2. Introduction, part two

- Optimization hierarchy
- Available solvers in JuMP
- Writing modular code
- Geometrical intuition

Optimization hierarchy

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

Algorithms: gradient descent, simplex, interior point method, quasi-Newton methods,...

Solvers: CPLEX, Mosek, Gurobi, ECOS, Clp, 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.

Optimization hierarchy

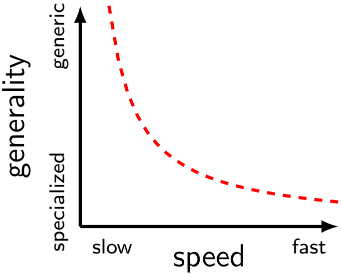
Models: LP, QP, SOCP, SDP, MIP, IP, MINLP, NLP,...

Algorithms: gradient descent, simplex, interior point method, quasi-Newton methods,...

Solvers: CPLEX, Mosek, Gurobi, ECOS, Clp, 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.



Optimization hierarchy

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

Algorithms: gradient descent, simplex, interior point method, quasi-Newton methods,...

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

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

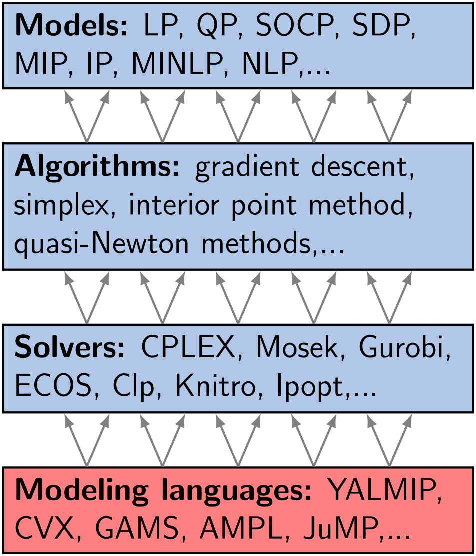
Solvers are *implementations* of algorithms. Sometimes they can be quite clever!

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

Availability varies:

- some are open-source
- some are commercial
- some have .edu versions

Optimization hierarchy



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)

Again, availability varies:

- some are open-source
- some are commercial
- some have .edu versions

Solvers in JuMP

Solver	Julia Package	`solver=`	License	LP	SOCP	MILP	NLP	MINLP	SDP
Artelys Knitro	KNITRO.jl	`KnitroSolver()`	Comm.				X	X	
BARON	BARON.jl	`BaronSolver()`	Comm.				X	X	
Bonmin	AmplNLWriter.jl	`BonminNLSolver()` *	EPL	X		X	X	X	
	CoinOptServices.jl	`OsilBonminSolver()`							
Cbc	Cbc.jl	`CbcSolver()`	EPL			X			
Clp	Clp.jl	`ClpSolver()`	EPL	X					
Couenne	AmplNLWriter.jl	`CouenneNLSolver()` *	EPL	X		X	X	X	
	CoinOptServices.jl	`OsilCouenneSolver()`							
CPLEX	CPLEX.jl	`CplexSolver()`	Comm.	X	X	X			
ECOS	ECOS.jl	`ECOSSolver()`	GPL	X	X				
FICO Xpress	Xpress.jl	`XpressSolver()`	Comm.	X	X	X			
GLPK	GLPKMath...	`GLPKSolver[LP|MIP]()`	GPL	X		X			
Gurobi	Gurobi.jl	`GurobiSolver()`	Comm.	X	X	X			
Ipopt	Ipopt.jl	`IpoptSolver()`	EPL	X			X		
MOSEK	Mosek.jl	`MosekSolver()`	Comm.	X	X	X	X		X
NLopt	NLopt.jl	`NLoptSolver()`	LGPL				X		
SCS	SCS.jl	`SCSSolver()`	MIT	X	X				X

Source: http://www.juliaopt.org/JuMP.jl/0.18/installation.html

Solvers in JuMP

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

```
using JuMP, Clp, ECOS, SCS
m = Model(solver = ClpSolver())
m = Model(solver = ECOSSolver())
m = Model(solver = SCSSolver())
```

