Introduction to AMPL and NEOS

Piotr Krzyżanowski

How do you solve real life optimization problems?

- Write your own code (in C or MATLAB) from scratch? Very bad idea!
- Use external numerical library which provides specialized solvers for various types of optimization problems[1]? Makes sense.
- Turn to DSL (Domain Specific Language) to describe the problem in a human-like way, then feed a solver with this description and wait for the result. In many cases, this is both the easiest and most robust approach — especially if you are *not* a numerical analyst!

AMPL is a DSL for optimization problems of small-to-medium (or even large) size.

NEOS Server offers free(!) access to a collection of various modern — including state-of-the-art commercial(!) — optimization solvers. Most of them accept AMPL input.

What is AMPL?

- AMPL stands for "**A** **M**athematical **P**rogramming **L**anguage"
- "*Mathematical programming*" is an old phrase for "*optimization*"
- AMPL was invented by R.Fourer, D.Gay and B.Kernighan[2] from Bell Labs in… 1985(!)
- Today it is still the most popular way to formulate optimization problems on NEOS Server
- Essentially every reasonable optimization solver reads AMPL!
- AMPL translator is a propertiary software. It interfaces external solvers to actually compute the solution.

Learning AMPL on a *single* easy example

Based on

- Introduction to AMPL. A Tutorial, originally prepared by Phil Kaminsky at Berkeley and modified by Deepak Rajan.
- The Diet Problem. on the NEOS Guide webpage.

Optimization "`Hello, world!`" task

Linear programming with constraints

Given vectors $c \in R^N, b \in R^M$ and matrix $A \in R^{M \times N}$, find $x \in R^N$ which minimizes

$$c^T x,$$

[1]There is no single *best-for-all* solver in mathematical optimization.
[2]The co-inventor of the **C language** (sic!) and AWK.

subject to conditions

$$Ax \leq b.$$

Linear programming with constraints

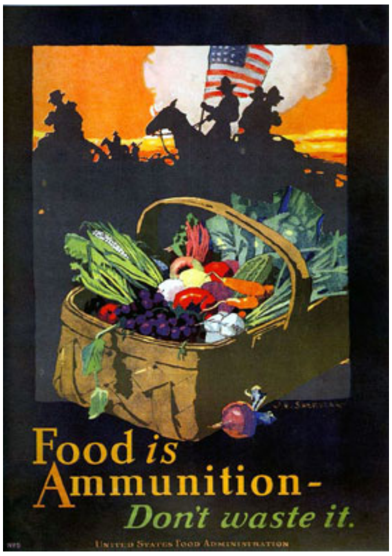
Example. (*Source: NEOS Guide*) The Diet Problem. Suppose there are three foods available: corn, milk, and bread, and there are restrictions on the number of calories (between 2000 and 2250) and the amount of Vitamin A (between 5000 and 50,000). The table lists, for each food, the cost per serving, the amount of Vitamin A per serving, and the number of calories per serving:

Food	Cost per serving	Vitamin A	Calories
Corn	0.18	107	72
Milk	0.23	500	121
Bread	0.05	0	65

Suppose that the maximum number of servings is 10.

The goal: Find the number of servings of each food to purchase (and consume) so as to minimize the cost of the food while meeting the specified nutritional requirements.

The Diet Problem



Rysunek 1: As seen by the army (Source: NEOS Guide)

The Diet Problem (cntd)

c, m, b — number of servings of corn, milk, bread, respectively.

We want to minimize

$$\text{cost} = 0.18c + 0.23m + 0.05b$$

subject to constraints

$$
\begin{aligned}
0 \le c, m, b &\le 10 && \text{(servings)} \\
5000 \le 107c + 500m &\le 50000 && \text{(vit. A)} \\
2000 \le 72c + 121m + 65b &\le 2250 && \text{(calories)}
\end{aligned}
$$

The Diet Problem described in AMPL

```
var c;  # servings of corn
var m;  # servings of milk
var b;  # servings of bread

minimize cost: 0.18*c + 0.23*m + 0.05*b;

subject to c_servings:  0 <= c <= 10;
subject to m_servings:  0 <= m <= 10;
subject to b_servings:  0 <= b <= 10;
subject to vitA:      500 <= 107*c + 500*m <= 50000;
subject to calories: 2000 <= 72*c + 121*m+ 65*b <= 2250;
```

Seems you have already known AMPL, haven't you?

