Algorithm Fall 2022 - Final project
FUV course scheduling

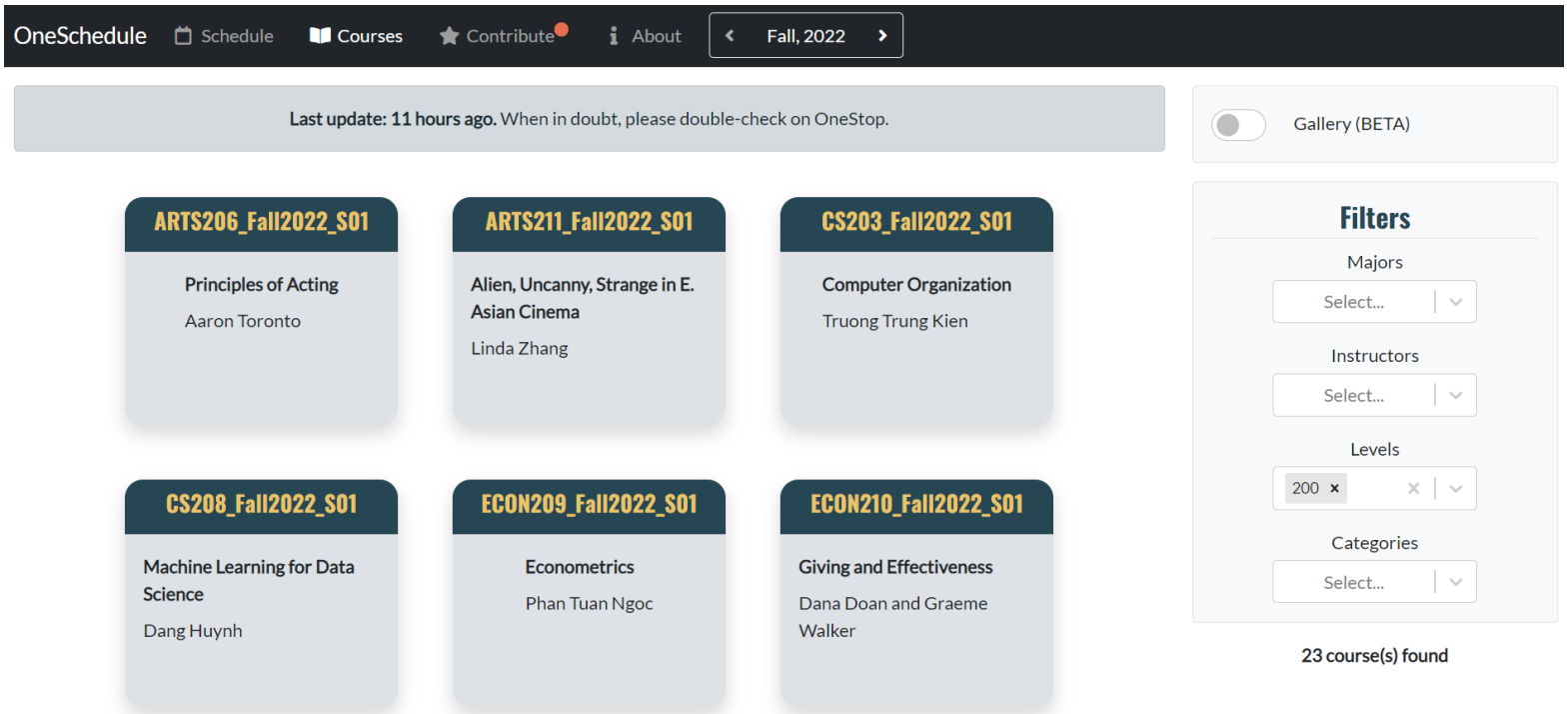
Nguyễn Hoàng Ngọc Hà, Hà Huy
Trần Kim Ngân, Nguyễn Công Thành

Instructor : Huynh Viet Linh

Problem

Problem

Develop an algorithm to schedule courses at FUV for each semester.



Input - Output

Input:

- A graph where
  - Each node is a course, and the corresponding maximum number of students.
  - Each edge between two nodes means these two courses can not be scheduled at the same time.
- A list of classrooms and the capacity (i.e. maximum number of students) of each classroom.
- A list of possible class time slots (e.g., 8am to 9:30am Mon/Wed,...).
- A list of time slots (i.e. 10am - 11am Monday) that all faculty members should be available (e.g., for weekly meeting).

Input - Output

Input:

- A graph where
  - Each node is a course, and the corresponding maximum number of students.
  - Each edge between two nodes means these two courses can not be scheduled at the same time.
- A list of classrooms and the capacity (i.e. maximum number of students) of each classroom.
- A list of possible class time slots (e.g., 8am to 9:30am Mon/Wed,...).
- A list of time slots (i.e. 10am - 11am Monday) that all faculty members should be available (e.g., for weekly meeting).

Output: A schedule if it is possible, otherwise return "No!" and suggest some constraints that should be reduced.

Time slot: A period of time that corresponds to a particular classroom and can be selected as a course time slot (eg 8:30 a.m. to 9:30 p.m. Monday, classroom 101).

