a set E of elements
- For each element $e \in E$, there is both a minimum (%) grade ℓ_e and a maximum (%) grade (%) u_e that the alloy must have.
- Alloy is made from a set R of raw materials, each costing c_r per unit and having a maximum amount K_r available $(\forall r \in R)$
- Raw material r is made up up α_{re} percent of element $e \in E$

Assumption: Production is "linear'

- All raw materials converted into alloy
- Final alloy element percentages is weighted average of element composition of input raw materials

Math Model

- x_r : Amount of raw r to produce

$$\min \sum_{r \in R} c_r x_r$$

$$\sum_{r \in R} (\alpha_{re} - \ell_e) x_r \geq 0 \quad \forall e \in E$$

$$\sum_{r \in R} (\alpha_{re} - u_e) x_r \leq 0 \quad \forall e \in E$$

$$\sum_{r \in R} x_r \geq d$$

$$0 \leq x_r \leq K_r \qquad \forall r \in R$$

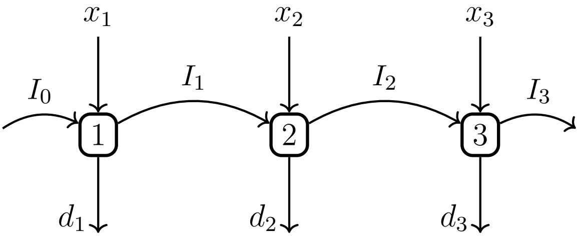
Check the Notebook

- Alloy.ipynb

Modeling Multi-Period Problems

- One of the most important uses of optimization is in multi-period planning.
- Partition time into a number of periods.
- Usually distinguished by Inventory or Carry-Over variables.
- Suppose there is a "planning horizon" $T = \{1, 2, \ldots, |T|\}$.
- Also suppose there is a known demand d_t for each $t \in T$
- Define...
  - x_t : Production level during period $t, \forall t \in T$
  - I_t : Inventory level at end of period $t, \forall t \in T$

Modeling Multi-Period Problems



$$I_0 + x_1 = d_1 + I_1$$
$$I_1 + x_2 = d_2 + I_2$$
$$I_{t-1} + x_t = x_t + I_t$$

- To model "losses or gains", just put appropriate multipliers (not 1) on the arcs

Another Story: Aggregate Planning

- Complex production process involving many pieces
  - Demands
  - Variable workforce size
  - Overtime possibilities
  - Inventory requirements

We're Making Shoes: ShoeCo

- Plan production of shoes for next several months
- Meet forecast demands on time
- Hire and/or lay off workers
- Make overtime decisions
- Objective: minimize total cost

ShoeCo: It's All Greek To Me

- Planning horizon $T = \{1, 2, \ldots |T|\}$. ($|T| = 4$).
- Meet demand d_t for shoes in period $t \in T$.
  $d = (3000, 5000, 2000, 1000)$
- Initial Shoe Inventory: $\mathcal{I}_0 = 500$
- Have $\mathcal{W}_0 = 100$ workers currently employed
- Workers paid $\$\alpha = 1500$/month for working $H = 160$ hours
- They can work overtime (max of $O = 20$ hours/worker) and get paid $\$\beta = 13$/hour.

ShoeCo: Greek Letter Zoo

- It take $a = 4$ hours of labor and $\delta = \$15$ in raw material costs to produce a shoe
- Hire-Fire costs: $\eta = 1600$ to hire a worker, $\zeta = \$2000$ to fire a worker.
- Running out of greek letters, $\iota = \$3$ holding cost incurred for each pair of shoes held at the end of the month.
  - Inventory costs are sometime compuer as cost of capital—You could better invest your money rather than having that investment tied up in produced inventory

Your Mission

- Minimize all costs: labor (regular + overtime), production, inventory, hiring and firing
- What decision variables do we need?
  - HINT: If you're having trouble getting the decision variables, try and write the objective

---

Decision Variables

- x_t: # of shoes to produce during month t
- I_t: Ending inventory in month t, $t \in T \cup \{0\}$
- w_t: # of workers available in month t, $t \in T \cup \{0\}$.
- o_t: # of overtime hours used in month t
- h_t: # workers hired at the beginning of month t
- f_t: # workers fired at the beginning of month t

Objective, Minimize Total Costs

1. Raw Material Costs: $\sum_{t \in T} \delta x_t$
2. Regular Labor Costs: $\sum_{t \in T} \alpha w_t$
3. Overtime Labor Costs: $\sum_{t \in T} \beta o_t$
4. Hiring Costs: $\sum_{t \in T} \eta h_t$
5. Firing Costs: $\sum_{t \in T} \zeta f_t$
6. Inventory Costs: $\sum_{t \in T} \iota I_t$

Constraints

Limit on Monthly Production

- Not given explicitly
- Determined by number of workers available and overtime decisions
- Math-speak: $ax_t \leq Hw_t + o_t \quad \forall t \in T$

Upper limit on overtime hours/month

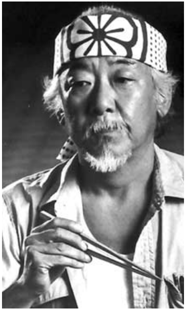
- Depends on how many workers you have
- Aggregate planning: Don't worry about individual workers
- Math-speak: $o_t \leq Ow_t \quad \forall t \in T$

Constraints

Demand must be met on time

- Equivalent to having nonnegative ending inventory each month (no backlogging)
- Math-speak: $I_t \geq 0 \quad \forall t \in T$
- This assumes we have balance between production, demand, and inventory
- We'll see backlogging later

Balance, Daniel-Son



Shoes

- Draw Picture, Math Speak:

$$I_{t-1} + x_t = d_t + I_t \quad \forall t \in T$$

- Boundary: $I_0 = \mathcal{I}_0$ (Maybe $I_{|T|} \geq \mathcal{I}_0$).

People

- Hiring/Firing Affects worker levels. Math speak:

$$w_t = w_{t-1} + h_t - f_t \quad \forall t \in T$$

- Boundary: $w_0 = \mathcal{W}_0$

Full Model

$$\min \sum_{t \in T} (\delta x_t + \alpha w_t + \beta o_t + \eta h_t + \zeta f_t + \iota I_t)$$

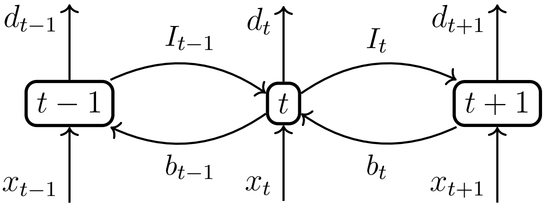
$$\begin{array}{rcl}
\text{s.t. } a x_t & \leq & H w_t + o_t \quad \forall t \in T \\
o_t & \leq & O w_t \quad \forall t \in T \\
I_{t-1} + x_t & = & d_t + I_t \quad \forall t \in T \\
I_0 & = & \mathcal{I}_0 \\
w_t & = & w_{t-1} + h_t - f_t \quad \forall t \in T \\
w_0 & = & \mathcal{W}_0 \\
x_t, I_t, w_t, h_t, f_t & \geq & 0 \quad \forall t \in T
\end{array}$$

- Check out the notebook ShoeCo.ipynb

Stuff Happens

- Suppose you don't have to meet forecast demands in every period.
- Meeting demand is often too stringent a requirement for the real-world
- Demand does not have to be met on time, but it must be met *eventually*
- There is a shortage cost $\theta = \$20$ per unit per month backlogged
- \$1 Question: How should the minimum cost compare with cost of earlier model?

Backlog model: Revised inventory balance



- Interpretation: b_t represents a flow from the future to the current period
- New inventory balance constraints, for $t = 1, \ldots, T$

$$I_{t-1} + b_t + x_t = d_t + I_t + b_{t-1}$$

- Backlog variables also have the sign restriction:

$$b_t \geq 0, \quad t = 1, \ldots, T$$

Problem with model?

In our model, it is feasible to have both $b_t > 0$ and $I_t > 0$

- In period t, we hold inventory *and* have backlogged demand
- This doesn't make sense! Should use inventory to satisfy the unmet demand

---

- It's OK: Won't happen in an optimal solution
  - Both $b_t > 0$ and $I_t > 0$ incur *costs* in objective
  - b_t and I_t *always appear together* in constraints

$$\begin{aligned} I_{t-1} + b_t + x_t + &= d_t + I_t + b_{t-1} \\ \Leftrightarrow (I_{t-1} - b_{t-1}) + x_t &= d_t + (I_t - b_t) \end{aligned}$$

  - Can decrease both by the same amount and still be feasible, until one becomes zero

Inventory position

- The quantity $I_t - b_t$ is sometimes called the inventory position.
  - It represents a *net* inventory level
  - Can be positive or negative (i.e., it is *unrestricted in sign*)
  - Positive $\Rightarrow I_t > 0$ and $b_t = 0$, we are holding inventory
  - Negative $\Rightarrow I_t = 0$ and $b_t > 0$, we have a backlog

---

- We need separate decision variables for (positive) inventory level and backlog, to account for the costs of those
- There is another way to think about backlogging

How to Model Backlogging

- Think of inventory being allowed to go negative, and let n_t be this "net inventory position"
- Picture still makes sense, since if inventory is negative, you need to "make up" for it during one of the next periods
- You can set last period demand $n_{|T|} \geq 0$ to ensure that all demand is *eventually* met.
- Cost function $F(n_t)$:

$$F(n_t) = \begin{cases} \iota n_t & \text{if } n_t \geq 0 \\ -\theta n_t & \text{if } n_t < 0 \end{cases}$$

- Is $F(n_t)$ a linear function of n_t? no!

