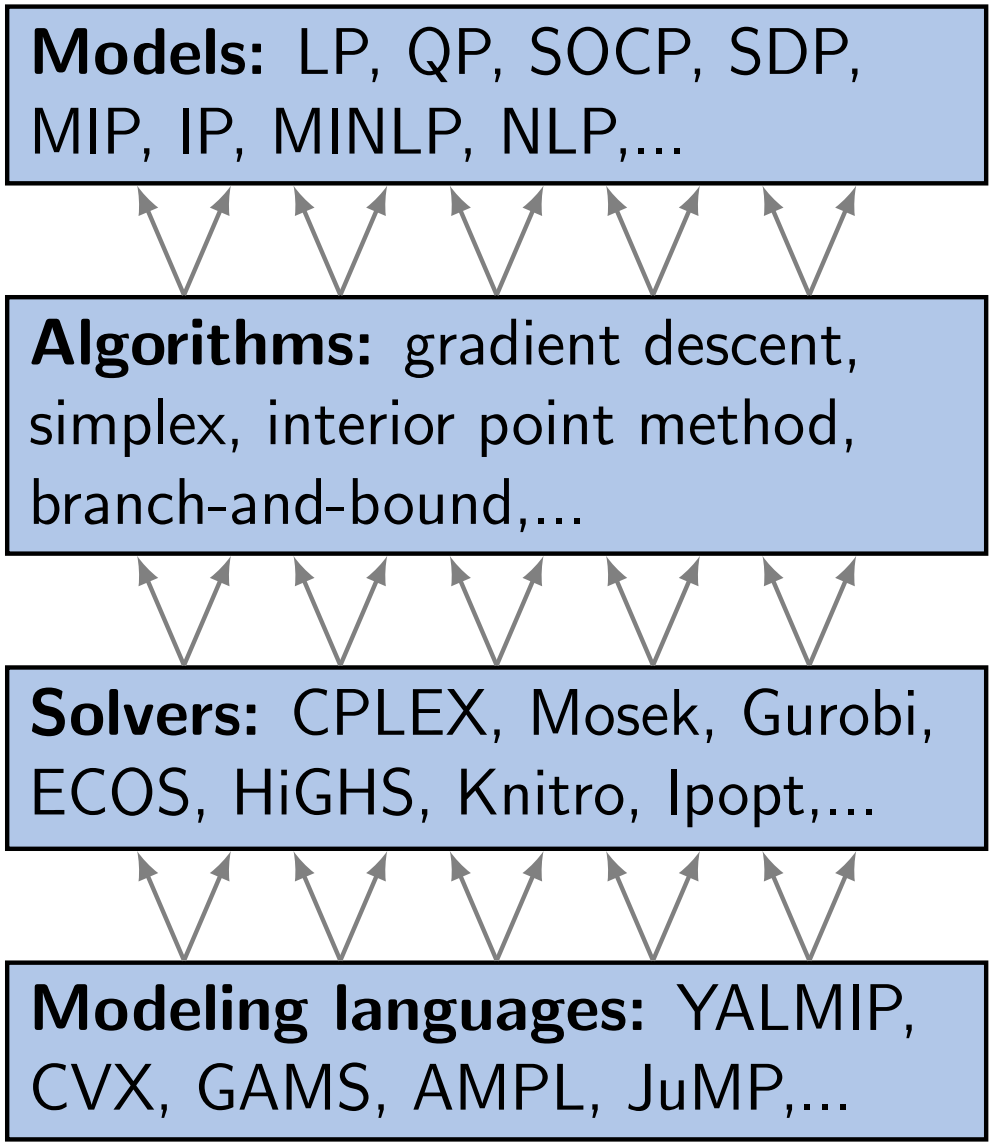
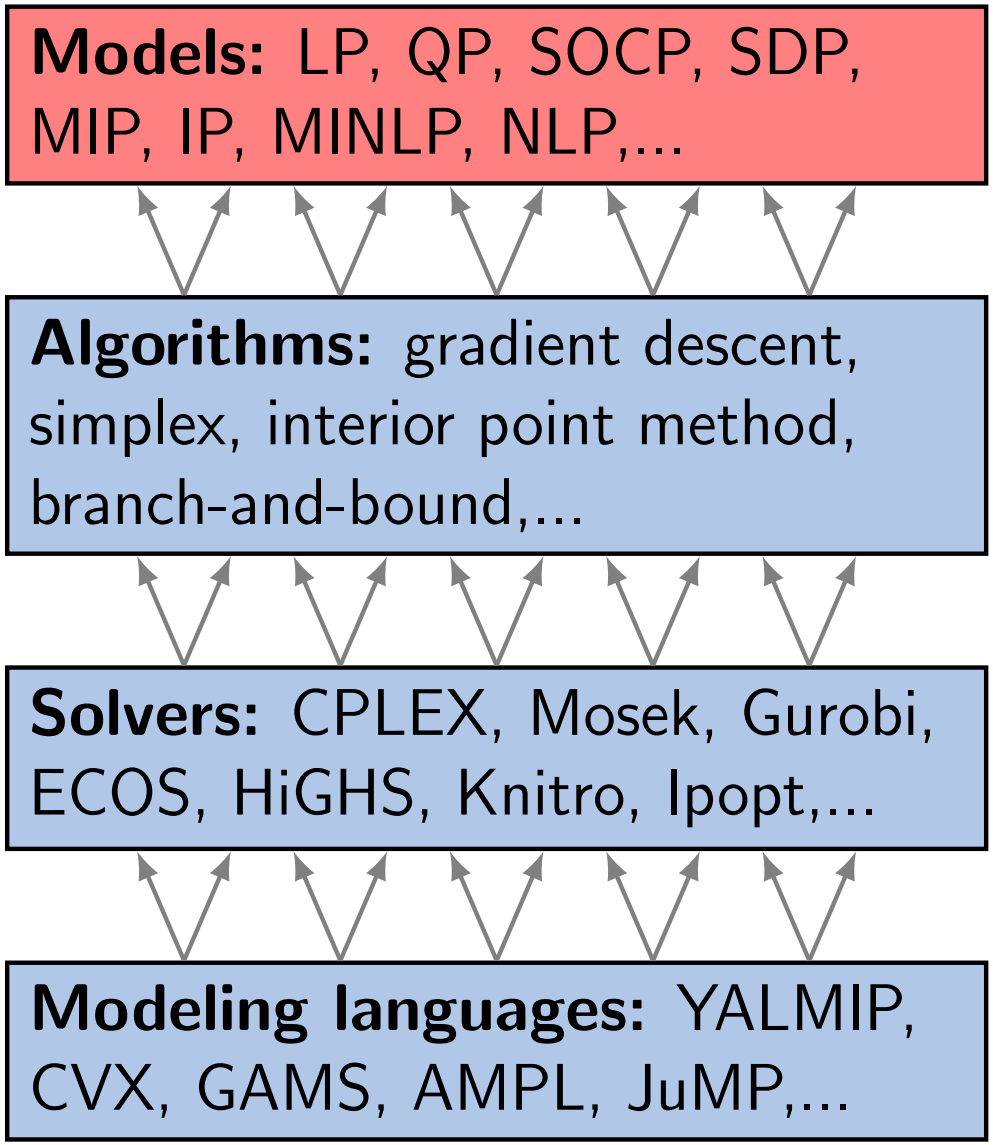
CS/ECE/ISYE524: Introduction to Optimization – Linear Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

January 29, 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.

