Algorithmic Methods for Mathematical Models

– COURSE PROJECT –

We are in charge of organizing a chess tournament for a number n of contestants. From now on, n will be assumed to be odd. The tournament will be arranged as a *single round-robin competition*: that is, each contestant will play exactly once against each of the other contestants. When the game between two contestants takes place, we have to decide who plays white (and therefore, moves first) and who plays black. Since it is widely acknowledged in the chess community that playing white is better, along the whole tournament a contestant should play black as many times as white.

We also have to determine when the games take place. To that end we have n time slots available. It is required that the number of games that are played simultaneously at each slot is always $\frac{n-1}{2}$, so that there is exactly one contestant who does not play. In order to decide when contestants are free and make everybody happy, a system of points has been established, which works as follows: before the tournament, the i-th contestant ($1 \leq i \leq n$) is given 100 points. Then they choose how to distribute these points according to their preferences among the n slots: p_{i1} points go to slot 1, p_{i2} points go to slot 2, ..., and p_{in} points go to slot n. For simplicity, let us assume that, in the input data, the p_{ij} are integers betwen 0 and 100 such that $\sum_{j=1}^{n} p_{ij} = 100$. These points allow defining the *score* of a schedule: if the 1-st contestant is free at slot s_1, the 2-nd contestant is free at slot s_2, etc., the score of the schedule is $p_{1s_1} + p_{2s_2} + \ldots$. Then different schedules can be compared according to their score. The goal is the find the schedule that maximizes the score.

For example, let us consider an instance of input where $n = 5$ and the matrix of points p_{ij} is

$$\begin{pmatrix} 10 & 20 & 30 & 40 & 0 \\ 0 & 0 & 50 & 50 & 0 \\ 100 & 0 & 0 & 0 & 0 \\ 60 & 30 & 0 & 0 & 10 \\ 20 & 25 & 15 & 15 & 25 \end{pmatrix}$$

A possible schedule for the tournament is as follows:

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
1 white vs. 4 black	2 white vs. 1 black	1 white vs. 5 black	3 white vs. 2 black	3 white vs. 1 black
2 white vs. 5 black	5 white vs. 3 black	4 white vs. 3 black	5 white vs. 4 black	4 white vs. 2 black

Note that contestant 1 is free at slot 4, contestant 2 is free at slot 3, contestant 3 is free at slot 1, contestant 4 is free at slot 2, and contestant 5 is free at slot 5. This means that the score of this schedule is $40 + 50 + 100 + 30 + 25 = 245$.

Another valid schedule could be obtained by swapping slots 1 and 2:

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
2 white vs. 1 black	1 white vs. 4 black	1 white vs. 5 black	3 white vs. 2 black	3 white vs. 1 black
5 white vs. 3 black	2 white vs. 5 black	4 white vs. 3 black	5 white vs. 4 black	4 white vs. 2 black

In this case contestant 3 is free at slot 2 and contestant 4 is free at slot 1 (the rest of the contestants do not change their free slot), so the score is $40 + 50 + 0 + 60 + 25 = 175$. This means that the first schedule is better than the second one.

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 an 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 of input to the problem, the value of n is the *size* of the instance.

   i. Implement an instance generator that produces random instances for a given size.

   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 take from 1 to 30 min.

(e) Compare the performance of CPLEX with 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 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.
   - Tables or graphs with the comparative results.

   Together with the report, you should give all sources (OPL code, programs of the meta-heuristics, instance generator) and instructions on how to use them, so that results can be 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 Tuesday 13/12/22, Thursday 15/12/22, Tuesday 20/12/22, and Thursday 22/12/22 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 **Sunday 11/12/22**.

   The idea is that the slides 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.