Another Nonlinear/Linear Trick

- To model the case where we are minimizing a convex piecewise linear function (like $F(\cdot)$ or $|\cdot|$), we can introduce a variable for each piece
- Write constraints $n_t = I_t - b_t \quad \forall t \in T$
  - Think of this as (Leftover - Shortage)
- Objective gets terms:

$$\sum_{t \in T} (\iota I_t + \theta b_t)$$

- This trick only works if we are *minimizing* costs. Then at most one of I_t and b_t will ever be positive in an optimal solution.
- We will learn more about modeling piecewise linear functions next time

CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

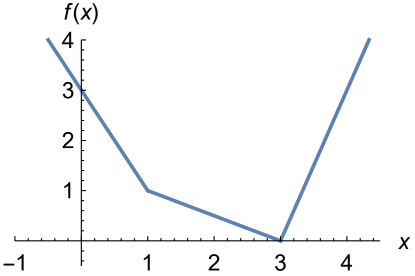
Piecewise linear functions

- Some problems do not appear to be LPs but can be converted to LPs using a suitable transformation.
- An important case: *convex piecewise linear functions*.

Consider the following **nonlinear** optimization:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to:} & x \geq 0 \end{array}$$

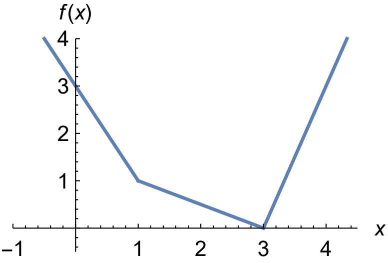
Where $f(x)$ is the function:

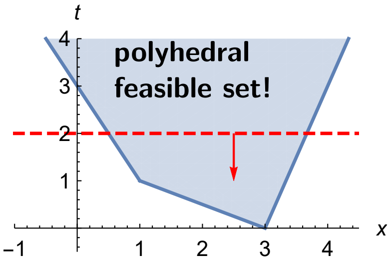


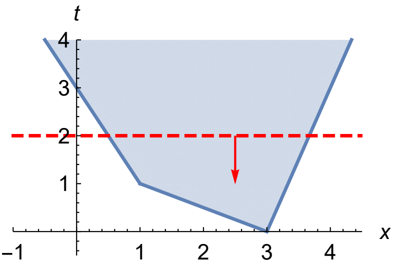
Piecewise Linear Functions

The trick is to convert the problem into **epigraph** form: add an extra decision variable t and turn the cost into a constraint!

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to:} & x \geq 0 \end{array} \quad \Longrightarrow \quad \begin{array}{ll} \underset{x,t}{\text{minimize}} & t \\ \text{subject to:} & t \geq f(x) \\ & x \geq 0 \end{array}$$







$$\begin{array}{ll} \underset{x,t}{\text{minimize}} & t \\ \text{subject to:} & t \geq f(x) \\ & x \geq 0 \end{array}$$

This feasible set is **polyhedral**. It is the set of (x, t) satisfying:

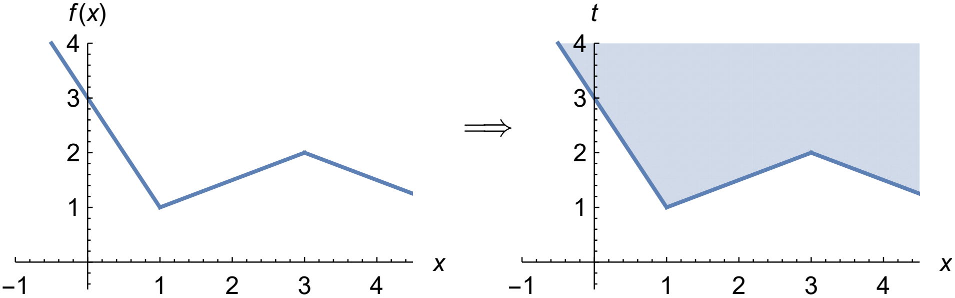
$$\left\{ t \geq -2x + 3, \quad t \geq -\tfrac{1}{2}x + \tfrac{3}{2}, \quad t \geq 3x - 9 \right\}$$

Equivalent linear program:

$$\begin{array}{lll} \underset{x,t}{\text{minimize}} & t & \\ \text{subject to:} & t \geq -2x + 3, & t \geq -\frac{1}{2}x + \frac{3}{2} \\ & t \geq 3x - 9, & x \geq 0 \end{array}$$

Piecewise linear functions

Epigraph trick only works if it's a **convex polyhedron**.

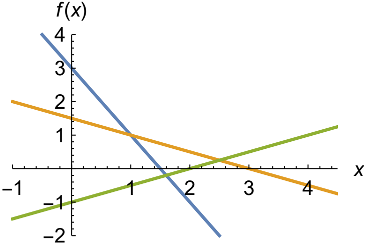


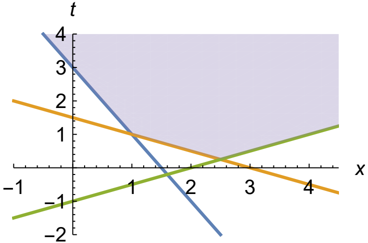
This epigraph is **not a convex polyhedron** so it cannot be the feasible set of a linear program.

Minimax problems

- The maximum of several linear functions is *always* convex. So we can minimize it using the epigraph trick. Example:

$$f(x) = \max_{i=1,\dots,k} \left\{ a_i^\mathsf{T} x + b_i \right\}$$



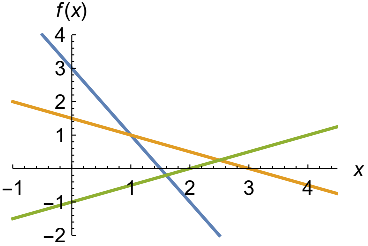


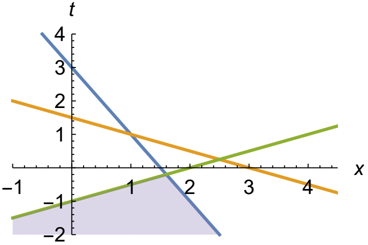
$$\min_{x} \max_{i=1,\dots,k} \left\{ a_i^\mathsf{T} x + b_i \right\} \quad \Longrightarrow \quad \begin{array}{ll} \min\limits_{x,t} & t \\ \text{s.t.} & t \geq a_i^\mathsf{T} x + b_i \quad i = 1, 2, \dots, k. \end{array}$$

Maximin problems

- The minimum of several linear functions is *always* concave. So we can maximize it using the epigraph trick. Example:

$$f(x) = \min_{i=1,\ldots,k} \{a_i^\mathsf{T} x + b_i\}$$





$$\max_x \min_{i=1,\ldots,k} \{a_i^\mathsf{T} x + b_i\} \quad \Longrightarrow \quad \begin{aligned} \max_{x,t} \quad & t \\ \text{s.t.} \quad & t \le a_i^\mathsf{T} x + b_i \quad \forall i \end{aligned}$$

Minimax and Maximin problems

- A minimax problem:

$$\min_x \max_{i=1,\ldots,k} \{a_i^\mathsf{T} x + b_i\} \implies \begin{array}{ll} \min\limits_{x,t} & t \\ \text{s.t.} & t \geq a_i^\mathsf{T} x + b_i \quad \forall i \end{array}$$

- A maximin problem:

$$\max_x \min_{i=1,\ldots,k} \{a_i^\mathsf{T} x + b_i\} \implies \begin{array}{ll} \max\limits_{x,t} & t \\ \text{s.t.} & t \leq a_i^\mathsf{T} x + b_i \quad \forall i \end{array}$$

Note: Sometimes called *minmax*, *min-max*, *min/max*.
Of course, minmax $\neq$ maxmin!

Absolute values

- Absolute values are piecewise linear! For $x \in \mathbb{R}$:

$$\begin{array}{ll} \min_{x} & |x| \\ \text{s.t.} & Ax \le b \end{array} \implies \begin{array}{ll} \min_{x,t} & t \\ \text{s.t.} & Ax \le b \\ & t \ge x \\ & t \ge -x \end{array}$$

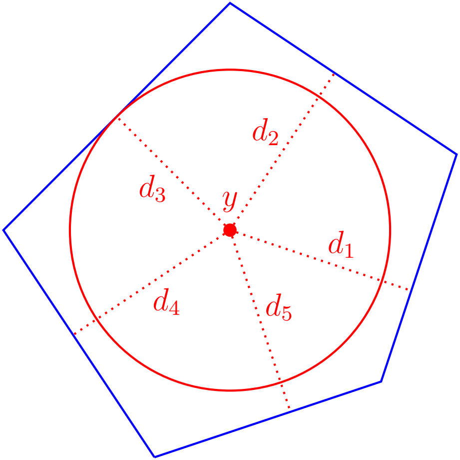
- So are sums of absolute values:

$$\min_{x,y} \quad |x| + |y| \implies \begin{array}{lll} \min_{x,y,t,r} & t + r & \\ \text{s.t.} & t \ge x, & t \ge -x \\ & r \ge y, & r \ge -y \end{array}$$

- But not differences! $\min_{x,y} \; |x| - 2|y|$ is not an LP.