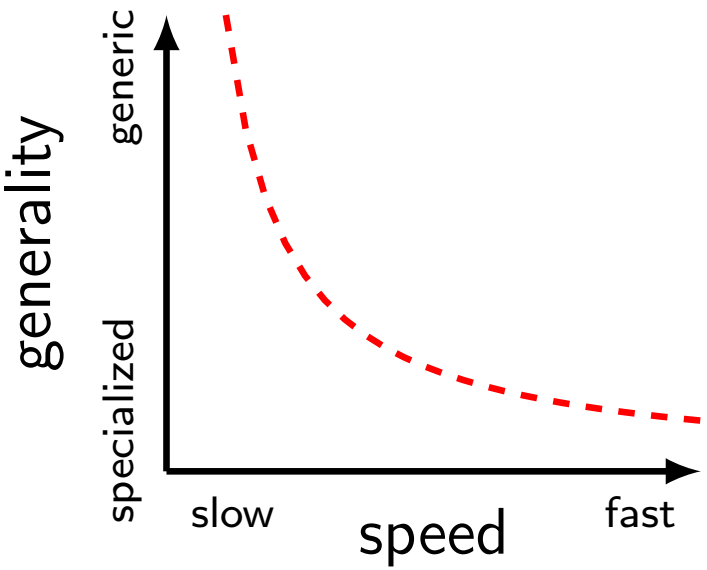
Models: LP, QP, SOCP, SDP, MIP, IP, MINLP, NLP,...

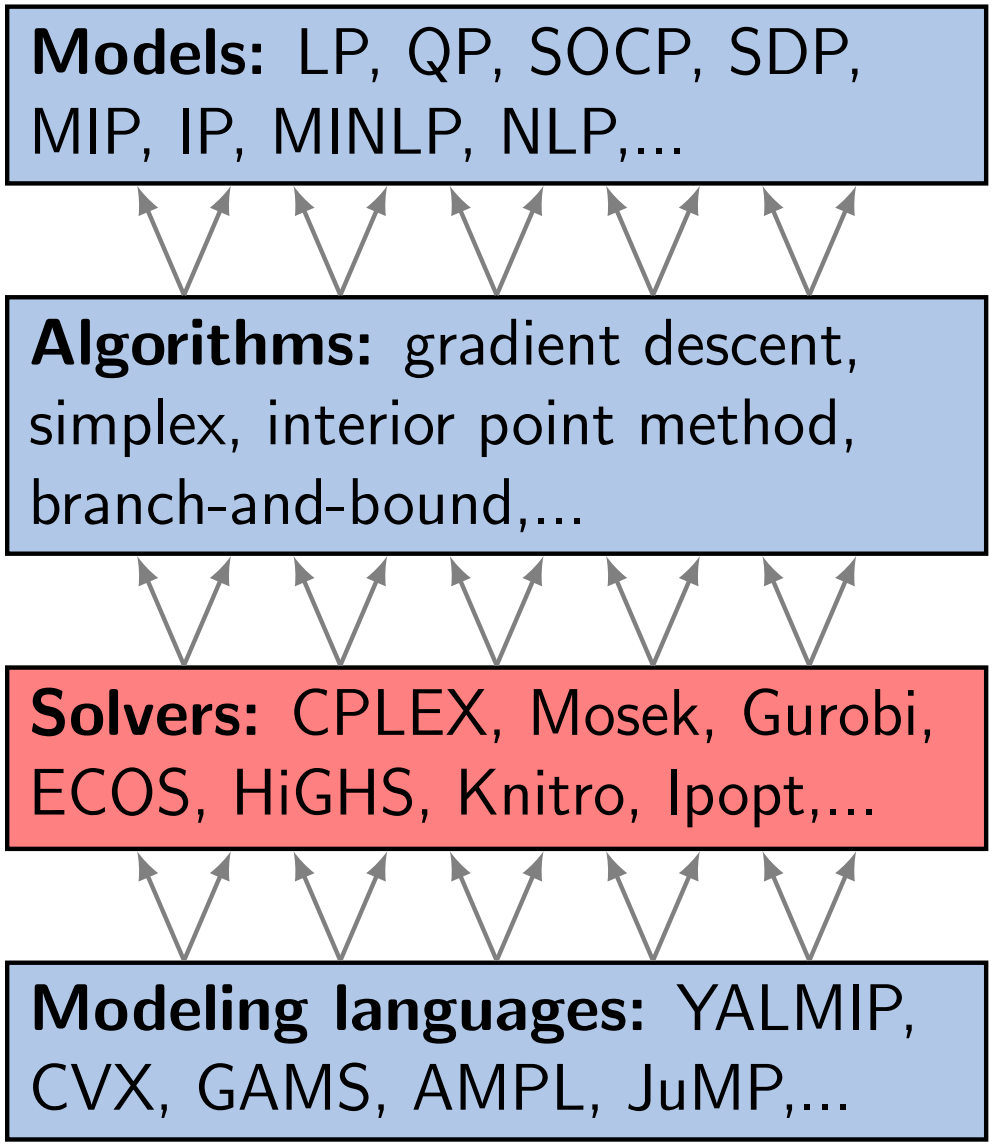
Algorithms: gradient descent, simplex, interior point method, branch-and-bound,...