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

```
m = Model()
setsolver(m, ClpSolver())
solve(m)
setsolver(m, ECOSSolver())
solve(m)
```

Solvers in JuMP

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

Every solver must be installed before it can be used: `Pkg.add("Clp")`

Some things to know:

- Installing a package may take a couple minutes, 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

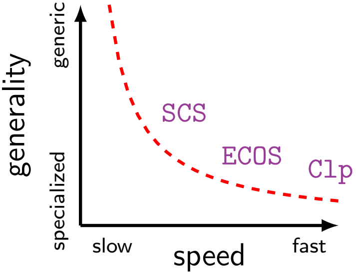
- Try `Clp`, `ECOS`, `SCS` solvers. Is the answer the same?
- Compare solvers using the `@time(...)` macro
- What happens if an unsuitable solver is used?

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*.

Top Brass 2.ipynb , Top Brass 3.ipynb

- Use *dictionaries* to make the code more modular
- Use *expressions* to make the code more readable
- Use `NamedArrays` for indexing over sets
- Try adding a new type of trophy!

Comparison: GAMS (1)

JuMP and GAMS are structurally very similar

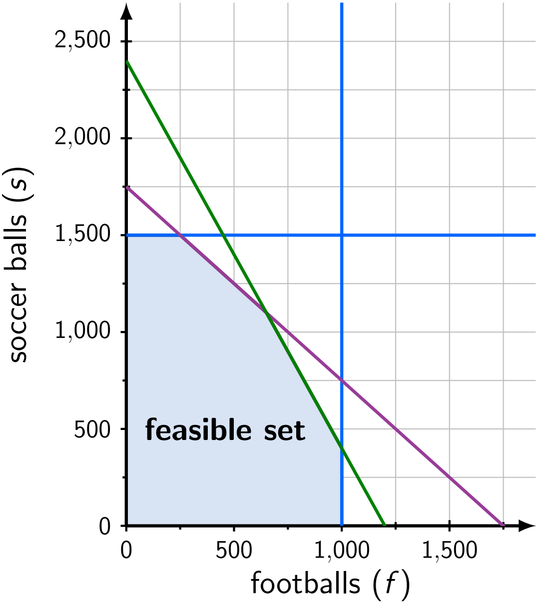
```
* TOP BRASS PROBLEM
set I/football, soccer/;
free variable profit "total profit";
positive variables x(I) "trophies";
* DATA section
parameters
        profit(I)   / "football" 12 , "soccer" 9 /
        wood(I)     / "football"  4 , "soccer" 2 /
        plaques(I)  / "football"  1 , "soccer" 1 /;
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

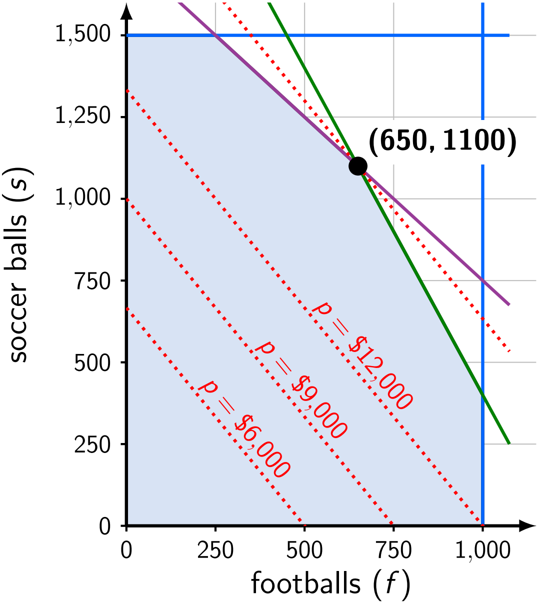
Geometry of Top Brass



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

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

Geometry of Top Brass



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

Which feasible point has the max profit?

$p = 12f + 9s$