Algorithmic Methods for Mathematical Models

– COURSE PROJECT –

The guild of bakers has designed a mobile app for centralizing the orders of customers. Throughout the day, customers post their orders with the app for the following day. Once the deadline for orders is over (say, at midnight), each baker chooses, from the list of all orders that are still unattended, which ones they will take. Next the app notifies the customer which baker takes care of their order, so that they know where to pick it up once ready. Then each baker bakes the bread of the orders they had chosen, and finally customers pick up their order at the corresponding bakery. For the management of orders, the app discretizes the time in a day in t time slots (e.g., in 1-hour slots: from 8AM to 9AM, from 9AM to 10AM, etc.), which are identified with the numbers from 1 to t.

We are in charge of a bakery and want to optimize the order selection process. There are n orders we can take, identified with the numbers from 1 to n. Each order has the following associated data:

- $profit_i$: profit in € that we make if order i is taken.
- $length_i$: number of time slots required for baking order i.
- $min\_deliver_i$: earliest time slot the customer could pick up order i.
- $max\_deliver_i$: latest time slot the customer could pick up order i.
- $surface_i$: surface in the oven that order i requires, measured in m^2.

Due to storage limitations and product quality considerations, orders have to be picked up at the same time slot as the baking process finishes. Moreover, the oven in the bakery has a limited surface capacity of *surface_capacity* m^2. At any time, the total surface of the bread that is being baked cannot exceed this capacity. The goal is to maximize the profit made with the choice of the orders we take.

For example, let us consider the following data: $n = 3$, $t = 4$, $profit = (2, 4, 3)$, $length = (2, 4, 1)$, $min\_deliver = (2, 1, 4)$, $max\_deliver = (3, 4, 4)$, $surface = (3, 2, 4)$, $surface\_capacity = 6$. In this case it is possible to take all orders, and the optimal profit is then $2 + 4 + 3 = 9$. A way of scheduling the orders is as follows (where an • at row i column j means order i is being processed at time slot j):

Order	**Time slot**			
	1	2	3	4
1	•	•		
2	•	•	•	•
3				•
total m^2	5	5	2	6

All rows in the table but the last one correspond to orders. The last row shows the total surface of bread that is being baked at any time. Notice that this value never exceeds 6, the capacity of the oven. Note also that the baking of orders 1, 2 and 3 takes 2, 4 and 1 time slots respectively, as required. Moreover, order 1 is to be picked up at time slot 2 (which is in the window $[2, 3]$), order 2 is to be picked up at time slot 4 (which is in the window $[1, 4]$), and order 3 is to be picked up at time slot 4 (which is in the window $[4, 4]$).

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

      ii. 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) 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 members of the group must participate in the presentation and know the whole work carried out in the project. The slots of Friday 15/12/23, Tuesday 19/12/23 and Friday 22/12/23 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 **13/12/23**.

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.