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.



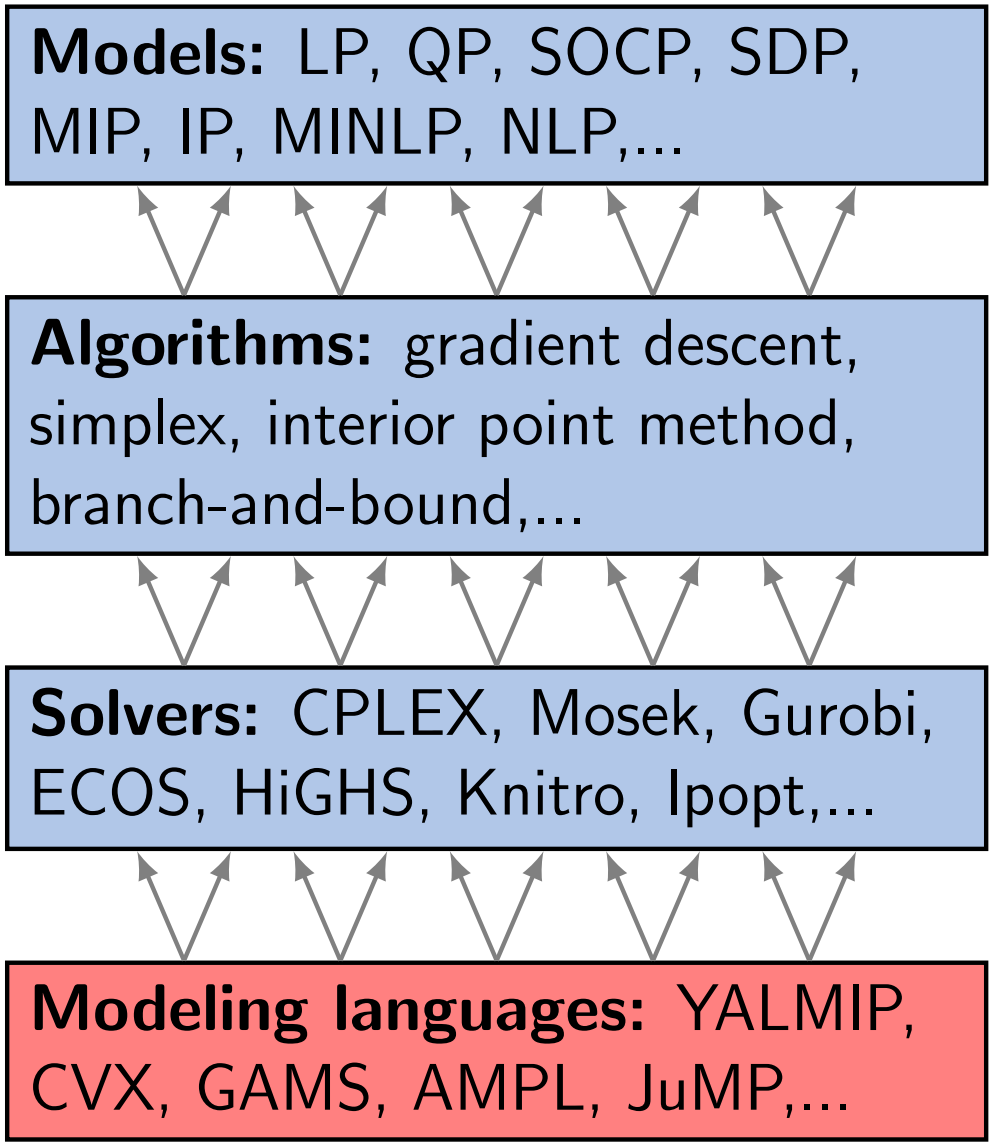


Solvers are *implementations* of algorithms. Sometimes they can be quite clever!