Chebyshev center

What is the largest sphere you can fit inside a polyhedron?



If y is the center, then draw perpendicular lines to each face of the polyhedron.

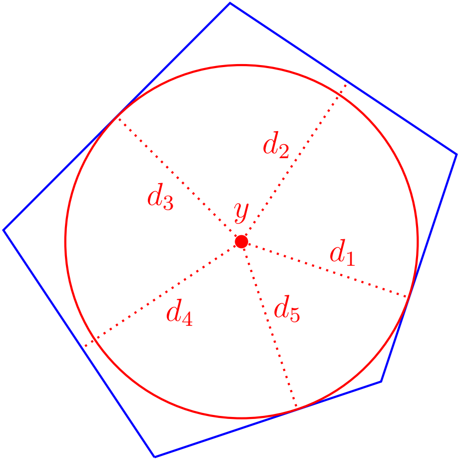
We want to maximize the smallest d_i. In other words,

$$\max_{y} \min_{i=1,\ldots,5} d_i(y)$$

(the y shown here is obviously not optimal!)

Chebyshev center

What is the largest sphere you can fit inside a polyhedron?



If y is the center, then draw perpendicular lines to each face of the polyhedron.

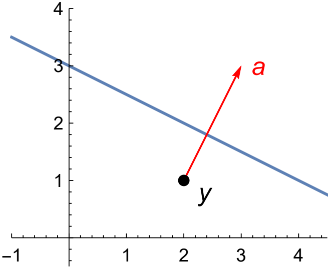
We want to maximize the smallest d_i. In other words,

$$\max_{y} \min_{i=1,\ldots,5} d_i(y)$$

The optimal y is the Chebyshev center

Chebyshev center

Finding the Chebyshev center amounts to solving an LP!



To compute the distance between y and the hyperplane $a^\mathsf{T} x = b$, notice that if the distance is r, then $y + \frac{r}{\|a\|} a$ belongs to the hyperplane:

$$a^\mathsf{T} \left(y + \frac{r}{\|a\|} a \right) = b$$

Simplifying, we obtain: $a^\mathsf{T} y + \|a\| r = b$

"The distance between y and each hyperplane is at least r" is equivalent to saying that $a_i^\mathsf{T} y + \|a_i\| r \leq b_i$ for each i.

Chebyshev center

Finding the Chebyshev center amounts to solving an LP!

The transformation to an LP is given by:

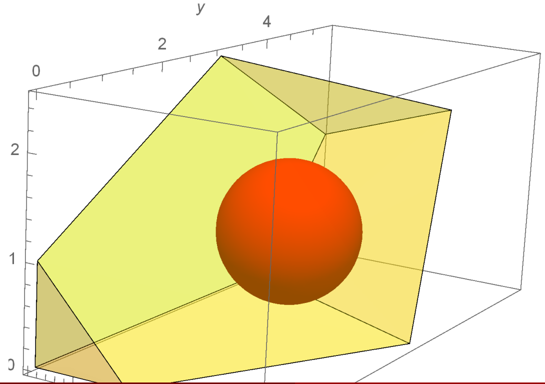
$$\begin{array}{lll} \max\limits_{y} & \min\limits_{i=1,\dots,k} d_i(y) & \\ \text{s.t.} & a_i^\mathsf{T} y \leq b_i & \forall i \end{array} \quad \Longrightarrow \quad \begin{array}{lll} \max\limits_{y,r} & r & \\ \text{s.t.} & a_i^\mathsf{T} y + \|a_i\| r \leq b_i & \forall i \end{array}$$

Chebyshev center

Example: find the Chebyshev center of the polyhedron defined by the following inequalities:

$$2x - y + 2z \leq 2,\ -x + 2y + 4z \leq 16,\ x + 2y - 2z \leq 8,$$
$$x \geq 0,\ y \geq 0,\ z \geq 0$$

Chebyshev.ipynb



Wash and Go With

- Project Scheduling: PERT (Project Evaluation and Review Technique)
- Often used synonomously with CPM: Critical Path Method

PERT

- I: Set of projects
- $P \subset I \times I$: Precedence relationships. $((i, j) \in P \Rightarrow i$ immediately follows $j)$
- a_i: Duration of activity $i \in I$

Modeling PERT

Variables

- t_i: Time activity starts

Constraints

- i cannot begin before j finishes:

$$t_i \geq t_j + a_j \qquad \forall (i, j) \in P$$

Objective

- Minimize the latest job completion time (makespan).

$$\min \max\{t_1 + a_1, t_2 + a_2 \ldots, t_{|I|} + a_{|I|}\}.$$

Mini-Max

- Minimax will haunt you

$$T^* = \min z$$

$$\begin{aligned} z &\geq t_i + a_i \ \forall i \in I \\ t_i &\geq t_j + a_j \ \forall (i,j) \in P \\ t_i &\geq 0 \ \forall i \in I \end{aligned}$$

Example: building a house

Several tasks must be completed in order to build a house.

- Each task takes a known amount of time to complete.
- A task may depend on other tasks, and can only be started once those tasks are complete.
- Tasks may be worked on simultaneously as long as they don't depend on one another.
- How fast can the house be built?

Job No.	Description	Immediate predecessors	Normal time (days)
a	Start		0
b	Excavate and pour footers	*a*	4
c	Pour concrete foundation	*b*	2
d	Erect wooden frame including rough roof	*c*	4
e	Lay brickwork	*d*	6
f	Install basement drains and plumbing	*c*	1
g	Pour basement floor	*f*	2
h	Install rough plumbing	*f*	3
i	Install rough wiring	*d*	2
j	Install heating and ventilating	*d,g*	4
k	Fasten plaster board and plaster (including drying)	*i,j,h*	10
l	Lay finish flooring	*k*	3
m	Install kitchen fixtures	*l*	1
n	Install finish plumbing	*l*	2
o	Finish carpentry	*l*	3
p	Finish roofing and flashing	*e*	2
q	Fasten gutters and downspouts	*p*	1
r	Lay storm drains for rain water	*c*	1
s	Sand and varnish flooring	*o,t*	2
t	Paint	*m,n*	3
u	Finish electrical work	*t*	1
v	Finish grading	*q,r*	2
w	Pour walks and complete landscaping	*v*	5
x	Finish	*s,u,w*	0

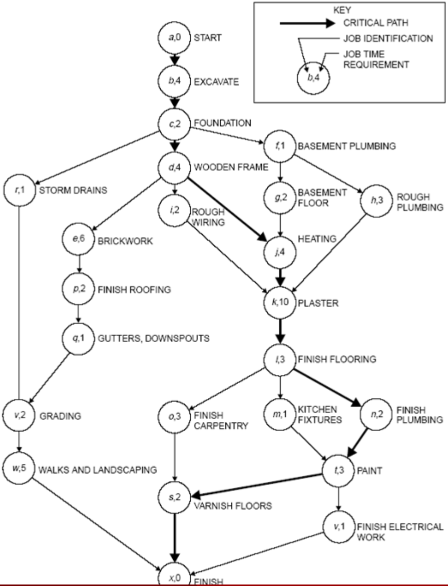
Example: building a house

The data can be visualized using a directed graph.

- Arrows indicate task dependencies.

What are the decision variables?

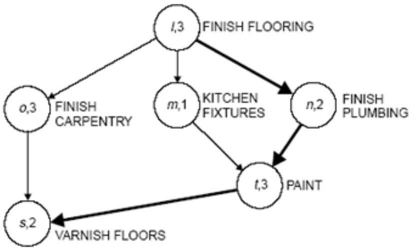
- t_i: *start time* of i^{th} task.
- precedence constraints are expressed in terms of t_i's.
- minimize t_x.



A small sample:

Let t_l, t_o, t_m, t_n, t_t, t_s be start times of the associated tasks.

Now use the graph to write the dependency constraints:



- Tasks o, m, and n can't start until task l is finished, and task l takes 3 days to finish. So the constraints are:
$$t_l + 3 \le t_o, \quad t_l + 3 \le t_m, \quad t_l + 3 \le t_n$$
- Task t can't start until tasks m and n are finished. Therefore:
$$t_m + 1 \le t_t, \quad t_n + 2 \le t_t,$$
- Task s can't start until tasks o and t are finished. Therefore:
$$t_o + 3 \le t_s, \quad t_t + 3 \le t_s$$

Example: building a house

Full implementation in Julia:

House.ipynb

- Follow-up: which tasks in the project are **critical** to finishing on time?
- Which tasks can withstand delays?
- related to notion of *duality* we will see later.

Next...

- more examples of sequential problems
- transportation/shipment problems
- assignment problems
- shortest path problems
- network flows

CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

Example: Sailco

Sailco manufactures sailboats. During the next 4 months the company must meet the following demands for their sailboats:

Month	1	2	3	4
Number of boats	40	60	70	25

At the beginning of Month 1, Sailco has 10 boats in inventory. Each month it must determine how many boats to produce. During any month, Sailco can produce up to 40 boats with regular labor and an unlimited number of boats with overtime labor. Boats produced with regular labor cost \$400 each to produce, while boats produced with overtime labor cost \$450 each. It costs \$20 to hold a boat in inventory from one month to the next. Find the production and inventory schedule that minimizes the cost of meeting the next 4 months' demands.

