2. Introduction, part two

- Optimization hierarchy
- Available solvers
- 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 cost function

Example: every linear program (LP) has:
- continuous variables
- linear constraints
- a linear cost function

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

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,...

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

<table>
<thead>
<tr>
<th>Solver</th>
<th>Julia Package</th>
<th>License</th>
<th>LP</th>
<th>SOCP</th>
<th>MILP</th>
<th>NLP</th>
<th>MINLP</th>
<th>SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artelys Knitro</td>
<td>KNITRO.jl</td>
<td>Comm.</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BARON</td>
<td>BARON.jl</td>
<td>Comm.</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonmin</td>
<td>AmplNLWriter.jl</td>
<td>EPL</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CoinOptServices.jl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cbc</td>
<td>Cbc.jl</td>
<td>EPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Clp</td>
<td>Clp.jl</td>
<td>EPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Couenne</td>
<td>AmplNLWriter.jl</td>
<td>EPL</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CoinOptServices.jl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPLEX</td>
<td>CPLEX.jl</td>
<td>Comm.</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ECOS</td>
<td>ECOS.jl</td>
<td>GPL</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FICO Xpress</td>
<td>Xpress.jl</td>
<td>Comm.</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GLPK</td>
<td>GLPKMath...</td>
<td>GPL</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gurobi</td>
<td>Gurobi.jl</td>
<td>Comm.</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Ipopt</td>
<td>Ipopt.jl</td>
<td>EPL</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOSEK</td>
<td>Mosek.jl</td>
<td>Comm.</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>NLopt</td>
<td>NLopt.jl</td>
<td>LGPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SCS</td>
<td>SCS.jl</td>
<td>MIT</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: https://jump.readthedocs.org/en/latest/installation.html
Solvers in JuMP

If you use the default model `m = Model()`,

- JuMP determines your model type (e.g. LP, MIP, etc.) and selects an appropriate solver.

- If there are many suitable solvers, JuMP picks the top solver on its internal list.

- Can check which solver was used: `internalmodel(m)`

You can tell JuMP which solver to use. For example:

```plaintext
using JuMP, Clp, ECOS, SCS
m1 = Model(solver = ClpSolver())
m2 = Model(solver = ECOSSolver())
m3 = Model(solver = SCSSolver())
```
Solvers in JuMP

Top Brass.ipynb

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

* 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";

JuMP and GAMS are structurally very similar
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 \quad 12f + 9s

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.

feasible set
Geometry of Top Brass

Which feasible point has the max profit?

\[ p = 12f + 9s \]