IE 206 - Term Project II

Daily Surgery Room Scheduling in METU Hospital

The head doctor of a local hospital wants to schedule the medical operations for daily surgery rooms. The doctor makes the schedules for the following 5 consecutive days. Operations that are performed for each day will be determined and scheduled according to some constraints explained below:

- There is a certain number of surgery rooms in the hospital, and the operations should be placed only in one of these rooms. In each room, only one operation will be performed at a given time. The number of rooms is fixed and equal to 3.
- You may assume that the number of doctors is sufficient to serve the patients.
- The time horizon at which you will perform scheduling is between 9:00 and 17:00 for each day. You may take this interval as [0, 480]. The data provided for each day is given as being in this interval.
- Each operation has a fixed operation time, and it needs to be completed within the given time window.
- The operations are aimed to take place in the given time interval. For instance, if the time window for an operation is [0,40], and its duration is 20 minutes, then 21st minute is considered as the latest start time. Otherwise, the end of the operation exceeds the time interval.
- There is a priority level for each patient who will take the operation, and it is represented by a 4-level integer value as 1, 2, 3, and 4. While Level-1 denotes the highest priority, Level- 4 denotes the least priority.

- Priority mainly refers to the waiting time for an operation. The patient with the higher priority should spend less waiting time in the system than the patients having lower priorities.

- Suppose you have two patients whose operations need to be completed in the time window [0,20], and each takes 10 minutes to operate. Then the one with higher priority should be completed before the other.

- Each operation also has a complexity level from 1 to 3. If the complexity of an operation is Level-1, then the room can be utilized directly after this operation. Otherwise, for operations with complexity levels 2 and 3, the operation room needs to undergo sanitization for 20 or 40 minutes, respectively, before it can be utilized again.

- The operations in the same room cannot overlap.

- As the duration of the operations is fixed, you can assume that there is no change in the operation duration (The operation duration is not probabilistic).

- When an operation cannot be scheduled on its original day, you need to make sure that the corresponding operation is scheduled on the very next day. This operation cannot be postponed two days in a row. Therefore, you need to apply the following steps:

  - If the postponed operation is Level-1, update its priority level to Level-0, representing the emergency level, and it is a higher priority than Level-1. This priority level is ONLY given to the cases which are postponed to the next day. The start time of the time window for Level-0 operation should be updated to 9:00, and the end time of the time window should be set as the summation of the start time and the duration of the operation.

  - If the postponed operation is not a Level-1 operation, update its priority level to one lower (i.e. from Level-3 to Level-2). The time window for this operation should remain unchanged.

  - The day of the patient will be updated.

- The scheduling algorithm should place all the operations in the list according to the rules above.

The framed boxes in Figure 1 indicate the time intervals during which the operations can be scheduled, and the actual schedules of the operations in that time intervals are denoted with yellow color for each room for a given day.

We provide the design of the classes that you will be using in this project in Figure 2. The diagrams will be helpful when you later write your code that involves interactions among the objects of the different classes. Please read carefully the details of the classes and objects below.

Mainly, the following terminology will be used in the project: *operation*, *schedule*, *patient*, and *interval*. These are the objects that we will use. An ***Interval*** has a left (start) and right (end) hand. An ***Operation*** has an ID, patient, duration, a time window (available interval) during which the surgery can take place, the scheduled interval during which the surgery is planned to be performed, and the day that the operation is performed. ***Patient*** has a name, surname, complexity level, and priority level, which is between 1 and 4, describing her/his situation and