Remember to **end every line with a semicolon**.

Save the model to file with extension `.mod`.

Solving an AMPL model on the NEOS Server

Go to NEOS Server webpage.

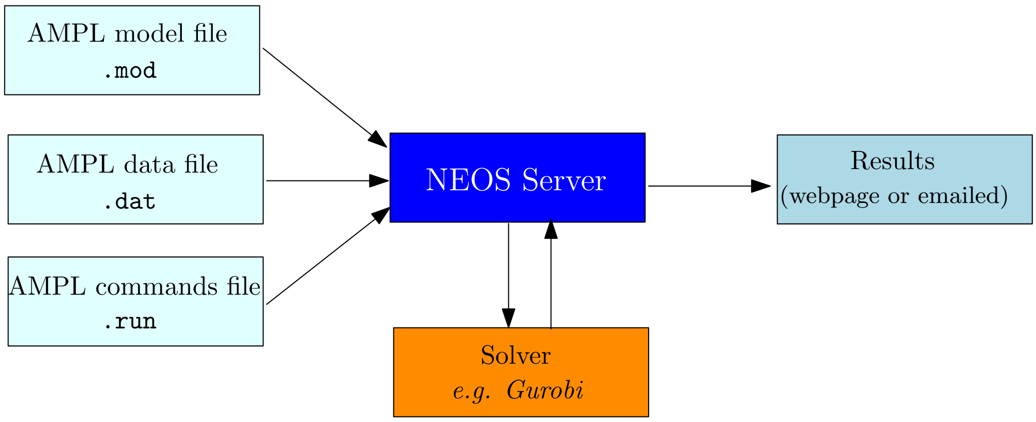
1. Set optimization problem type and solver.
   - We choose `Linear Programming` class and `Gurobi/AMPL` solver.
2. Provide Model File
   - We provide `diet.mod` with AMPL description of the problem
3. Provide Data and Command files
   - There is no specific data file in oure example, and the simplest (yet universal) command file is

   ```
   solve;                         # run the solver
   display _varname, _var;        # print the results
   ```
4. Submit and wait for the results!
   - Our problem is very easy, so we also opt for `Short Priority` queue with maximum CPU time of 5 minutes.

Solving an AMPL model on the NEOS Server

The Diet Problem solution

After waiting a couple of moments for the webpage to reload, we end up with



Rysunek 2: Workflow on NEOS server

```
*************************************************************
   NEOS Server Version 5.0
   Job#     : 8470919
   ...skipped lines...
*************************************************************
You are using the solver gurobi_ampl.
...more skipped lines...
Gurobi 9.0.1: optimal solution; objective 3.15
: _varname     _var      :=
1   c          1.94444
2   m         10
3   b         10
```

So the optimal solution is to serve the maximum allowed amount of milk and bread, accompanied with 1.94444 portions of corn. The total cost is then as low as \$3.15.

More abstract formulation of The Diet Problem

Consider N foods and for $i = 1, \ldots, N$ define:

- x_i — number of servings of i-th food to purchase
- c_i — cost of i-th food per serving (in dollars)
- v_i — vit. A content in i-th food per serving (in units)
- n_i — nutrition value of i-th food per serving (in calories)
- M_i — max. number od servings of i-th food

We want to minimize

$$\text{cost} = \sum_{i=1}^{N} c_i x_i$$

subject to constraints

$$
\begin{aligned}
0 \le x_i &\le S_i & i &= 1, \ldots, N \\
V_{\min} \le \sum_{i=1}^{N} v_i x_i &\le V_{\max} & &\text{(vit. A)} \\
N_{\min} \le \sum_{i=1}^{N} n_i x_i &\le N_{\max} & &\text{(calories)}
\end{aligned}
$$

