Algorithmic Methods for Mathematical Models

– COURSE PROJECT –

Secret agent James Bond is in a dangerous mission fleeing from the enemy. In order to get back to his base, he has to enter some codes in the access control of the door. But he has to do it swiftly, because the enemy is treading on his heels.

More precisely, let us assume that the codes are sequences of digits 0 and 1 of length m. To unlock the door, there is a prefixed set S of n different codes that have to be entered. Each of these codes has to be entered exactly once, in any order, with the following exception: the first and the last codes that are entered have to be the sequence consisting of m zeros (which is always included in S).

Unfortunately for agent Bond, the access control is a bit outdated and has a rather awkward interface, which consists of 3 buttons (RIGHT, FLIP, ENTER) and a display. At any time the display shows a sequence of m digits 0 and 1 (initially, the sequence of m zeros). There is a cursor which at the beginning is at the leftmost position, and which can be moved from one position to the next one on the right by pressing the RIGHT button. The digit the cursor is on can be flipped with the FLIP button. Finally, when the sequence is ready and the cursor is at the rightmost position, the ENTER button must be pressed to register it. Then the cursor moves back automatically to the leftmost position, but the last sequence that was entered is kept on the display. The same procedure is applied for the next code, until all codes have been entered.

Agent Bond wants to enter the codes as quickly as possible, to avoid getting trapped by the enemy. The goal is to minimize the number of times that the FLIP button has to be pressed (the RIGHT and ENTER buttons are not included in this count because the number of times they have to be pressed is always the same: note that, in order to enter each code, the RIGHT button will have to be pressed $m - 1$ times, and the ENTER button once).

1. **Work to be done:**
   (a) State the problem formally. Specify the inputs and the outputs, as well as any auxiliary sets of indices that you may need, and the objective function.

   (b) Build a **mixed integer linear** programming model for the problem and implement it in OPL.

   (c) Because of the complexity of the problem, heuristic algorithms can also be applied. Here we will consider the following:

      i. a greedy constructive algorithm,

      ii. a greedy constructive + a local search procedure,

      iii. GRASP as a meta-heuristic algorithm. You can reuse the local search procedure that you developed in the previous step.

      Design the three algorithms and implement them in the programming language you prefer.

   (d) Tuning of parameters and instance generation:

      Given an instance consisting of a set S of codes, we define its *size* as $n = |S|$.

      i. Implement an instance generator that generates random instances for a given value of n.

      ii. Tune the α parameter of the GRASP constructive phase with a set of randomly generated instances of large enough size.

iii. Generate problem instances with increasingly larger size. Solving each instance with CPLEX should span from 1 to 30 min.

(e) Compare the performance of solving the model with CPLEX with applying the heuristic algorithms, both in terms of computation time and of quality of the solutions as a function of the size of the instances.

(f) Prepare a report and a presentation of your work on the project.

2. **Report:**

Prepare a report (8-10 pages) in PDF format including:

- The formal problem statement.
- The mixed integer linear programming model, with a definition and a short description of the variables, the objective function and the constraints. Do not include OPL code in the document, but rather their mathematical formulation.
- For the meta-heuristics, the pseudo-code of your constructive, local search, and GRASP algorithms, including equations for describing the greedy cost function(s) and the RCL.
- Tables or graphs with the tuning of parameters and instance generation.
- Tables or graphs with the comparative results.

Together with the report, you should also give all sources (OPL code, programs of the meta-heuristics, instance generator, etc.) and instructions on how to use them, so that results can be easily reproduced.

3. **Presentation:**

You are expected to make a presentation of your work (7-10 minutes long) at the end of the course.

The slots of Thursday 26/05/22 and Monday 30/05/22 will be devoted to these presentations. The schedule will be announced in its due time.

The slides of the presentation in PDF format should be delivered together with the report by **Tuesday 24/05/22**.

The idea is that the presentation can contain figures, plots, equations, algorithms, etc. with a very short text that helps to understand them. It is expected that you give a full explanation of those contents during your presentation. On the other hand, the report should contain that explanation in a well-organized manner as a text.