Introduction

The What and Why of Modelling and Mathematical Programming

Romeo Rizzi, Alice Raffaele

University of Verona

romeo.rizzi@univr.it, alice.raffaele@unitn.it

October 13, 2020

Overview

From Problems to Models

Here is our first question:

- Find two numbers whose sum is 10 and whose difference is 2;
- Through basic **problems** like this one we have acquired the mathematical competence to introduce variables to compactly express a problem at hand in some algebraic language;
- Introducing variables x and y, the above question gets formulated as a linear system. Since Gauss thought about an efficient algorithm to solve them, linear systems have become one of the most important **models** offered by mathematics;
- The main point is: once you **formulate** your problem within a model you can then apply the whole body of knowledge developed for that model. And then ...
  Just casting the above problem in the language of linear system is regarded as *delivering its solution.*

Problems versus Models.
Submodels versus More General Models.

Time for another question: is it a problem or a model?

- Most of the time it is just a question of perspective!
  If you like solving problems, you might be lead to regard linear systems as a problem rather than as a model (*this is perfect, but don't let you miss the importance of using models*);
- AN EXAMPLE: The following **problem** is a **submodel** of linear systems which could suffice to our needs:
  INPUT: two numbers S and D;
  TASK: find two numbers whose sum is S and difference is D.

Models versus Problems
Problems versus Instances

Let's have a closer look at our example problem/model:

- INPUT: two numbers S and D;
- TASK: find two numbers whose sum is S and difference is D;
- Here S and D are parameters, which means that the above "general" problem (=model) actually embodies already an infinite number of problems, called **instances** of the general problem;
- **With big relevance:** these instances can all be encoded as pairs of numbers (S, D) and fed to a computer for solution.

Linear Programming (LP)

- Linear Programming (LP) is a powerful model which encompasses linear systems (i.e., having them as a submodel);
- INPUT: besides a system of linear constraints (not only equalities but also linear inequalities) over a given set of variables, you are also given a linear bijective function over the same variables (called decision variables);
- TASK: You are in seek of a feasible solution maximizing the given objective function value. (If you have no preferences over the possible solutions, just take a dull objective function);
- **IMPORTANT FACT:** since the end of the Second World War, efficient algorithms exist to solve the general LP problem.

Integer Linear Programming (ILP)

Here is a variation of our very first starting question:

- Find two **natural** numbers whose sum is 10 and whose difference is 2;
- Integer Linear Programming (ILP) is a powerful model which encompasses LP:
  INPUT: same as for LP; OUTPUT: same as for LP, but this time we are more restrictive on what we allow as solutions. Only assignments of **integer** values to the variables are allowed.
- **IMPORTANT FACTS:**
  - ILP encompasses LP **by too far**, in the sense that it is unfortunately NP-hard (no general efficient algorithm exists unless P=NP);
  - the decision version of ILP (i.e., when we have no objective function) is clearly in NP.

Good versus Bad news

Coming back into the surface: is it bad or good news that ILP is NP-hard?
Again, it depends on your perspective:

- If you like solving problems you might be lead to fall in desperation and cry your hottest tears;
- But if you are interested in making good money out from a job in mathematics, then things could get into another light.

The fact that ILP is NP-hard, with decision form in NP, means that all problems in NP (that is, all problems i**N**portant in the a**P**plications, i.e., when every "solvable" yes-instance admits a practical compact solution certificate whose verification lies in P) can be formulated within the ILP model.
Thus, ILP may well play the role of a sort of universal model/language for the truly interesting problems in applications like general optimal management.

A huge opportunity

As mathematicians, we can make a rewarded profession out of our proficiency in models like ILP.

More broadly, let's make a sport out of it

- Can you cast your problem of interest as optimizing some function (depending on a set of fixed parameters encoding it within a given objective function class) subject to a list of constraints (again encoded within a given class of possible constraints)?
- If yes, then your problem comprises an infinite number of possible instances. You can encode the instances, feed them to a computer, and design algorithms for their more-or-less-efficient solution;
- If yes, your problem can work as a possible model: if someone else has a problem that can be cast within your objective function class and within your family of possible constraints, then he can rest on your model and whatever theorem or algorithm has been obtained for it.

Mathematical Programming. What the hell is this?

- **Mathematical Programming** is the shared effort of the mathematical community in growing a body of more-or-less-well-studied models. You get a different model whenever you fix a different pair (objective function class, class of possible constraints);
- If this is not wide enough for your needs, be warned that we were only trying to make the point go through, but the share truth is another: Math. Prog. is a much wider effort than this, including topics like multi-objective optimization, stochastic optimization, robust optimization, etc. .

Main contents of this course (casual order)

- Basics of Linear Programming
- Mathematical Modelling (with AMPL/GMPL and Gurobi)
- Constraint Programming (with MiniZinc)
- Boolean Satisfiability (with GSAT)
- How to solve an optimization problem:
  - Exact methods
  - Approximation algorithms
- Polyhedral Combinatorics:
  - Basic notions
  - Branch & Cut
  - Complete and incomplete formulations
  - Gomory's cut and cutting planes
  - Separation oracles and callbacks
  - Compact formulations
- Game theory and Agent-based Simulation (with NetLogo)
- **Projects**

Goals

- **Design** mathematical models;
- **Experiment** computational and informatics skills;
- **Opportunities** to get in touch with working and/or research environments;
- **Connections** with professionals and disciplines for motivation.

The exam, but the course too

- **Project-based** (from industry, from other research centers or universities, from colleagues or on research lines of interest by the department, or even from ourselves included the students themselves): **group projects & cooperative learning**;
- Depending on the project, the following phases will be required as part of the exam or might naturally follow:
  - Study of a topic or subject;
  - Study of a technique;
  - Deployment, implementation, experiments and documentation;
  - Design of a didactic problem and exposition;
  - Writing a paper;
  - Stage, internship, thesis.
- We also encourage projects that contribute to more "technical" material we strive resorting onto in offering active and interactive learning experiences to our students.

Conclusion

- We realized that you already have a long career in modelling;
- But we also got crisper on what modelling actually is, on its strategic role within applied mathematics, and on some opportunities which lie out there;
- We gave a rough interpretation of the term "Mathematical Programming". This discipline is mainly concerned with models but in several different ways: from using models proficiently, to the proposing of them, to the study of the structural and algorithmic properties of models.