More abstract formulation of The Diet Problem in AMPL

```
param N;
param c{i in 1..N}; # cost of ith food per serving (in dollars)
param v{i in 1..N}; # vit. A content in ith food per serving (in units)
param n{i in 1..N}; # nutrition value of ith food per serving (in calories)
param M{i in 1..N}; # max number of ith food serving
param Vmin;
param Vmax;
param Nmin;
param Nmax;

var x{i in 1..N};    # number of ith food servings

minimize cost: sum{i in 1..N} c[i]*x[i];

subject to servings{i in 1..N}:     0 <= x[i] <= M[i];
subject to vitA:      Vmin <= sum{i in 1..N} v[i]*x[i] <= Vmax;
subject to calories:  Nmin <= sum{i in 1..N} n[i]*x[i] <= Nmax;
```

AMPL data file for the abstract formulation of The Diet Problem

```
param N :=  3;
param c :=  1   0.18
            2   0.23
            3   0.05; # cost of ith food per serving (in dollars)
param v :=  1   107
            2   500
            3   0; # vit. A content in ith food per serving (in units)
param n :=  1   72
            2   121
            3   65; # nutrition value of ith food per serving (in calories)
param M :=  1   10
            2   10
            3   10; # max number of ith food serving
param Vmin := 500;
```

```
param Vmax := 50000;
param Nmin := 2000;
param Nmax := 2250;
```

The Diet Problem with integer/binary number of servings

What if we can only serve whole portions? Will *two* (=round(1.9444)) portions of corn be the optimal solution?

We need to add **integer type** constraint to our LP problem. This makes the problem *computationally* much harder!

But in its AMPL description we only have to change **one** line:

```
var x{i in 1..N} integer; # number of ith food servings (INTEGER!!)
```

Sometimes we want to further restrict allowed values of the unknown to either `0` or `1` ("yes-no", or **binary type** variables):

```
var x{i in 1..N} binary; # number of ith food servings (only 0 or 1)
```

Even more abstract formulation of The Diet Problem

Check out on the NEOS Guide web page on the Diet Problem or read Sections 4 and 5 of Introduction to AMPL. A Tutorial.

```
set F;
set N;

param a{F,N} >= 0;
param c{F} >= 0;
param Fmin{F} >= 0;
param Fmax{i in F} >= Fmin[i];
param Nmin{j in N} >= 0;
param Nmax{j in N} >= Nmax[j];

var x{i in F} >= Fmin[i];

minimize cost: sum {i in F} c[i]*x[i];

subject to nutritional_reqs{j in N}:
    Nmin[j] <= sum {i in F} amt[i,j]*x[i] <= Nmax[j];
```

Learning more AMPL

- Section 6 of Introduction to AMPL. A Tutorial.
- AMPL: A Modeling Language for Mathematical Programming. This is ***the*** AMPL book. Free to read.

Beyond AMPL

There are other languages to describe optimization problems, some of them tailored to specific solvers:

- Pyomo — open source AML[3] for Python
- GAMS — first AML, invented in 1976
- JuMP — AML for Julia
- Gurobi Python API — AML for Gurobi propertiary solver, aimed at Python users.

Gurobi for MIMUW students

We have access to a free *academic* license on `students.mimuw.edu.pl` (read the terms carefully).

- Set your environmental variables in `.bashrc`:

```
export GUROBI_HOME="/opt/gurobi"
export PATH="${PATH}:${GUROBI_HOME}/bin"
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${GUROBI_HOME}/lib"
```

Gurobi for MIMUW students (cntd)

- Use Gurobi in many ways:
  - from the command line:

  ```
  gurobi_cl $GUROBI_HOME/examples/data/coins.lp
  ```

  - interactively, using the Gurobi shell:

  ```
  gurobi.sh
  m = read('/opt/gurobi/examples/data/coins.lp')
  m.optimize()
  ```

  - from Python:

  ```
  python3 $GUROBI_HOME/examples/python/mip1.py
  ```

- Sad but true warning: Gurobi–AMPL interface **must** be purchased separately.

Learn more from Gurobi webpage: tutorials, manuals, etc..

[3] AML = Algebraic Modeling Language.