Example: Sailco

Summary of problem data:

- Regular labor: \$400/boat (at most 40 boats/month).
- Overtime labor: \$450/boat (no monthly limit).
- Holding a boat in inventory costs \$20/month.
- Inventory initially has 10 boats.
- Demand for next 4 months is:

Month	1	2	3	4
Number of boats	40	60	70	25

What are the decision variables?

Example: Sailco

Remember: Decision variables aren't always things that you decide directly!

For this problem, the decision variables are:

- x_1, x_2, x_3, x_4: boats produced each month with regular labor.
- y_1, y_2, y_3, y_4: boats produced each month with overtime.
- h_1, h_2, h_3, h_4, h_5: boats in inventory at start of each month.

Parameters:

- d_1, d_2, d_3, d_4: demand at each month

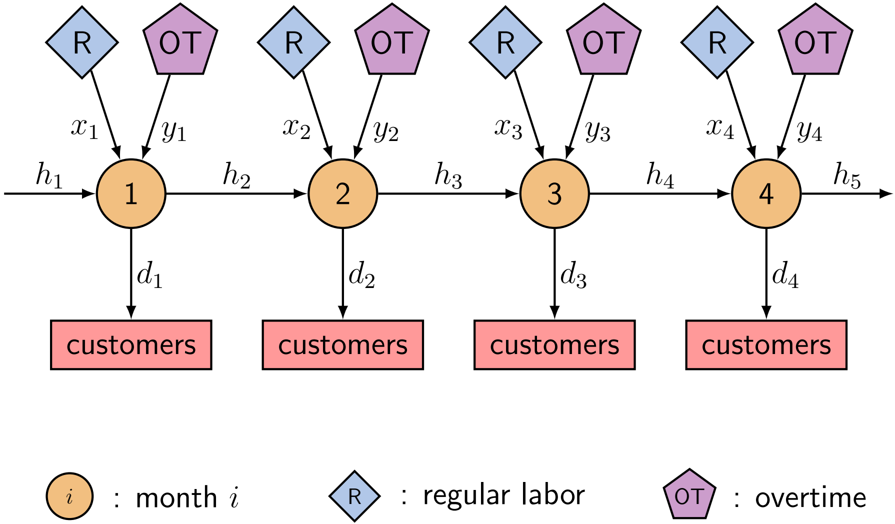
Example: Sailco

The constraints are:

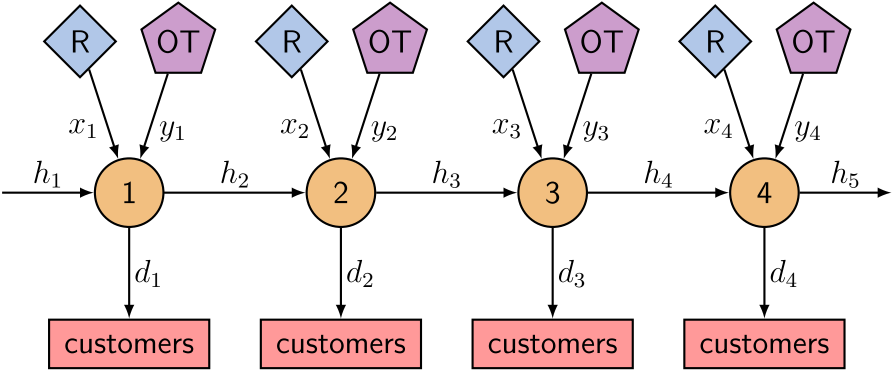
- $0 \leq x_i \leq 40$ (monthly limit of regular production)
- $y_i \geq 0$ (unlimited overtime production)
- Conservation of boats:
    - $h_i + x_i + y_i = d_i + h_{i+1}$ (for $i = 1, 2, 3, 4$)
    - $h_1 = 10$ (initial inventory)

Solution: Sailco.ipynb

Example: Sailco



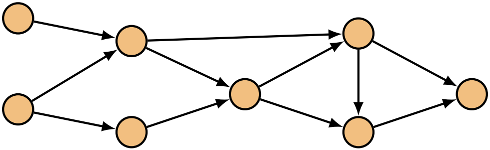
Example: Sailco



- Arrows indicate flow of boats
- conservation at nodes: $h_1 + x_1 + y_1 = d_1 + h_2$, etc.

Minimum-cost flow problems

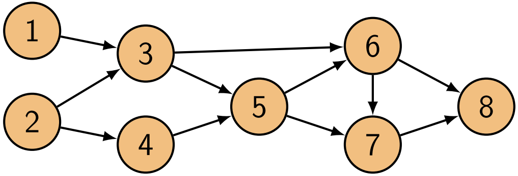
- Many optimization problems can be interpreted as **network flow problems** on a directed graph.



- Decision variables: **flow on each edge**.
- Edges have flow costs and capacity constraints
- Each node can:
  - produce/supply flow (source)
  - consume/demand flow (sink)
  - conserve flow (relay)

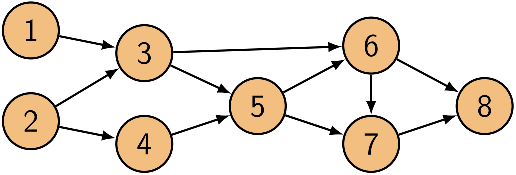
What is the minimum-cost feasible flow?

Minimum-cost flow problems



- The set of nodes: $\mathcal{N} = \{1, \ldots, 8\}$.
- The set of directed edges: $\mathcal{E} = \{(1,3), (2,3), (2,4), \ldots\}$.
- Each node $i \in \mathcal{N}$ supplies a flow b_i. Node i is called a *source* if $b_i > 0$, a *relay* if $b_i = 0$, and a *sink* if $b_i < 0$.
- **Decision variables**: x_{ij} is the flow on edge $(i,j) \in \mathcal{E}$.
- **Flow cost**: c_{ij} is cost per unit of flow on edge $(i,j) \in \mathcal{E}$.

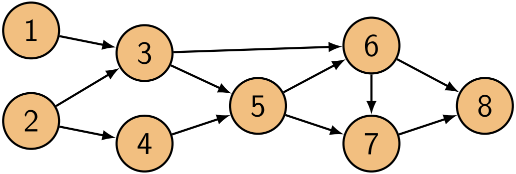
Minimum-cost flow problems



- **Decision variables**: x_{ij} is the flow on edge $(i,j) \in \mathcal{E}$.
- **Capacity constraints**: $p_{ij} \le x_{ij} \le q_{ij} \qquad \forall (i,j) \in \mathcal{E}$.
- **Conservation**: $\sum_{j \in \mathcal{N}:(i,j)\in\mathcal{E}} x_{ij} - \sum_{j \in \mathcal{N}:(j,i)\imath\mathcal{E}} x_{ji} = b_i \quad \forall i \in \mathcal{N}$.
- **Total cost**: $\sum_{(i,j)\in\mathcal{E}} c_{ij}x_{ij}$.

Note: $b_i, c_{ij}, p_{ij}, q_{ij}$ are parameters.

Minimum-cost flow problems

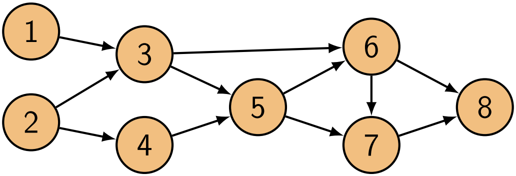


$$\begin{array}{lll} \underset{x_{ij}\in\mathbb{R}}{\text{minimize}} & \sum_{(i,j)\in\mathcal{E}} c_{ij}x_{ij} & \\ \text{subject to:} & \sum_{j\in\mathcal{N}:(i,j)\in\mathcal{E}} x_{kj} - \sum_{j\in\mathcal{N}:(j,i)\in\mathcal{E}} x_{ji} = b_i & \forall i \in \mathcal{N} \\ & p_{ij} \le x_{ij} \le q_{ij} & \forall (i,j) \in \mathcal{E} \end{array}$$

"Flow Balance" Mantra

"Out" - "In" = "Net Supply"

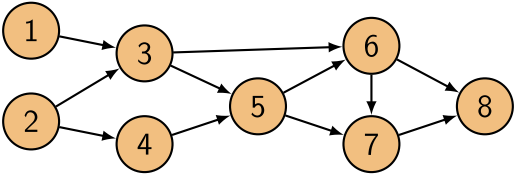
Minimum-cost flow problems



Expanded conservation constraint:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}}_{A = \text{incidence matrix}} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{24} \\ x_{35} \\ x_{36} \\ x_{45} \\ x_{56} \\ x_{57} \\ x_{67} \\ x_{68} \\ x_{78} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{bmatrix}$$

Minimum-cost flow problems



The entire model (compact form):

$$\begin{aligned} \underset{x \in \mathbb{R}^{|\mathcal{E}|}}{\text{minimize}} \quad & c^{\mathsf{T}} x \\ \text{subject to:} \quad & Ax = b \\ & p \leq x \leq q \end{aligned}$$

Note: The matrix A is a node-arc incidence matrix

Balanced problems

The incidence matrix has the property that all columns sum to zero: $1^\top A = 0$.

This is because each column corresponds to a single edge, which has one origin node $(+1)$ and one destination (-1).