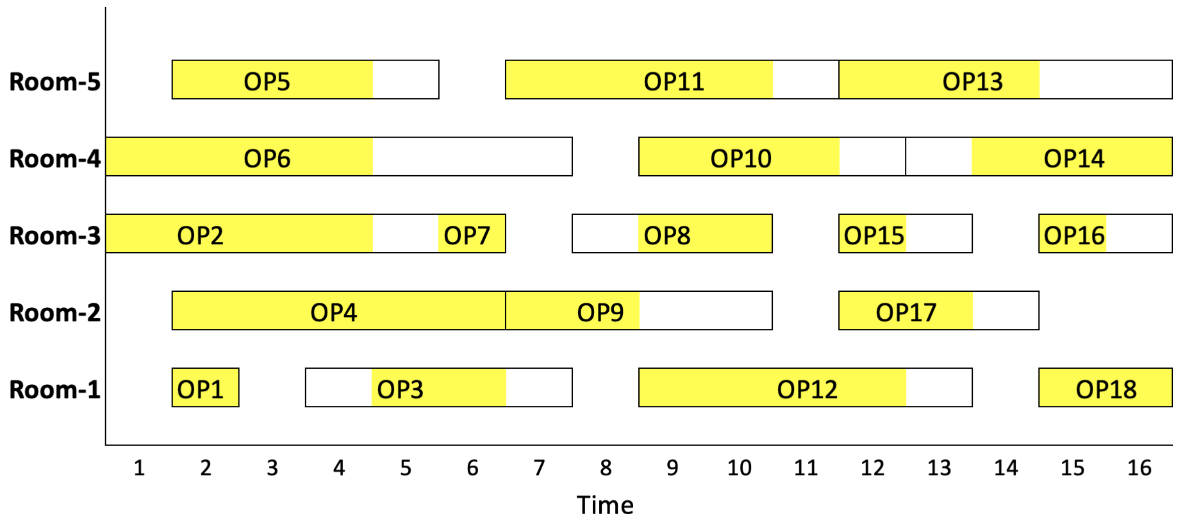

Figure 1: An example of scheduled operations with presenting the given time windows and operation durations

the day that the patient is initially listed to have the operation. A ***Schedule*** contains the daily planning horizon, total planning days that the schedule will be performed, number of available rooms in the hospital, and final schedule of the operations.

Our design involves four classes: ***Interval*** , ***Operation***, ***Schedule*** and ***Patient***. We then write a main script to read event data from an Excel file and then instantiate objects of the different classes in order to create a feasible schedule.

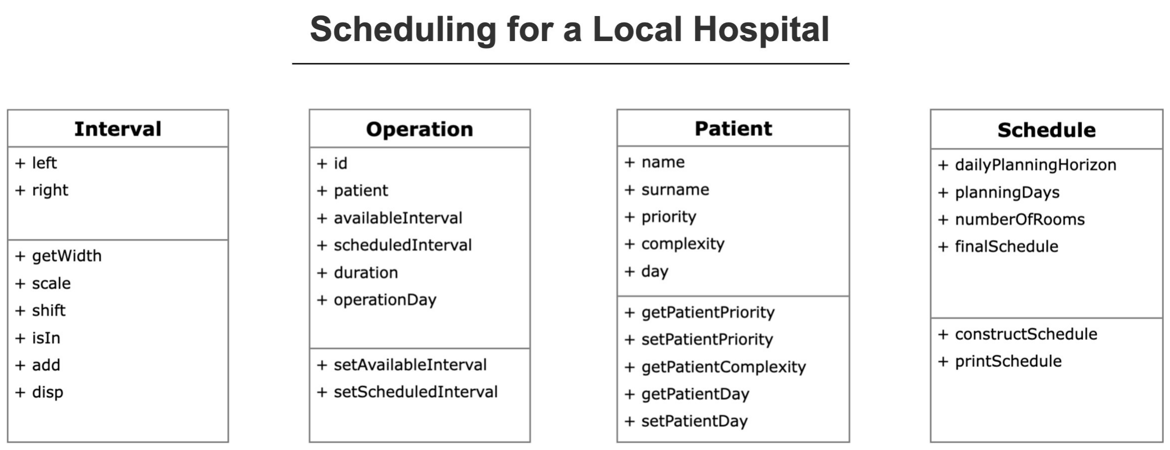

Figure 2: Diagram for the Classes

So which class should you work on first? It would be better to start with the most independent class, the one that does not depend on other classes. So start with class ***Interval*** . As you complete the classes, do not change the names of the properties, methods, and parameters. You may add extra methods or properties to the classes if needed (DO NOT define additional classes), but make sure that you explain your additional declarations in detail, both in the report and in the script as a comment.

1) **Interval** Class
   Class Interval has left and right properties. The description of these properties is as follows

- left: start point of the interval
- right: endpoint of the interval

The description of the methods is as follows:

- getWidth: Gives the length of the interval.
- scale: Scales the interval by a factor.
- shift: Shifts the interval by a constant.
- isIn: Checks whether a given interval is included in another interval. It returns true (1) if it is included, false otherwise.
- add: Adds predetermined values to the start and endpoints to the interval (the values that will be added to start and endpoints may differ).
- disp: Displays the interval in this format: (left,right).

Download the Interval.m from ODTUClass. The methods that you need to implement are marked with '%**MISSION**'.

2) **Operation** Class
An operation has an ID, a Patient, an available Interval (time window) in which it can be scheduled, a duration, and a time at which it's finally scheduled. Description of the properties is as follows:

- id: Id number of the operation
- patient: Patient (object of type Patient)
- availableInterval: Available interval for scheduling an operation (object of type Interval)
- scheduledInterval: Scheduled start time (object of type Interval)
- duration: Length of the operation in minutes
- operationDay: the day that the operation will be held (Initially no value is assigned to this property. It will take a value when the operation is scheduled.)

Description of the methods are as follows:

- setScheduledInterval: A setter method to set scheduledInterval property of the object Operation when it is scheduled.
- setAvailableInterval: A setter method to set availableInterval property of the object Operation when the operation is to be postponed to the next day.

