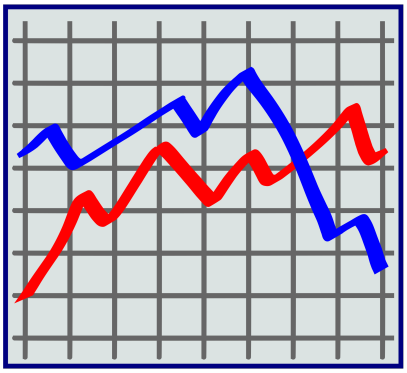
Syllabus for Computational OR

Ryo Kimura

March 20, 2019



Course Description

Computational experiments are a vital component of many operations research (OR) projects, providing empirical justification and practical insights for theoretical results. This course covers the key practical aspects of designing and conducting computational experiments in optimization, including model implementation, design and analysis of computational experiments, and customization of solver behavior (especially in implementing decomposition methods). We use mixed integer programming (MIP) models as our main framework. While this course is not meant to provide an introduction to coding, we will also briefly cover good coding practice.

Topics

- **Coding Practice**: fundamentals, coding tools (build systems, version control, documentation), debugging (debuggers, unit testing)
- **Model Implementation**: optimization solvers, basic implementation, solver parameters, modular design (DMAS framework)
- **Computational Experiments**: test instances, running experiments (metrics, timing, logging/parsing), performance analysis/plots
- **Solver Customization**: implementing decomposition methods (Dantzig-Wolfe, Benders, Lagrangian decomposition), callbacks (user/lazy cuts, primal heuristics, termination), branch-cut-and-price

In addition, one or more of the following topics may also be covered: modeling languages (AMPL/GAMS/OPL, JuMP), convex optimization (Mosek/CVX), constraint programming (CPO/Gecode, constraint-based scheduling, custom constraints), machine learning (scikit-learn/TensorFlow)

Prerequisites

This course assumes familiarity with basic MIP models and some coding background (at least one introductory programming course or equivalent). In addition, modules 10-12 assume slightly more background in integer programming (i.e., branch-and-bound and decomposition methods).

References

All course materials (tutorials/articles, relevant links, example code) will be posted on the course Canvas site. In addition to the references listed in the tutorials, the following references may be useful throughout the course:

- Winston, W. L., *Operations Research: Applications and Algorithms*, 4th Ed. (Cengage Learning, 2004) - **OR modeling**
- Severance, C., *Python for Everybody* (https://books.trinket.io/pfe/) - **Python programming**
- Conforti, M., Cornuéjols, G., and Zambelli, G., *Integer Programming* (Springer, 2014) - **integer programming theory**

Course Format and Evaluation

This course does not have scheduled class sessions/lectures. Instead, the instructor will upload tutorials/articles, relevant links, and example code to the course Canvas site weekly, which students will be expected to read in a timely manner.

Students are required to submit a course project utilizing computational OR. They may select a project from a number of predefined options, or they may propose their own project which can be aligned with their research.

Tentative Course Outline

Module 1	Fundamentals (SSH/SFTP, Linux, Bash, Python, C++)
Module 2	Optimization Software (problem classes, Gurobi, CPLEX)
Module 3	Basic Modeling (IP review, basic implementation, output)
Module 4	Modular Modeling (DMAS framework, solver parameters)
Module 5	Coding Tools (Git, Make/CMake, Sphinx/Doxygen)
Module 6	Debugging (reading documentation, pdb/GDB, unit testing)
Module 7	Test Instances (data wrangling, random generation)
Module 8	Running Experiments (metrics, timing, logging/parsing)
Module 9	Analyzing Experiments (pandas, performance plots)
Module 10	Using Decompositions (Dantzig-Wolfe, Benders, Lagrangian)
Module 11	Callbacks (user/lazy cuts, primal heuristics, termination)
Module 12	Beyond Gurobi/CPLEX (branch-cut-and-price, DIP/SCIP)
Module 13	Special: Constraint-based Scheduling (Gecode, CPO)
Module 14	Special: Machine Learning (scikit-learn, TensorFlow)