Since $Ax = b$ is a constraint, we must therefore have in all feasible solutions: $1^\top Ax = 1^\top b = 0$. Therefore:

$$\sum_{i \in \mathcal{N}} b_i = 0 \quad \text{(total supply = total demand)}$$

- If $\sum_{i \in \mathcal{N}} b_i = 0$, the model is called **balanced**.
- Unbalanced models are *always* infeasible.
- Note: balanced models may still be infeasible.

Balanced problems

Unbalanced models still make sense in practice, e.g. we may have excess supply or allow excess demand. These cases can be handled by making small modifications to the problem, such as changing "$=$" to "$\leq$".

- E.g. if have oversupply ($\sum_i b_i > 0$), change all source node balance to $\leq$
- Can also add "dummy nodes" to balance things if you wish

Minimum-cost flow problems

Many problem types are actually min-cost flow models:

- transportation problems
- assignment problems
- transshipment problems
- shortest path problems
- max-flow problems

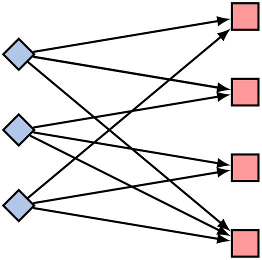
Let's look at these in more detail...

Legend: : source : relay : sink

Transportation problems

The objective is to transport a particular commodity from several possible sources to several possible destinations while minimizing the total cost.

- Sources have known supply limits
- Destinations each have demands
- Edges may have capacity limits
- Each link has an associated cost



Transportation Problem

MCNF on a bipartite graph

Transportation example

Millco has three wood mills and is planning three new logging sites. Each mill has a maximum capacity and each logging site can harvest a certain number of truckloads of lumber per day. The cost of a haul is $2/mile of distance. If distances from logging sites to mills are given below, how should the hauls be routed to minimize hauling costs while meeting all demands?

Logging site	Distance to mill (miles) Mill A	Mill B	Mill C	Maximum truckloads/day per logging site
1	8	15	50	20
2	10	17	20	30
3	30	26	15	45
Mill demand (truckloads/day)	30	35	30	

Note: problem is balanced!

Transportation example

- Arrange nodes as: $\begin{bmatrix} 1 & 2 & 3 & A & B & C \end{bmatrix}$ (sources, sinks).
- Graph is fully connected. Incidence matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

Julia code: Millco.ipynb

Transportation example

Logging site	Distance to mill (miles) Mill A	Mill B	Mill C	Maximum truckloads/day per logging site
1	8	15	50	20
2	10	17	20	30
3	30	26	15	45
Mill demand (truckloads/day)	30	35	30	

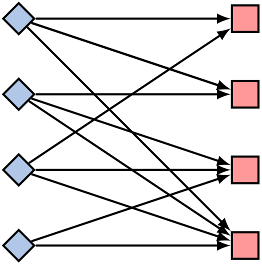
Solution is:

	A	**B**	**C**	
1	20	0	0	20
2	10	20	0	30
3	0	15	30	45
	30	35	30	

Assignment problems

We have n people and n tasks. The goal is to assign each person to a task. Each person has different preferences (costs) associated with performing each of the tasks. The goal is to find an assignment that minimizes the total cost.

- It's just a transportation problem!
- Each source has supply $= 1$
- Each sink has demand $= 1$
- Edges are unconstrained



Assignment Problem

- MCNF on $G = (L \cup R, E)$
- $b_i = 1 \forall i \in L, b_i = -1 \forall i \in R$

What about the integer constraint? More about this later...

Assignment example

The coach of a swim team needs to assign swimmers to a 200-yard medley relay team to compete in a tournament. The problem is that his best swimmers are good in more than one stroke, so it's not clear which swimmer to assign to which stroke. Here are the best times for each swimmer:

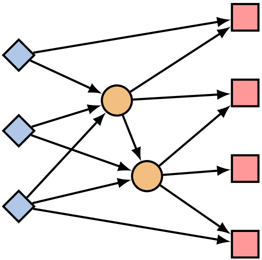
Stroke	Carl	Chris	David	Tony	Ken
Backstroke	37.7	32.9	33.8	37.0	35.4
Breaststroke	43.4	33.1	42.2	34.7	41.8
Butterfly	33.3	28.5	38.9	30.4	33.6
Freestyle	29.2	26.4	29.6	28.5	31.1

Julia code: Swim Relay.ipynb

Transshipment problems

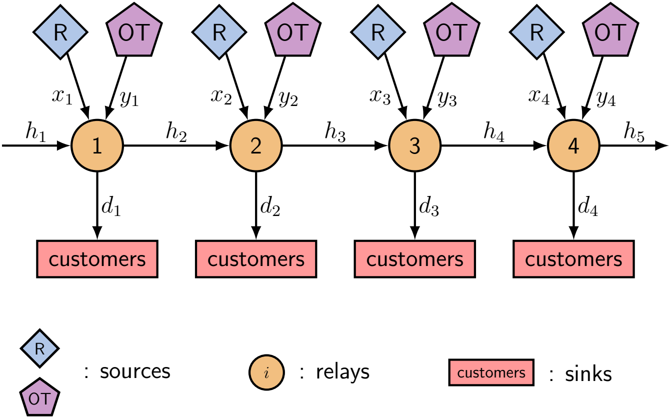
The same as a transportation problem, but in addition to sources and destinations, we also have warehouses that can store goods. The warehouses are **relay nodes**.

- Sources have known supply limits
- Destinations each have demands
- Links may have capacity limits
- Each link has an associated cost
- For warehouses, inflow = outflow.



Sailco problem is a transshipment problem!

Transshipment example: Sailco

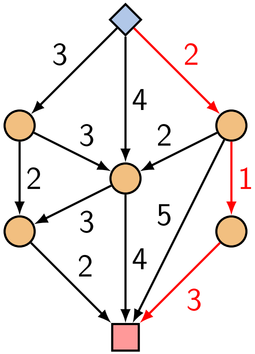


- The "warehouses" are the different months.
- Storing in inventory = shipping to the future.

Shortest/longest path problems

We have a directed graph and edge lengths. The goal is to find the shortest or longest path between two given nodes.

- Again, a transportation problem!
- Edge cost = length of path.
- The source has supply $= 1$
- The sink has demand $= 1$
- To find longest path, just change the min to a max!
- Only works for max if graph does not have a loop/cycle

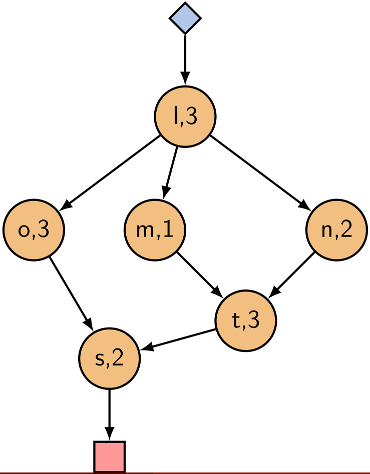


Again we need integer constraints on the edges...

Longest path example

The house building example is a longest path problem!

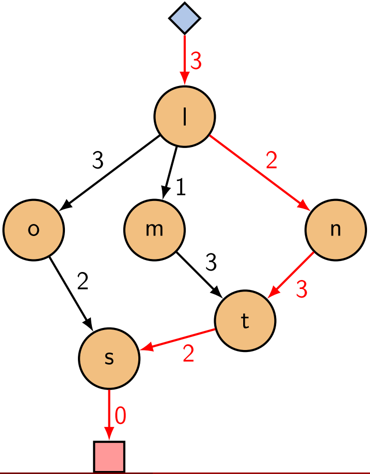
- Add source and sink nodes
- Move times out of nodes and onto preceding edges



Longest path example

The house building example is a longest path problem!

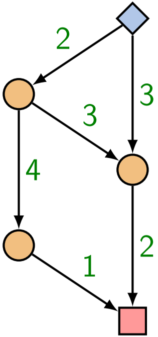
- Add source and sink nodes
- Move times out of nodes and onto preceding edges. The cost on the edge now means "it takes this long to finish the task at the destination node."
- Each path says "it takes at least this long"— Longest path gives the shortest time we have to wait.



Max-flow problems

We are given a directed graph and **edge capacities**. Find the maximum flow that we can push from source to sink.

- Edges have max capacities
- Flow can split!
- notions of supply and demand don't make sense...
- add a feedback path and make every node a relay!

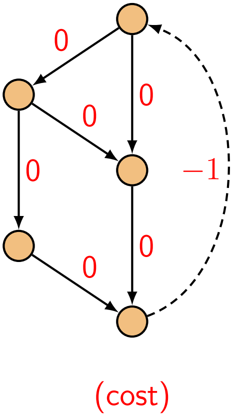


(edge capacity)

Max-flow problems

We are given a directed graph and **edge capacities**. Find the maximum flow that we can push from source to sink.

- Edges have max capacities
- Flow can split!
- notions of supply and demand don't make sense...
- add a feedback path and make every node a relay!



Solve minimum-cost flow where feedback path has cost (-1) and all other paths have zero cost.

Max Flow

Maximum Flow Problem

Given a capacitated network $G = (N, A)$, with capacities $u \in \mathbb{R}_+^{|A|}$, a source node $s \in N$, and a sink node $t \in V$, what is the maximum flow that can be sent from s to t.

- Model "what goes in = what come out".