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

Availability varies:

- some are open-source
- some are commercial



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

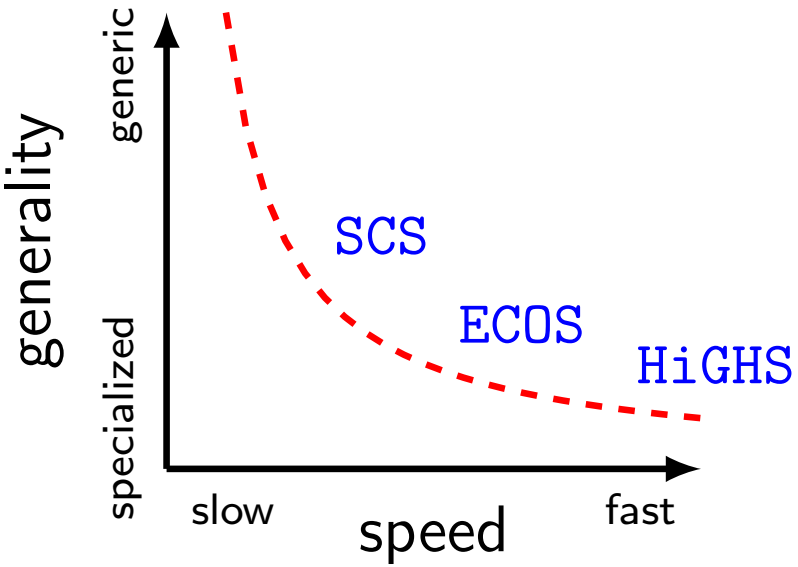
- Try `HiGHS`, `ECOS`, `SCS` solvers. Is the answer the same?
- Compare solvers using the `@time` macro
- What happens if an unsuitable solver is used?

