LINMA 2471: Optimization models and methods

21 septembre 2015

1 Introduction : AMPL

AMPL (which stands for "A mathematical programming language") is an algebraic modelling language which enables the solving of high complexity problems for large (or small) scale models and will be used throughout this course. Knowing or identifying what type of problem we're dealing with is important and allows us to decide which will be the most adequate method to solve it. Certain methods work well with certain types of models, some are better than others, etc.
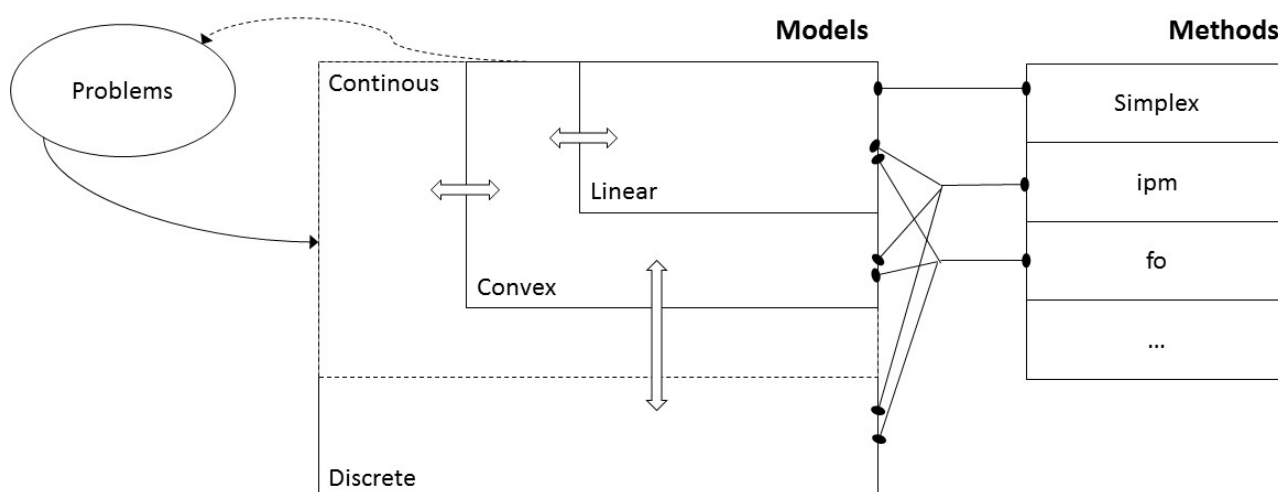

FIGURE 1 – Visualization of the relation between models and methods

AMPL works by first reading a text file which contains all the useful information of the model, this file usually ends in ".mod", it then parses it and tries to solve the problem. Parameters, variables, objective function and constraints are all defined in this file. The solving part is done by communicating with a solver, AMPL gives it all the information, the solver then sends back the solution. Consider the following optimization problem :

$$\min c^T x$$
$$Ay \leq b$$
$$l \leq y \leq u$$

For this problem AMPL will give the solver the usefull values (c,A,b,l,u,...) it neads to solve the problem. There exists quite a few of these solvers, some mode adapted to certain model types (linear, convex,...). We have for example :

— minos (basic solver for linear and nonlinear problems)
— cplex (can be used for linear, convex, mixted integer models)
— gurobi (very similar to cplex)
— knitro (good for nonlinear models)
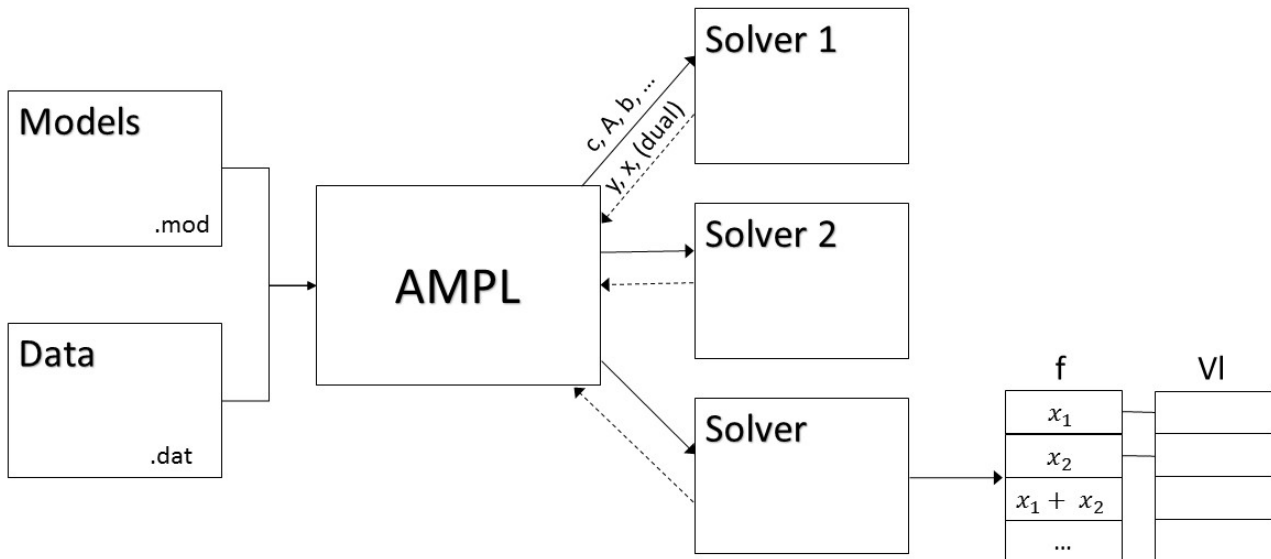— $\cdots$

FIGURE 2 –

AMPL can also work with an additionnal data file (".dat") which is used when parameters are left in the model file. This allows us to avoid changing the entire file when looking at different values of parameters, and only having to change them once in the data file.

A few examples and the basic syntax for AMPL can be found in the Tutorials Dropbox, given on Moodle.

Everything in AMPL has a name, whether it be variables, constants, or even constraints (which represent a dual variable) and each command and declaration ends with a semicolon (";"). Certain commands are worth being reminded here, for example :

— Changing solvers : **option solver … ;**

— Displaying dual variable : **display Protein ;**
— Displaying variable : **display Protein.body ;**
— Reset the whole model : **reset ;**
— Chosing a model file : **data data1.dat ;**
— Chosing a data file : **model data1.mod ;**
— Solving the chosen model : **solve ;**