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0$$

- Be sure to add an arc from $t \to s$ in A

Max Flow Problem (as MCNF)

$$-\min -x_{ts}$$

$$\begin{array}{rrcll} \text{s.t.} & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} & = & 0 & \forall i \in N \\ & x_{ij} & \leq & u_{ij} & \forall (i,j) \in A \\ & x_{ij} & \geq & 0 & \forall (i,j) \in A \end{array}$$

- Note that due to flow balance, objective is the same as

$$\sum_{j:(j,t)\in A} x_{jt}$$

Let's Have a Picnic!

- The Hatfields, Montagues, McCoys and Capulets are going on their annual family picnic.
- Four cars are available to transport the families to the picnic.
- The cars can carry the following numbers of people: car 1, 4; car 2, 3; car 3, 3; car 4, 4.
- There are four people in each family, and no car can carry more than two people from any one family.
- Determine the maximum number of people that can be transported to the picnic.

Max Flow Problem

- Let's try and model this is a max flow problem

The $0.0001 Question

- Who Can Make This A Max Flow Problem?

Picnic.ipynb

Integer solutions

Some minimum-cost flow problems require integer solutions (assignment problems and shortest path problems). Is there a way of guaranteeing integer solutions? **yes!**

Definition: A matrix A is *totally unimodular* (TU) if every square submatrix of A has determinant 0, 1, or -1.

- The definition includes 1×1 submatrices, so every entry of A must be 0, 1, or -1.
- ex. $\begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ is TU but $\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ is not.

Integer solutions

Theorem: If A is TU and b is an integer vector, then the vertices of $\{x \mid Ax \leq b\}$ have integer coordinates.

Theorem: Every (node-arc) incidence matrix is TU.

What does this mean? (**a lot!**)

- If a minimum-cost flow problem has integer supplies, integer demands, and integer edge capacities, then there is a minimum-cost **integer** flow.
- every assignment problem is an LP.
- every shortest path problem is an LP.

Next class...

- Duality theory
- Shadow price interpretation
- Sensitivity analysis

CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

February 24, 2024

The Top Brass example revisited

$$\begin{array}{lll} \underset{f,s}{\text{maximize}} & 12f + 9s & \\ \text{subject to:} & 4f + 2s \leq 4800, & f + s \leq 1750 \\ & 0 \leq f \leq 1000, & 0 \leq s \leq 1500 \end{array}$$

Suppose the maximum profit is $p^\star$. How can we *bound* $p^\star$?

- Finding a *lower* bound is easy... pick any feasible point!
  - $\{f = 0, s = 0\}$ is feasible. So $p^\star \geq 0$ (we can do better...)
  - $\{f = 500, s = 1000\}$ is feasible. So $p^\star \geq 15000$.
  - $\{f = 1000, s = 400\}$ is feasible. So $p^\star \geq 15600$.
- Each feasible point of the LP yields a lower bound for $p^\star$.
- Finding the largest lower bound = solving the LP!

Estimating upper bounds

$$\begin{array}{ll} \underset{f,s}{\text{maximize}} & 12f + 9s \\ \text{subject to:} & 4f + 2s \leq 4800, \quad f + s \leq 1750 \\ & 0 \leq f \leq 1000, \quad 0 \leq s \leq 1500 \end{array}$$

Suppose the maximum profit is $p^\star$. How can we *bound* $p^\star$?

- Finding an *upper* bound is harder... (use the constraints!)
  - $12f + 9s \leq 12 \cdot 1000 + 9 \cdot 1500 = 25500$. So $p^\star \leq 25500$.
  - $12f + 9s \leq f + (4f + 2s) + 7(f + s)$
    $\leq 1000 + 4800 + 7 \cdot 1750 = 18050$. So $p^\star \leq 18050$.
- Combining the constraints in different ways yields different upper bounds on the optimal profit $p^\star$.

Estimating upper bounds

$$\begin{aligned} \underset{f,s}{\text{maximize}} \quad & 12f + 9s \\ \text{subject to:} \quad & 4f + 2s \le 4800, \quad f + s \le 1750 \\ & 0 \le f \le 1000, \quad 0 \le s \le 1500 \end{aligned}$$

Suppose the maximum profit is $p^\star$. How can we *bound* $p^\star$?

What is the **best** upper bound we can find by combining constraints in this manner?

Estimating upper bounds

$$\begin{array}{ll} \underset{f,s}{\text{maximize}} & 12f + 9s \\ \text{subject to:} & 4f + 2s \le 4800, \quad f + s \le 1750 \\ & 0 \le f \le 1000, \quad 0 \le s \le 1500 \end{array}$$

- Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \ge 0$ be the multipliers. If we can choose them such that for *any* feasible f and s, we have:

$$12f + 9s \le \lambda_1(4f + 2s) + \lambda_2(f + s) + \lambda_3 f + \lambda_4 s \qquad (1)$$

Then, using the constraints, we will have the following upper bound on the optimal profit:

$$12f + 9s \le 4800\lambda_1 + 1750\lambda_2 + 1000\lambda_3 + 1500\lambda_4$$

Estimating upper bounds

$$\begin{array}{ll} \underset{f,s}{\text{maximize}} & 12f + 9s \\ \text{subject to:} & 4f + 2s \leq 4800, \quad f + s \leq 1750 \\ & 0 \leq f \leq 1000, \quad 0 \leq s \leq 1500 \end{array}$$

- Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$ be the multipliers. If we can choose them such that for *any* feasible f and s, we have:

$$12f + 9s \leq \lambda_1(4f + 2s) + \lambda_2(f + s) + \lambda_3 f + \lambda_4 s \qquad (1)$$

Rearranging (1), we get:

$$0 \leq (4\lambda_1 + \lambda_2 + \lambda_3 - 12)f + (2\lambda_1 + \lambda_2 + \lambda_4 - 9)s$$

We can ensure this always holds by choosing $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ to make the bracketed terms nonnegative.

Estimating upper bounds

$$\begin{aligned} \max_{f,s} \quad & 12f + 9s \\ \text{subject to:} \quad & 4f + 2s \leq 4800, \quad f + s \leq 1750 \\ & 0 \leq f \leq 1000, \quad 0 \leq s \leq 1500 \end{aligned}$$

- **Recap**: If we choose $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$ such that:

$$4\lambda_1 + \lambda_2 + \lambda_3 \geq 12 \quad \text{and} \quad 2\lambda_1 + \lambda_2 + \lambda_4 \geq 9$$

Then we have a *upper* bound on the optimal profit:

$$p^\star \leq 4800\lambda_1 + 1750\lambda_2 + 1000\lambda_3 + 1500\lambda_4$$

Finding the best (smallest) upper bound is... an LP!

The dual of Top Brass

$$
\begin{array}{ll}
\underset{f,s}{\text{maximize}} & 12f + 9s \\
\text{subject to:} & 4f + 2s \leq 4800, \quad f + s \leq 1750 \\
 & 0 \leq f \leq 1000, \quad 0 \leq s \leq 1500
\end{array}
$$

To find the best upper bound, solve the **dual** problem:

$$
\begin{array}{ll}
\underset{\lambda_1,\lambda_2,\lambda_3,\lambda_4}{\text{minimize}} & 4800\lambda_1 + 1750\lambda_2 + 1000\lambda_3 + 1500\lambda_4 \\
\text{subject to:} & 4\lambda_1 + \lambda_2 + \lambda_3 \geq 12 \\
 & 2\lambda_1 + \lambda_2 + \lambda_4 \geq 9 \\
 & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0
\end{array}
$$

The dual of Top Brass

Primal problem:

$$\begin{aligned} \underset{f,s}{\text{maximize}} \quad & 12f + 9s \\ \text{subject to:} \quad & 4f + 2s \le 4800 \\ & f + s \le 1750 \\ & f \le 1000 \\ & s \le 1500 \\ & f, s \ge 0 \end{aligned}$$

Solution is $p^\star$.

Dual problem:

$$\begin{aligned} \underset{\lambda_1,\dots,\lambda_4}{\text{minimize}} \quad & 4800\lambda_1 + 1750\lambda_2 \\ & \quad + 1000\lambda_3 + 1500\lambda_4 \\ \text{subject to:} \quad & 4\lambda_1 + \lambda_2 + \lambda_3 \ge 12 \\ & 2\lambda_1 + \lambda_2 + \lambda_4 \ge 9 \\ & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \ge 0 \end{aligned}$$

Solution is $d^\star$.

- Primal is a maximization, dual is a minimization.
- There is a dual variable for each primal constraint.
- There is a dual constraint for each primal variable.
- (any feasible primal point) $\le p^\star \le d^\star \le$ (any feasible dual point)

The dual of Top Brass

Primal problem:

$$\max_{f,s} \quad \begin{bmatrix} 12 \\ 9 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} f \\ s \end{bmatrix}$$

$$\text{s.t.} \quad \begin{bmatrix} 4 & 2 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f \\ s \end{bmatrix} \le \begin{bmatrix} 4800 \\ 1750 \\ 1000 \\ 1500 \end{bmatrix}$$

$$f, s \ge 0$$

Dual problem:

$$\min_{\lambda_1,\dots,\lambda_4} \quad \begin{bmatrix} 4800 \\ 1750 \\ 1000 \\ 1500 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}$$

$$\text{s.t.} \quad \begin{bmatrix} 4 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} \ge \begin{bmatrix} 12 \\ 9 \end{bmatrix}$$

$$\lambda_1, \lambda_2, \lambda_3, \lambda_4 \ge 0$$

Using matrix notation...

