EXAM - grade production for Discrete Optimization and Decision Management

Roberto Zanotti and Romeo Rizzi, AY 2024-25

The course ends with a final oral exam. This is an individual exam. Your final mark is obtained by adding to the mark of this oral the marks earned with your homework and project. As such, you produce your final mark during three separate moments:

1. HOMEWORK: up to 5 points to be added to the final mark. Homework is not compulsory but we advise you to face these before the project. Not only to raise your final mark but also to train and attune to the course and its tools from the very start.
2. PROJECT: up to 8 points to be added to the final mark. These depend only on the correctness and quality of the solutions computed, not on what will be addressed at the oral exam. The project is compulsory.
3. INDIVIDUAL ORAL EXAM, where you can be asked about the solutions and code you have submitted for the homework or the project, or you may be asked to prove that same kind of practical competence, or you may be asked questions on the topics in the list concluding this document.

HOMEWORK

A set of exercise problems to train an active problem-solving competence either direct or mediated by models. Some of these exercises require you to write an ILP or LP model. Others require an algorithm that directly solves the problem. Either way, you are asked to write a concrete code in your preferred programming language/environment (with or without the call to a Mathematical Programming Library like Gurobi) that automatically settles the issues posed. Your code, running on your machine, solves the instances sent to it by our evaluation and feedback services running on our server. The points scored on the various exercises will determine the bonus mark for the homework (5 points maximum). You can monitor how it grows at any new submission through the scoreboard service. This bonus mark does not expire. You can work in groups; these groups do not need to be static but may change with the exercises. But be aware that at your oral exam, you might be asked to explain the code you have submitted or to exhibit the knowledge or competence it is based upon.

PROJECT

The project requires you to write one or more ILP formulations to settle the various subtasks of a given computationally hard problem. Another peculiarity distinguishing the project from the homework is that the benchmark of instances (of various sizes, the small ones will facilitate your testing) for you to solve is fixed and made available to you in advance. As such, you will submit one single .zip (or .tar) file with the solutions for the various instances comprising the benchmark (plus the codes that produced them). As with the homework, you can work in groups. Again, the free constitution of the groups is left to you. And the achieved grade does not expire.

INDIVIDUAL ORAL

The dates/times of the final oral exam will be agreed with us by each individual student, separately, according to his needs and our availabilities. The exam must be scheduled for when the student has decided he will be done with his homework and project. During the exam, our goal is to assess: that the student perfectly understands what he has submitted for the homework and the project that the student has adequately acquired the active skills exhibited in his solutions to the homework and the project why he has made certain choices in the project and what he could have done differently that he has adequate competence and knowledge about the topics in the following list

Skills to be exhibited in the general oral exam:
1. knowledge of the theorems discussed (at least their statements)
2. knowledge of some notable models
3. ability to reason with examples and counterexamples
4. being able to provide examples and classes of exact, approximate, and metaheuristic algorithms
5. mastery of the language of the LP and of the ILP
6. ability to model a problem on an abstract level
7. knowledge of the topics in the following list

LIST OF TOPICS

Basic notions of Algorithms and Complexity

realistic operator/executor, algorithm=recipe/instructions for an operator, correctness and termination requirements mathematical programming: instances and problems=models computation time, worst-case criterion, asymptotic upper bound complexity, polynomial versus exponential, P, NP, and co-NP. SAT, statement of Cook's theorem, reduction from SAT to PLI

Linear Programming (reference: Vanderbei chapters 2,3,4,5, but no need to read the proof concerning Bland's rule)

linear systems and Gauss algorithm expression in matrix form of linear systems and early notions of linear algebra (linear combinations, linear independence, invertible matrices, determinant, Kramer's formula) total unimodular matrices and integrality of polyhedra (see: `https://theory.stanford.edu/~jvondrak/MATH233B-2017/lec3.pdf` or, for a more lengthy video: `https://www.youtube.com/watch?v=sKljREy6PFY`) two classes of totally unimodular matrices: incidence matrices of bipartite graphs (for proof of this and Koenig's theorem see the compact writing above or this video: `https://www.youtube.com/watch?v=uixM9GLvN6k`) incidence matrices of directed graphs (see: `https://www.math.unipd.it/~luigi/courses/metmodoc1617/m07.assTum.en.pdf`)

the language of linear programming: lines, planes, hyperplanes, gradient, convexity, polyhedral cones, polyhedra, polytopes, feasible region, void, bounded, unbounded, double representation of vertices and faces double representation of polyhedron = polytope + cone (`https://scaron.info/blog/polyhedra-and-polytopes.html`) graphic solution of a linear programming problem in two variables (`https://www.youtube.com/watch?v=gbL3vYq3cPk`) every LP prblem can be put in standard maximization form ($\max c'x : Ax \leq b, x \geq 0$) the canonical maximization form $\max c'xAx = b, x \geq 0$, basic solution, distinction between the vertex as a point in R^n and the basic solution which identifies it (but is not always unique), simplex algorithm, two-phases simplex algorithm (`https://www.youtube.com/watch?v=_wnqe5_CLU0`)

duality theory: dual problem, statements of the theorems of weak duality, strong duality, and complementary slacks.

Know how to model a problem with PL or ILP

With the PL you should know how to possibly address:

maximum flow and minimum cut, only for bipartite graphs: minimum node-cover, maximum matching and perfect matching of maximum value

formulations of min, max, min-max, max-min, absolute value

portfolio selection with MAD risk measure

ADVANCED (only for the laude): minimum cost spanning trees, maximum matching in non-bipartite graphs

With the PLI you should know how to possibly address:
maximum matching in generic graphs, modeling subtour elimination - connectivity constraints - MTZ, modeling a piecewise linear function, set partitioning, set packing, set covering, dependent decisions, Disjunctive constraints, soft constraints, ADVANCED (only for the laude): TSP

ADVANCED (only for the laude): cutting planes - generic Gomory cuts (`https://www.youtube.com/watch?v=1i0rKtH_YPs`)

Exact Algorithms
simple enumeration algorithms

implicit enumeration algorithms

branch & bound

branch & cut, with generic cuts (Gomory cuts) or based on families of problem-specific cuts (subcircuit elimination inequalities)

Approximation Algorithms
for the minimum node cover problem (any 2-approximation algorithm)

ADVANCED: for the TSP (a 2-approximation algorithm will be enough)

Meta-Heuristics
generative heuristics and refinement heuristics

greedy (examples: change and coins)

Local search (examples: 2-opt for the TSP): Definition and application examples to well-known problems (TSP, knapsack, etc.)

Simulated annealing, tabu search, genetic algorithms

Math-heuristics: different examples like Proximity Search, Local Branching, RINS

Kernel search: Basic principles and fundamental elements