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)

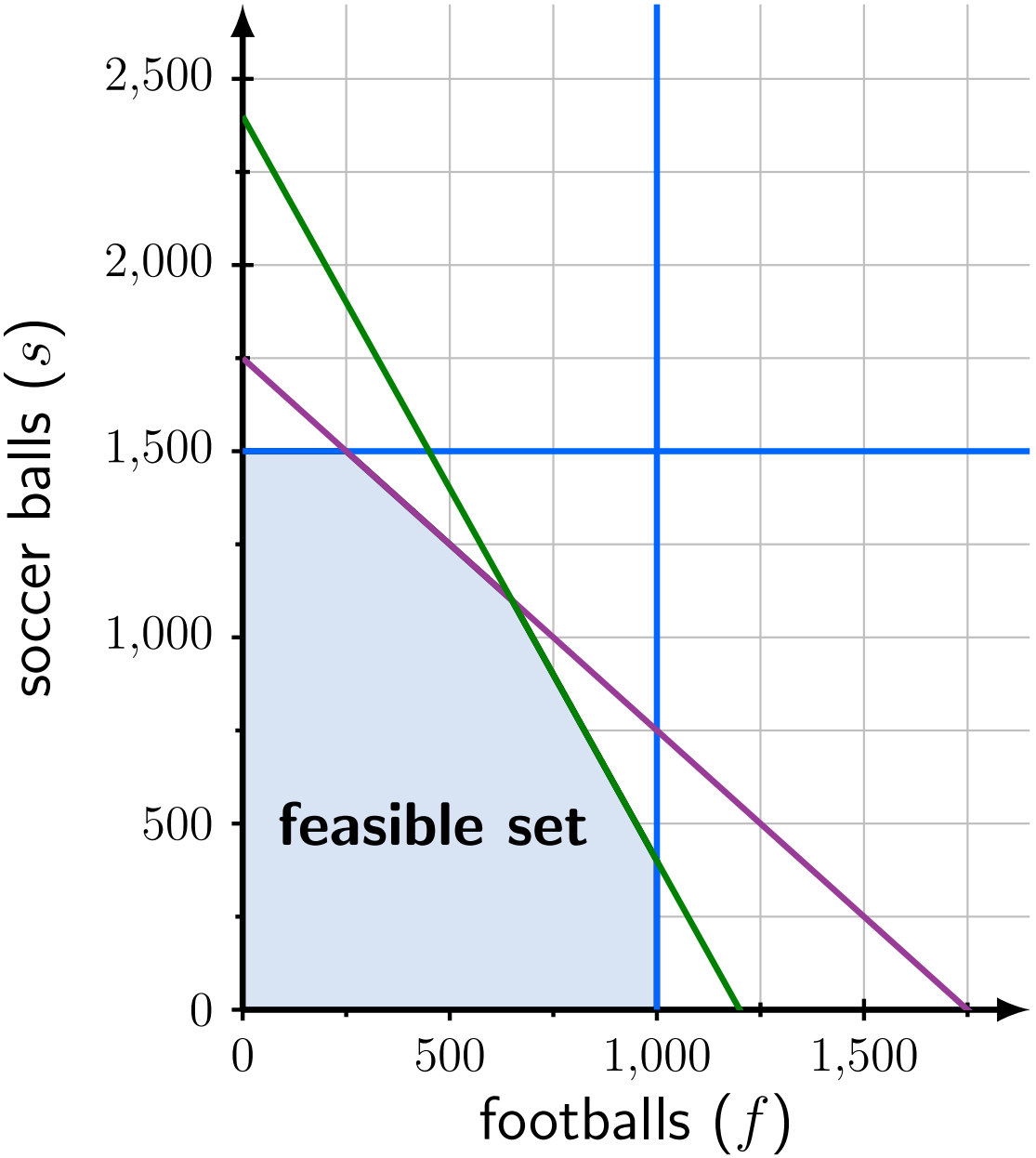
JuMP and GAMS are structurally very similar

```
* 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)

```
* 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;
```

JuMP and GAMS are structurally very similar



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

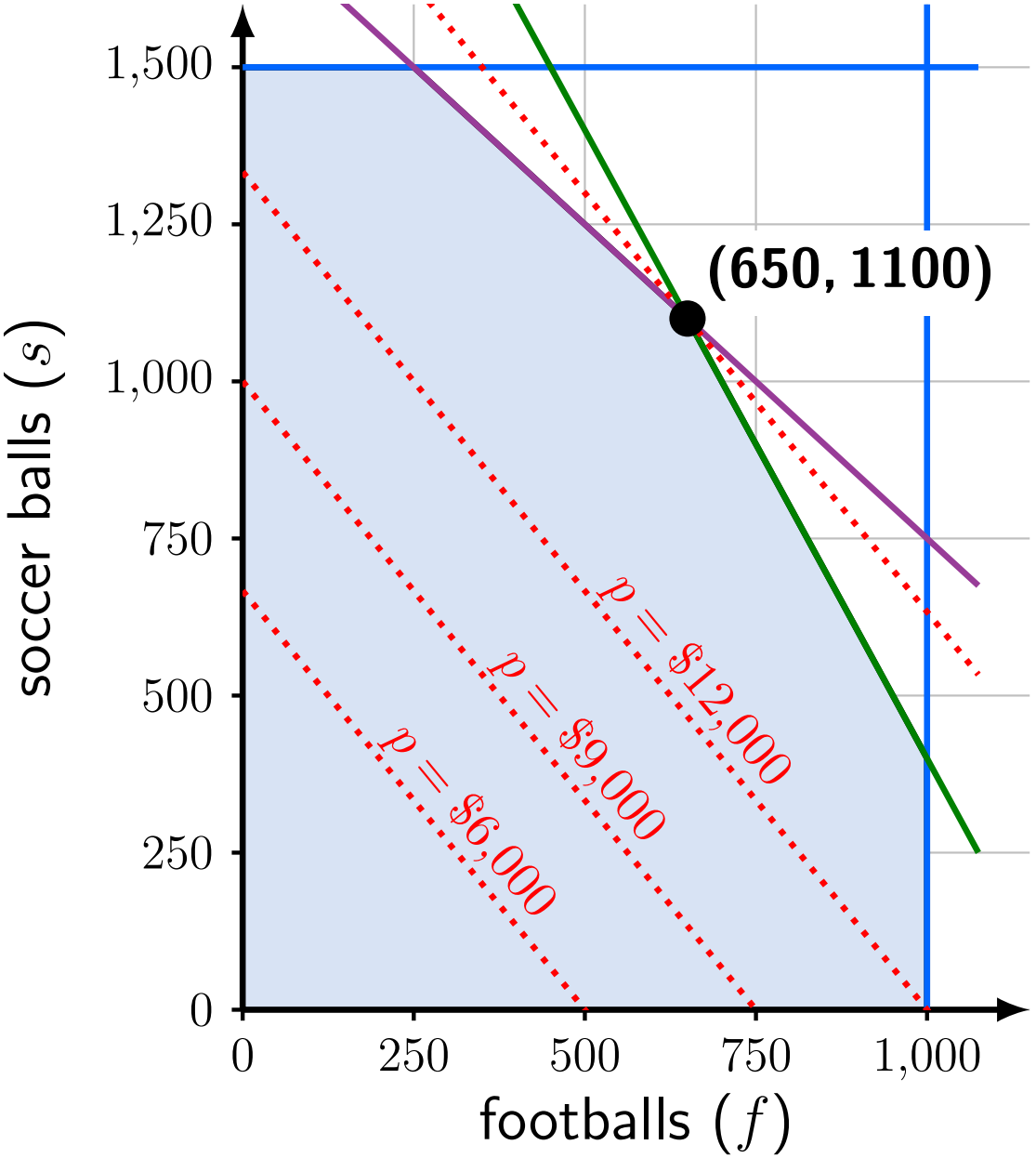
$$\text{s.t.} \quad 4f + 2s \leq 4800$$