Code: Top Brass dual.ipynb

Danger!

> **Warning**
>
> JuMP's definition of duality is independent of the objective sense. That is, the sign of feasible duals associated with a constraint depends on the direction of the constraint and not whether the problem is maximization or minimization. **This is a different convention from linear programming duality in some common textbooks.** If you have a linear program, and you want the textbook definition, you probably want to use `shadow_price` and `reduced_cost` instead.

Take Away Message

Be careful interpreting the dual solution value when using Jump `dual()`

Weak Duality

Primal problem (P)

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^\mathsf{T} x \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Dual problem (D)

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & b^\mathsf{T} \lambda \\ \text{subject to:} \quad & A^\mathsf{T} \lambda \geq c \\ & \lambda \geq 0 \end{aligned}$$

If x and λ are feasible points of (P) and (D) respectively:

$$c^\mathsf{T} x \leq p^\star \leq d^\star \leq b^\mathsf{T} \lambda$$

Weak Duality: The value of every feasible dual solution provides an (upper) bound on the value of every feasible primal solution.

Weak Duality

Primal problem (P)

$$\begin{array}{ll} \underset{x}{\text{maximize}} & c^{\mathsf{T}}x \\ \text{subject to:} & Ax \leq b \\ & x \geq 0 \end{array}$$

Dual problem (D)

$$\begin{array}{ll} \underset{\lambda}{\text{minimize}} & b^{\mathsf{T}}\lambda \\ \text{subject to:} & A^{\mathsf{T}}\lambda \geq c \\ & \lambda \geq 0 \end{array}$$

If x and λ are feasible points of (P) and (D) respectively:

$$c^{\mathsf{T}}x \leq p^{\star} \leq d^{\star} \leq b^{\mathsf{T}}\lambda$$

Strong Duality: if $p^{\star}$ and $d^{\star}$ exist and are finite, then $p^{\star} = d^{\star}$. This is a powerful and amazing fact.

General LP duality

Primal problem (P)

$$\begin{array}{ll} \underset{x}{\text{maximize}} & c^{\mathsf{T}}x \\ \text{subject to:} & Ax \leq b \\ & x \geq 0 \end{array}$$

Dual problem (D)

$$\begin{array}{ll} \underset{\lambda}{\text{minimize}} & b^{\mathsf{T}}\lambda \\ \text{subject to:} & A^{\mathsf{T}}\lambda \geq c \\ & \lambda \geq 0 \end{array}$$

1. optimal $p^\star$ is attained
2. unbounded: $p^\star = +\infty$
3. infeasible: $p^\star = -\infty$

1. optimal $d^\star$ is attained
2. unbounded: $d^\star = -\infty$
3. infeasible: $d^\star = +\infty$

Which combinations are possible? Remember: $p^\star \leq d^\star$.

General LP duality

Primal problem (P)

$$\begin{array}{ll} \underset{x}{\text{maximize}} & c^\mathsf{T} x \\ \text{subject to:} & Ax \leq b \\ & x \geq 0 \end{array}$$

Dual problem (D)

$$\begin{array}{ll} \underset{\lambda}{\text{minimize}} & b^\mathsf{T} \lambda \\ \text{subject to:} & A^\mathsf{T} \lambda \geq c \\ & \lambda \geq 0 \end{array}$$

There are **exactly four** possibilities:

1. (P) and (D) are both feasible and bounded, and $p^\star = d^\star$.
2. $p^\star = +\infty$ (unbounded primal) and $d^\star = +\infty$ (infeasible dual).
3. $p^\star = -\infty$ (infeasible primal) and $d^\star = -\infty$ (unbounded dual).
4. $p^\star = -\infty$ (infeasible primal) and $d^\star = +\infty$ (infeasible dual).

More properties of the dual

To find the dual of an LP that is **not** in standard form:

1. convert the LP to standard form
2. write the dual
3. make simplifications

Example: What is the dual of the dual? *the primal!*

$$
\begin{array}{rl}
\min\limits_{\lambda} & b^\mathsf{T}\lambda \\
\text{s.t.} & A^\mathsf{T}\lambda \geq c \\
& \lambda \geq 0
\end{array}
$$

equiv

$$
\begin{array}{rl}
-\max\limits_{\lambda} & (-b)^\mathsf{T}\lambda \\
\text{s.t.} & (-A^\mathsf{T})\lambda \leq (-c) \\
& \lambda \geq 0
\end{array}
$$

dual

$$
\begin{array}{rl}
-\min\limits_{z} & (-c)^\mathsf{T}z \\
\text{s.t.} & (-A)z \geq (-b) \\
& z \geq 0
\end{array}
$$

equiv

$$
\begin{array}{rl}
\max\limits_{z} & c^\mathsf{T}z \\
\text{s.t.} & Az \leq b \\
& z \geq 0
\end{array}
$$

More duals

Standard form:

$$\begin{array}{ll} \max_x & c^\mathsf{T} x \\ \text{s.t.} & Ax \le b \\ & x \ge 0 \end{array} \quad \xleftrightarrow{\text{dual}} \quad \begin{array}{ll} \min_\lambda & b^\mathsf{T} \lambda \\ \text{s.t.} & \lambda \ge 0 \\ & A^\mathsf{T}\lambda \ge c \end{array}$$

Free form:

$$\begin{array}{ll} \max_x & c^\mathsf{T} x \\ \text{s.t.} & Ax \le b \\ & x \text{ free} \end{array} \quad \xleftrightarrow{\text{dual}} \quad \begin{array}{ll} \min_\lambda & b^\mathsf{T} \lambda \\ \text{s.t.} & \lambda \ge 0 \\ & A^\mathsf{T}\lambda = c \end{array}$$

Mixed constraints:

$$\begin{array}{ll} \max_x & c^\mathsf{T} x \\ \text{s.t.} & Ax \le b \\ & Fx = g \\ & x \text{ free} \end{array} \quad \xleftrightarrow{\text{dual}} \quad \begin{array}{ll} \min_{\lambda,\mu} & b^\mathsf{T} \lambda + g^\mathsf{T}\mu \\ \text{s.t.} & \lambda \ge 0 \\ & \mu \text{ free} \\ & A^\mathsf{T}\lambda + F^\mathsf{T}\mu = c \end{array}$$

More duals

Equivalences between primal and dual problems

Minimization	Maximization
Nonnegative variable $\geq$	Inequality constraint $\leq$
Nonpositive variable $\leq$	Inequality constraint $\geq$
Free variable	Equality constraint $=$
Inequality constraint $\geq$	Nonnegative variable $\geq$
Inequality constraint $\leq$	Nonpositive variable $\leq$
Equality constraint $=$	Free Variable

Simple example

Why should we care about the dual?

1. It can sometimes make a problem easier to solve

$$\begin{array}{ll} \max\limits_{x,y,z} & 3x + y + 2z \\ \text{s.t.} & x + 2y + z \leq 2 \\ & x, y, z \geq 0 \end{array} \quad \overset{\text{dual}}{\Longleftrightarrow} \quad \begin{array}{ll} \min\limits_{\lambda} & 2\lambda \\ \text{s.t.} & \lambda \geq 3 \\ & 2\lambda \geq 1 \\ & \lambda \geq 2 \\ & \lambda \geq 0 \end{array}$$

   - Dual is much easier in this case!
   - Many solvers take advantage of duality.

2. Duality is related to the idea of sensitivity: how much do each of your constraints affect the optimal cost?

Sensitivity

Primal problem:

$$\begin{aligned} \underset{f,s}{\text{maximize}} \quad & 12f + 9s \\ \text{subject to:} \quad & 4f + 2s \le 4800 \\ & f + s \le 1750 \\ & f \le 1000 \\ & s \le 1500 \\ & f, s \ge 0 \end{aligned}$$

Solution is $p^\star$.

Dual problem:

$$\begin{aligned} \underset{\lambda_1,\dots,\lambda_4}{\text{minimize}} \quad & 4800\lambda_1 + 1750\lambda_2 \\ & + 1000\lambda_3 + 1500\lambda_4 \\ \text{subject to:} \quad & 4\lambda_1 + \lambda_2 + \lambda_3 \ge 12 \\ & 2\lambda_1 + \lambda_2 + \lambda_4 \ge 9 \\ & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \ge 0 \end{aligned}$$

Solution is $d^\star$.

If Millco offers to sell me more wood at a price of \$1 per board foot, should I accept the offer?

Sensitivity

Primal problem:

$$\begin{aligned} \underset{f,s}{\text{maximize}} \quad & 12f + 9s \\ \text{subject to:} \quad & 4f + 2s \le 4800 \\ & f + s \le 1750 \\ & f \le 1000 \\ & s \le 1500 \\ & f, s \ge 0 \end{aligned}$$

Solution is $p^\star$.