3) **Patient** Class
A patient has a name, surname, priority level, and a day. Description of the properties is as follows:

- name: Name of the patient
- surname: Surname of the patient
- priority: Priority of the patient's situation
- complexity: Complexity of the patient's operation
- day: the day that the patient initially listed to have an operation

The description of the method is as follows:

- getPatientPriority: A getter method to obtain the priority level of the object Patient.
- setPatientPriority: A setter method to set the priority level of the object Patient.
- getPatientComplexity: A getter method to obtain the complexity level of the object Patient.
- getPatientDay: A getter method to obtain the day of the object Patient.
- setPatientDay: A setter method to set the day of the object Patient.

4) **Schedule** Class
A schedule has a daily planning horizon, total planning days, number of rooms, and a final schedule. Description of the properties is as follows:

- dailyPlanningHorizon: Planning horizon in a day that the event can be scheduled (object of type Interval)
- planningDays: total number of days that the schedule will be planned
- numberOfRooms: Number of rooms on hand
- finalSchedule: The cell array of scheduled operations (Initially no value is assigned to this property. This property will take a value after the schedule has generated)

The description of the methods is as follows:

- constructSchedule: Adds operations to the property finalSchedule with the help of a heuristic that you will construct. It is not the constructor method of the class. Therefore, in the beginning, the property finalSchedule is an empty cell array, and it will be updated in this method.
- printSchedule: Prints the resulting schedule in a proper format.

Your Task

Your first task is to complete the classes. Then, you need to design heuristics for scheduling the operations for the given objectives. In other words, you need to construct three heuristics to solve the scheduling problem in order to serve each objective function, see below.

- schedule_objective_1: The first objective is to maximize the number of operations which is completed within its initial available time interval.

- schedule_objective_2: The second objective is to maximize the utilization of the rooms.
- schedule_objective_3: The third objective is to minimize the late completion time.

For *schedule objective 3*, none of the operations can start before the earliest time. However, an operation can start within the available time interval and can violate the latest time window. In case of a time window violation, there will be a penalty cost corresponding to the duration that exceeds the completion time. If a task cannot be completed and is delayed to the next day, then the penalty will be the whole duration of that task.

How to run?

You will need a main script to run your program. In this script, you will be reading the data from an Excel file, *InputData.xslx*, and instantiate the objects that you need with the help of classes that you have defined. Then with the help of these objects, you will form the final schedule using the method *constructSchedule* in the Schedule class. Then, you need to print your schedule with the help of *printSchedule* method in the Schedule class. In other words, you will use the main script for only reading data, instantiating the objects, calling the required methods of classes, and printing the final schedule.

Output Format

Suppose you have two rooms in the hospital. *printSchedule* method, included in class Schedule, should give output as follows:

Table 1: Sample Output

Room No	**Available Interval**	**Duration (min)**	**Sched. Interval**	**Patient Name**	**Patient Surname**	**Patient Priority**	**Operat-ion Day**
1	(0,40)	20	(0,20)	Melissa	Karagur	1	1
1	(0,40)	20	(20,40)	Alper	Sener	2	1
1	(40,120)	60	(40,100)	Dilay	Ozkan	3	1
1	(240,480)	120	(240,360)	Melis	Boran	1	2
2	(60,160)	80	(80,160)	Can	Er	1	2
2	(20,80)	40	(20,60)	Beril	Akkaya	2	2

Please note that the schedule given above is only an example. According to your objective, it may change. You will give the output,

- in an Excel file,
- as a Gantt Chart (for each day).

Project Report

The report should be written in Times New Roman with a font size of 12. It needs to include a cover page, table of contents, and page numbers. The number of pages, excluding the cover page and table of contents, should not exceed 10 pages. Please pay attention to report requirements since you will be getting points from the format also.

Your report should have a brief introduction and conclusion. In between these sections, you will present *flow charts* for each algorithm that you design for each objective function. Make sure that format of your *flow charts* are as in the lecture notes, and they express the steps of your algorithm in a clear way. You are required to justify how each of the algorithms serves the given objective functions. You need to include these in appropriate sections for the sake of the understanding of your report. Please DO NOT copy and paste the script files to your report. You will lose points if you do so.

After explaining the two algorithms developed, you are expected to analyze the following issues as well:

- For the *schedule_objective_1* report the following:
  - the objective function value,
  - utilization of the rooms,
  - the number of operations shifted in each priority level.
- For the *schedule_objective_2* report the following:
  - the objective function value,
  - the number of operations that are postponed to the next day,
  - the number of operations shifted in each priority level.
- For the *schedule_objective_3* report the following:
  - the objective function value,
  - utilization of the rooms,
  - the number of operations that are postponed to the next day,
  - the number of operations shifted in each priority level.

  Moreover, answer the following questions. Support your answers using your computational environment.
- What if the number of rooms is increased by 1 for both objectives? What changes in the schedules?
  - Considering the new case above, is the usage of rooms fair, or is there a room used more than the others?