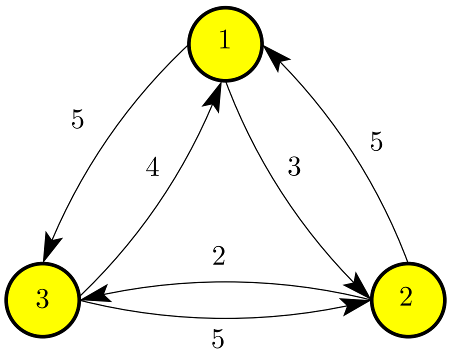
Algorithmic Methods for Mathematical Models
– COURSE PROJECT –

A group of N people who are interested in mining cryptocurrencies have decided to join forces and form a cooperative. Each of them has contributed with the same initiation fee, and the collected money has been employed to purchase a computer with the latest technology in GPUs. Since the computer can only be used by a single user at a time, the members of the cooperative agree to share it according to the following policy. Each month each member declares the days (the same number for all) when they may want to have access. Conflicts are resolved by establishing a *priority* between members: if member i has priority over member j, whenever i needs the computer, if j is using it at that moment, then j must immediately hand it over to i. To decide how priorities are defined, for each pair of members i and j who request to use the machine on some common day (and therefore could be potentially in conflict), member i bids $m_{ij} > 0$ euros for having priority over j, and reciprocally member j bids $m_{ji} > 0$ euros for having priority over i. All collected money then goes to the coffers of the cooperative. By convention, if there is no possible clash between i and j (in particular, if $i = j$), then $m_{ij} = m_{ji} = 0$.

Given N and the $N \times N$ matrix (m_{ij}), your goal is to decide how to define the priorities to resolve the conflicts between the members of the cooperative, in such a way that the income for the cooperative is maximized, and **no loops of priorities are formed**.

For example, let us assume that $N = 3$, and let us identify the members of the cooperative with the numbers 1, 2 and 3. The graph on the left shows the bids of each of the members, and the matrix on the right is the corresponding numerical representation.



$$\begin{pmatrix} 0 & 3 & 5 \\ 5 & 0 & 2 \\ 4 & 5 & 0 \end{pmatrix}$$

If we gave 1 priority over 3, 3 priority over 2 and 2 priority over 1, the cooperative would collect $5 + 5 + 5 = 15$ €. However, this is not a valid solution: if some day the three members happened to request the computer, a deadlock would be reached.

Instead, the optimal solution is to give 3 priority over 1, 2 priority over 1, and 3 priority over 2. In this case there is no deadlock, and the cooperative collects $4 + 5 + 5 = 14$ €.

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, and with the comparative results.

   Together with the report, you should give all sources (OPL code, programs of the meta-heuristics, instances, 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 at the end of the course (at most 10 minutes long; overtime presentations will be terminated). All group members must participate in the presentation and know all the work in the project. The slots of 27/05/25 and 30/05/25 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 with the report by **25/05/25**.

   The slides can contain plots, equations, algorithms, ... with a very short text that helps to understand them. You are expected to give a full explanation of those contents in the presentation. On the other hand, the report should contain that explanation in a well-organized way as a text.