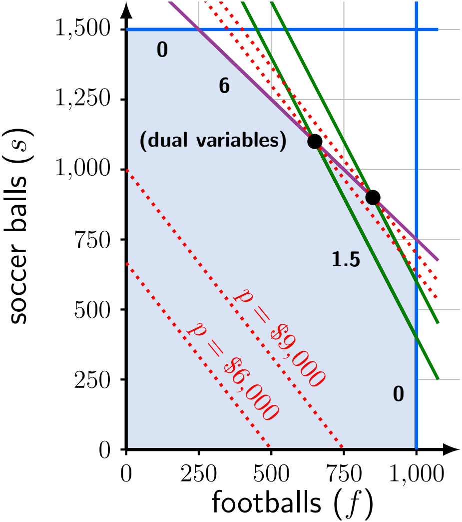
Dual problem:

$$\begin{aligned} \underset{\lambda_1,\dots,\lambda_4}{\text{minimize}} \quad & 4800\lambda_1 + 1750\lambda_2 \\ & + 1000\lambda_3 + 1500\lambda_4 \\ \text{subject to:} \quad & 4\lambda_1 + \lambda_2 + \lambda_3 \ge 12 \\ & 2\lambda_1 + \lambda_2 + \lambda_4 \ge 9 \\ & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \ge 0 \end{aligned}$$

Solution is $d^\star$.

- changes in primal *constraints* are changes in the dual *cost*.
- a small change to the feasible set of the primal problem can change the optimal f and s, but $\lambda_1, \dots, \lambda_4$ will not change!
- if we increase 4800 by 1, then $p^\star = d^\star$ increases by λ_1.

Sensitivity of Top Brass



$$\max_{f,s} \quad 12f + 9s$$
$$\text{s.t.} \quad 4f + 2s \le 5200$$
$$f + s \le 1750$$
$$0 \le f \le 1000$$
$$0 \le s \le 1500$$

What happens if we add 400 wood?

Profit goes up by \$600!

shadow price is \$1.50

Units

- In Top Brass, the primal variables f and s are the number of football and soccer trophies. The total profit is:

$$\text{(profit in \$)} = \left(12 \, \frac{\$}{\text{football trophy}}\right)(f \text{ football trophies}) + \left(9 \, \frac{\$}{\text{soccer trophy}}\right)(s \text{ soccer trophies})$$

- The dual variables also have units. To find them, look at the cost function for the dual problem:

$$\text{(profit in \$)} = (4800 \text{ board feet of wood})\left(\lambda_1 \, \frac{\$}{\text{board feet of wood}}\right) + (1750 \text{ plaques})\left(\lambda_2 \, \frac{\$}{\text{plaque}}\right) + \cdots$$

λ_i is the price that item i is worth to us.

Sensitivity in general

Primal problem (P)

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^{\mathsf{T}} x \\ \text{subject to:} \quad & Ax \le b + e \\ & x \ge 0 \end{aligned}$$

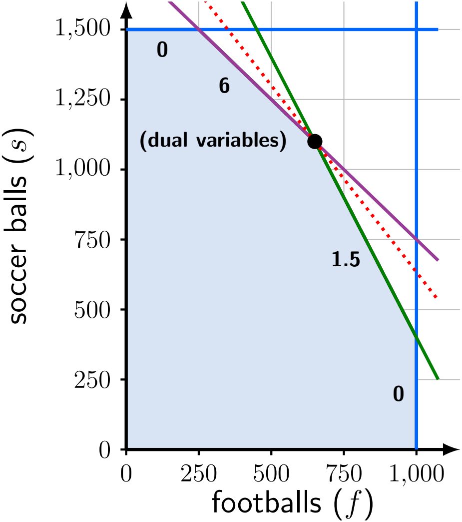
Dual problem (D)

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & (b + e)^{\mathsf{T}} \lambda \\ \text{subject to:} \quad & A^{\mathsf{T}} \lambda \ge c \\ & \lambda \ge 0 \end{aligned}$$

Suppose we add a small e to the constraint vector b.

- The optimal $x^\star$ (and therefore $p^\star$) may change, since we are changing the feasible set of (P). Call new values $\hat{x}^\star$ and $\hat{p}^\star$.
- As long as e is small enough, the optimal λ will not change, since the feasible set of (D) is the same.
- Before: $p^\star = b^{\mathsf{T}} \lambda^\star$. After: $\hat{p}^\star = b^{\mathsf{T}} \lambda^\star + e^{\mathsf{T}} \lambda^\star$
- Therefore: $(\hat{p}^\star - p^\star) = e^{\mathsf{T}} \lambda^\star$. Letting $e \to 0$, $\nabla_b (p^\star) = \lambda^\star$.

Sensitivity of Top Brass



$$\max_{f,s} \quad 12f + 9s$$

$$\text{s.t.} \quad 4f + 2s \leq 4800$$

$$f + s \leq 1750$$

$$0 \leq f \leq 1000$$

$$0 \leq s \leq 1500$$

Constraints that are loose at optimality have corresponding dual variables that are zero; those items aren't *worth* anything.

Complementary slackness

- At the optimal point, some inequality constraints become *tight*. Ex: wood and plaque constraints in Top Brass.
- Some inequality constraints may remain loose, even at optimality. Ex: brass football/soccer ball constraints. These constraints have *slack*.

> Either a primal constraint is tight **or** its dual variable is zero.

The same thing happens when we solve the dual problem. Some dual constraints may have slack and others may not.

> Either a dual constraint is tight **or** its primal variable is zero.

These properties are called *complementary slackness*.

Proof of complementary slackness

- **Primal**: $\max_x \ c^\top x$ s.t. $Ax \leq b, \ x \geq 0$
- **Dual**: $\min_\lambda \ b^\top \lambda$ s.t. $A^\top \lambda \geq c, \ \lambda \geq 0$

Suppose (x, λ) is feasible for the primal and the dual.

- Because $Ax \leq b$ and $\lambda \geq 0$, we have: $\lambda^\top Ax \leq b^\top \lambda$.
- Because $c \leq A^\top \lambda$ and $x \geq 0$, we have: $c^\top x \leq \lambda^\top Ax$.

Combining both inequalities: $c^\top x \leq \lambda^\top Ax \leq b^\top \lambda$.

By strong duality, $c^\top x^\star = {\lambda^\star}^\top A x^\star = b^\top \lambda^\star$

Proof of complementary slackness

$$c^\mathsf{T} x^\star = {\lambda^\star}^\mathsf{T} A x^\star = b^\mathsf{T} \lambda^\star$$

$u_i v_i = 0$ means that: $u_i = 0$, or $v_i = 0$, or *both*.

The first equation says: ${x^\star}^\mathsf{T}(A^\mathsf{T}\lambda^\star - c) = 0$.
But $x^\star \geq 0$ and $A^\mathsf{T}\lambda^\star \geq c$, therefore:

$$\sum_{i=1}^{n} x_i^\star (A^\mathsf{T}\lambda^\star - c)_i = 0 \quad \Longrightarrow \quad x_i^\star (A^\mathsf{T}\lambda^\star - c)_i = 0 \quad \forall i$$

Similarly, the second equation says: ${\lambda^\star}^\mathsf{T}(A x^\star - b) = 0$.
But $\lambda^\star \geq 0$ and $A x^\star \leq b$, therefore:

$$\sum_{j=1}^{m} \lambda_j^\star (A x^\star - b)_j = 0 \quad \Longrightarrow \quad \lambda_j^\star (A x^\star - b)_j = 0 \quad \forall j$$

Another simple example

Primal problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x_1 + x_2 \\ \text{subject to:} \quad & 2x_1 + x_2 \geq 5 \\ & x_1 + 4x_2 \geq 6 \\ & x_1 \geq 1 \end{aligned}$$

Dual problem:

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & 5\lambda_1 + 6\lambda_2 + \lambda_3 \\ \text{subject to:} \quad & 2\lambda_1 + \lambda_2 + \lambda_3 = 1 \\ & \lambda_1 + 4\lambda_2 = 1 \\ & \lambda_1, \lambda_2, \lambda_3 \geq 0 \end{aligned}$$

Question: Is the feasible point $(x_1, x_2) = (1, 3)$ optimal?

- Second primal constraint is slack, therefore $\lambda_2 = 0$.
- Solving dual equations gives $\lambda_1 = 1, \lambda_2 = 0, \lambda_3 = -1$
- The only dual solution satisfying complementary slackness is not feasible: This does not satisfy $\lambda_i \geq 0$

$(1, 3)$ is **not optimal** for the primal.

Another simple example

Primal problem:

$$\begin{array}{rl} \underset{x}{\text{minimize}} & x_1 + x_2 \\ \text{subject to:} & 2x_1 + x_2 \geq 5 \\ & x_1 + 4x_2 \geq 6 \\ & x_1 \geq 1 \end{array}$$

Dual problem:

$$\begin{array}{rl} \underset{\lambda}{\text{maximize}} & 5\lambda_1 + 6\lambda_2 + \lambda_3 \\ \text{subject to:} & 2\lambda_1 + \lambda_2 + \lambda_3 = 1 \\ & \lambda_1 + 4\lambda_2 = 1 \\ & \lambda_1, \lambda_2, \lambda_3 \geq 0 \end{array}$$

Another question: Is $(x_1, x_2) = (2, 1)$ optimal?

- Third primal constraint is slack, therefore $\lambda_3 = 0$.
- Costs should match, so $5\lambda_1 + 6\lambda_2 = 3$.
- Solving dual constraints gives: $\lambda_1 = \frac{3}{7}$, $\lambda_2 = \frac{1}{7}$, $\lambda_3 = 0$, which is dual feasible!

$(2, 1)$ is **optimal** for the primal. (Objective values are =!)