This report briefly describes the steps taken to complete the Discrete Optimization project.

It was chosen problem 1 - "Series assignment", all results included here come from list of students provided in file *cursusA.csv*. For details regarding the problem instructions, please refer to *statement.pdf* file.

1 Series assignment

1.1 Integer programming formulation

Sets

- S : Set of students
- $O \subseteq E$: Set of oral exams
- S_e : Set of exams $e \in O$ that student needs to take

Parameters

- $max(e)$: Limit number of students per series of exam e
- K_e : Number of series needed per exam $e \in O$

$$n\_students_e = \sum_{s \in S} a_{s,e} \quad \forall e \in O$$

$$K_e = \left\lceil \frac{n\_students_e}{max(e)} \right\rceil$$

Decision variables

- $z_{s,e,i}$: Binary variable, 1 if student s is assigned to series i of exam e, 0 otherwise
- $max_{e,i}$: Integer variable representing the maximum number of students in series i of exam e

Objective function

- Minimize the sum of squares of the maximum number of students in each series for each exam:

$$\text{Minimize} \sum_{e \in O} \sum_{i \in K_e} (max_{e,i})^2$$

To ensure homogeneity in the series assignment, we minimize the difference in the number of students across different series of the same exam.

Constraints

- Each student must be assigned to exactly one series for each oral exam they are taking:

$$\sum_{s \in S} z_{s,e,i} = 1, \quad \forall e \in S_e, \forall i \in K_e$$

- Ensure students are not assigned to series for exams they are not taking:

$$z_{s,e,i} = 0, \quad \forall s \in S, \forall e \in O \setminus S_e, \forall i \in K_e$$

- The number of students assigned to any series must not exceed the maximum number of students per exam's series:

$$\sum_{s \in S} z_{s,e,i} \leq max(e) \quad \forall e \in O, \forall i \in K_e$$

- Similar to previous constraint but, it must not exceed the variable $max_{e,i}$:

$$\sum_{s \in S} z_{s,e,i} \leq max_{e,i} \quad \forall e \in O, \forall i \in K_e$$

- Ensure series are filled sequentially:

$$max_{e,i} \geq max_{e,i+1} \quad \forall e \in O, \forall i \in [1 : |K_e| - 1]$$

Formulation

$$\begin{aligned}
&\text{Minimize } \sum_{e \in O} \sum_{i \in K_e} (max_{e,i})^2 \\
&\text{subject to} \\
&\qquad \sum_{s \in S} z_{s,e,i} = 1, \ \forall e \in S_e, \forall i \in K_e \\
&\qquad z_{s,e,i} = 0, \quad \forall s \in S, \forall e \in O \setminus S_e, \forall i \in K_e \\
&\qquad \sum_{s \in S} z_{s,e,i} \leq max(e) \quad \forall e \in O, \forall i \in K_e \\
&\qquad \sum_{s \in S} z_{s,e,i} \leq max_{e,i} \quad \forall e \in O, \forall i \in K_e \\
&\qquad max_{e,i} \geq max_{e,i+1} \quad \forall e \in O, \forall i \in [1 : |K_e| - 1] \\
&\qquad z_{s,e,i} \in \{0, 1\} \quad \forall s \in S, \forall e \in O, \forall i \in K_e \\
&\qquad max_{e,i} \geq 0 \quad \forall e \in O, \forall i \in K_e
\end{aligned}$$

The results regarding the value of the objective function and the time to reach a solution are presented in Table 1.

1.2 Heuristic - Simulated annealing

It is a metaheuristic based on local search algorithm that tries to escape from local minima.

The pseudocode[1] of the algorithm is presented below in Figure 1.

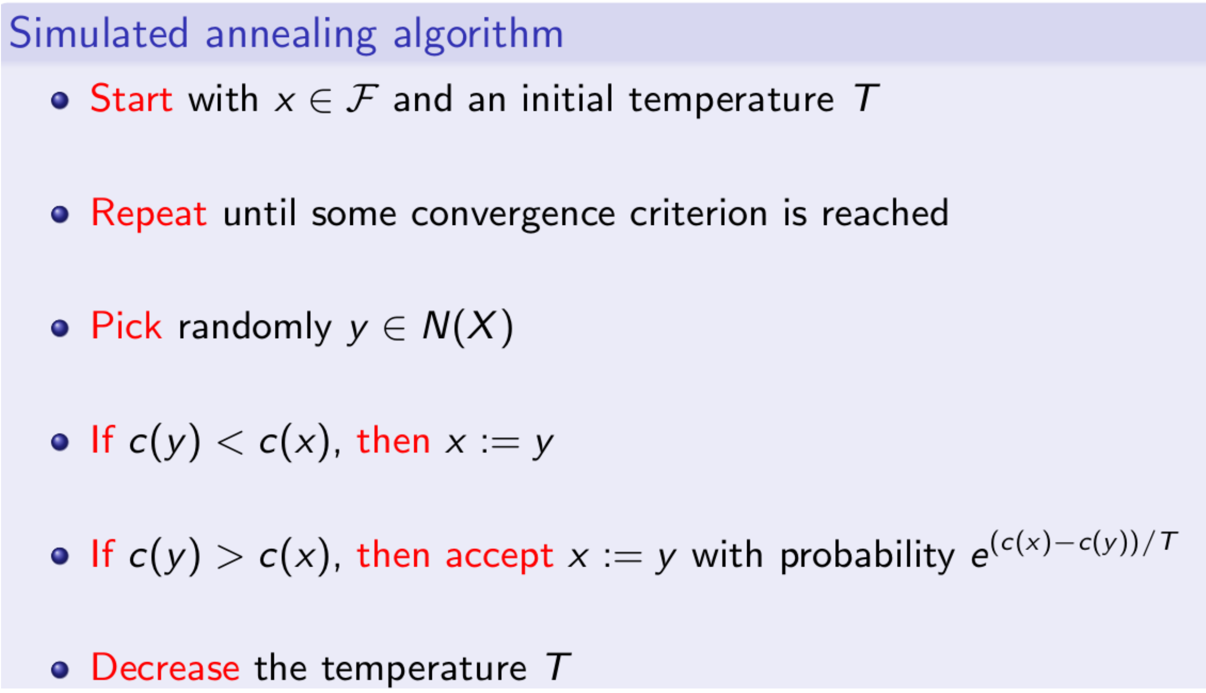

Figure 1: Simulated annealing algorithm.

As requested in the statement, the heuristic solution has to provide the best possible solution in less than 2 minutes.

To enhance the best possible outcome from the heuristic, function `generate_initial_assignment()` only provides available random series k_i to students s. If an oral's exam series exceeds its maximum student per day it is no longer available.

Our heuristic solution makes use of three hyper-parameters: (1) `max_iterations=8700`, (2) `initial_temperature=100` and (3) `cooling_rate=0.9992`.

As expected the heuristic solution requires more time to provide a possible solution, we make use of `max_iterations` parameter to set the limit to 2 minutes in our machine, see Table 1.

Table 1: Results from the two Exam sessions.

	Ip model		**Heuristic**	
Session	**Objective funct**	**Time**	**Objective funct**	**Time**
January	13501	∼14s	∼13503	∼52s
June	58859	∼16s	∼58863	∼ 1m19s
Total	72360	∼26s	∼72365	∼ 1m55s

Note: To consider that reading and pre-processing the data takes around 9 seconds.

[1] All material was extracted from the course lectures.