$$f + s \leq 1750$$

$$0 \leq f \leq 1000$$

$$0 \leq s \leq 1500$$

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



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

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

$$f + s \leq 1750$$

$$0 \leq f \leq 1000$$

$$0 \leq s \leq 1500$$

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

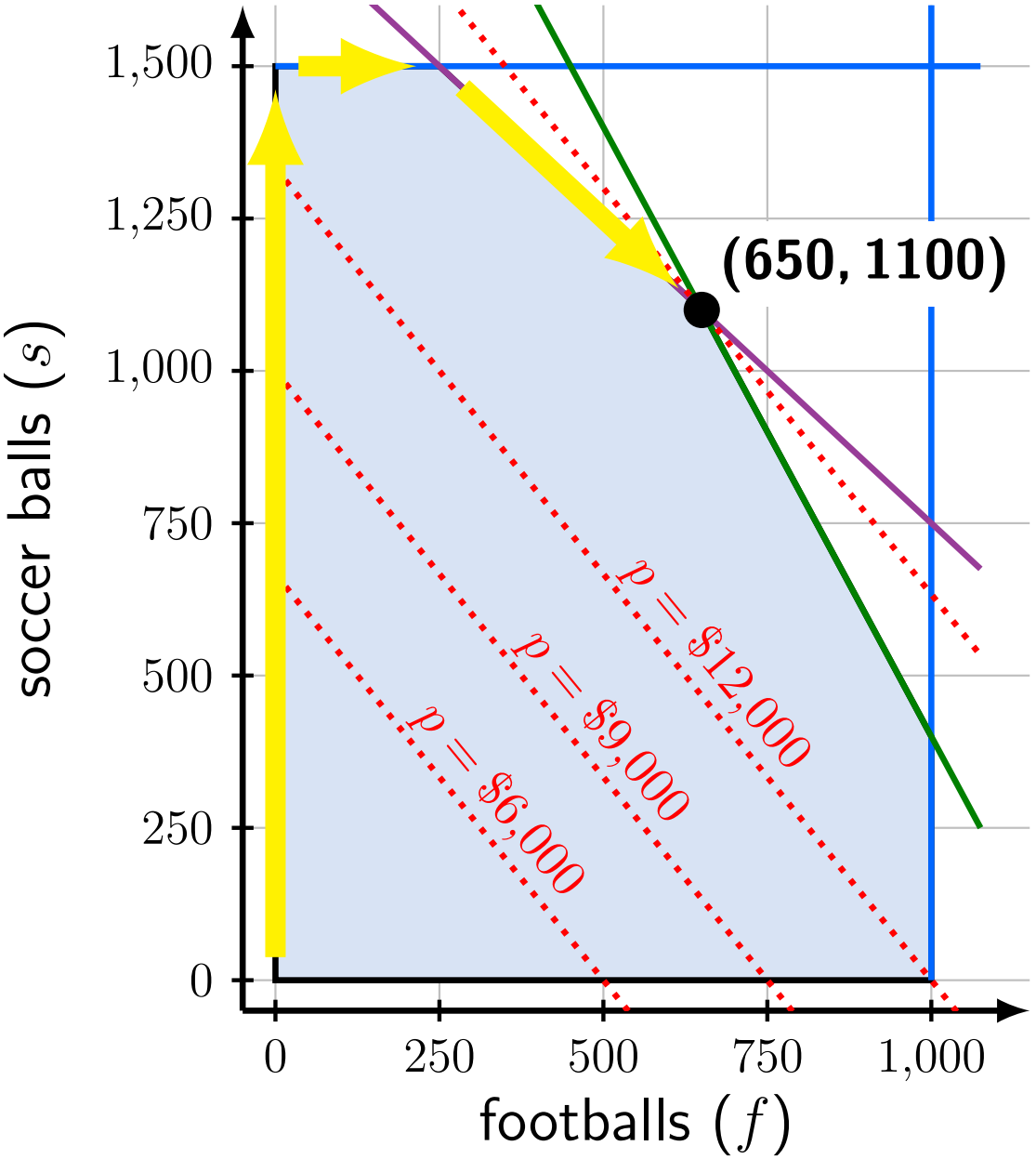
0. Start from an extreme point.
1. Find an improving direction d. If none exists, STOP. The extreme point is an optimal solution.
2. Move along d until you hit a new extreme point. Go to 1.

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).



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

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

$$f + s \leq 1750$$

$$0 \leq f \leq 1000$$

$$0 \leq s \leq 1500$$

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

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