Time slot: A period of time that corresponds to a particular classroom and can be selected as a course time slot (eg 8:30 a.m. to 9:30 p.m. Monday, classroom 101).

Assuming there are m time slots

$$T = [\text{timeslot}_1, \text{timeslot}_2, \ldots, \text{timeslot}_m],$$

and n courses

$$[\text{course}_1, \text{course}_2, \ldots, \text{course}_n].$$

Time slot: A period of time that corresponds to a particular classroom and can be selected as a course time slot (eg 8:30 a.m. to 9:30 p.m. Monday, classroom 101).

Assuming there are m time slots

$$T = [\text{timeslot}_1, \text{timeslot}_2, \ldots, \text{timeslot}_m],$$

and n courses

$$[\text{course}_1, \text{course}_2, \ldots, \text{course}_n].$$

To schedule, each course needs 2 time slots.

1. $m \geqslant 2n$

$$A = \underbrace{[\text{course}_1, \text{course}_1, \text{course}_2, \text{course}_2, \ldots, \text{course}_n, \text{course}_n, \ldots, \text{NaN}, \text{NaN}]}_{m \text{ elements}}$$

$$T = \underbrace{[\text{timeslot}_1, \text{timeslot}_2, \ldots, \text{timeslot}_m]}_{m \text{ elements}},$$

1. $m \geqslant 2n$

$$A = \underbrace{[\text{course}_1, \text{course}_1, \text{course}_2, \text{course}_2, \ldots, \text{course}_n, \text{course}_n, \ldots, \text{NaN}, \text{NaN}]}_{m \text{ elements}}$$

$$T = \underbrace{[\text{timeslot}_1, \text{timeslot}_2, \ldots, \text{timeslot}_m]}_{m \text{ elements}},$$

$\longrightarrow$ Considering all permutations of A, for each permutation, we will assign the ith element of A with the ith element of T to obtain a schedule for n course.

1. $m \geqslant 2n$

$$A = \underbrace{[\text{course}_1, \text{course}_1, \text{course}_2, \text{course}_2, \ldots, \text{course}_n, \text{course}_n, \ldots, \text{NaN}, \text{NaN}]}_{m \text{ elements}}$$

$$T = \underbrace{[\text{timeslot}_1, \text{timeslot}_2, \ldots, \text{timeslot}_m]}_{m \text{ elements}},$$

$\longrightarrow$ Considering all permutations of A, for each permutation, we will assign the ith element of A with the ith element of T to obtain a schedule for n course.

2. $m < 2n$: suggest adding time slots and rerunning the algorithm.

Need to check 3 more conditions:

i) For each course, the course capacity cannot exceed the capacity of the respective classroom.

ii) Each course has a different class hour than the courses with which it conflicts.

iii) There are some time slots that all faculty members should be available.

Need to check 3 more conditions:

i) For each course, the course capacity cannot exceed the capacity of the respective classroom.

ii) Each course has a different class hour than the courses with which it conflicts.

iii) There are some time slots that all faculty members should be available.

However, time complexity: $\Omega(n!)$

Time complexity: $\Omega(n!)$

We only need 1 schedule while all permutations of A maybe contain so many schedules!

$\longrightarrow$ Find a way to construct a schedule that satisfies the three conditions i), ii), ii)) instead of considering all permutations.

Sort T in descending order of capacity.

$$T = [x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j, z_1, z_2, \ldots, z_k],$$

$x_1, x_2, \ldots, x_i$: time slots of capacity I, $y_1, y_2, \ldots, y_j$: time slots of capacity II, $z_1, z_2, \ldots, z_k$: time slots of capacity III.

Capacity I $>$ capacity II $>$ capacity III.

In turn, type I capacity courses are assigned (if possible) to type I time slots.

In turn, type I capacity courses are assigned (if possible) to type I time slots.

Do the same for courses with capacity type II, noting that if all type II time slots have been considered and are still insufficient, the remaining type I time slots may be considered.

In turn, type I capacity courses are assigned (if possible) to type I time slots.

Do the same for courses with capacity type II, noting that if all type II time slots have been considered and are still insufficient, the remaining type I time slots may be considered.

Similar to the type III capacity course.

$\longrightarrow$ We get a schedule satisfying i).

To satisfy ii), in each step relating a course to a time slot, it is sufficient to determine whether the conflicting courses have been scheduled at that time, if so, go to the next time slot. Repeat this process until a suitable time slot is identified.

Similar for iii).

Question: Is this the correct way?

Question: Is this the correct way? If the algorithm is unable to schedule the class, is there truly no suitable schedule, or is it just that our algorithm cannot generate it?

Question: Is this the correct way? If the algorithm is unable to schedule the class, is there truly no suitable schedule, or is it just that our algorithm cannot generate it?

Answer: In general, we cannot be certain that the algorithm's conclusion that our courses cannot be scheduled is accurate.

Why?

The same issue is likely to appear in all algorithms that chose a schedule for each classroom in turn.

The time slot of a course will affect the available time slots of the courses that clash with it; hence, each choice of time slot for a course will generate a different case for the remaining time slots, making it impossible to control the result.

In fact, this is an NP problem (source: Prof Linh Huynh).

In fact, this is an NP problem (source: Prof Linh Huynh).

$\longrightarrow$ We will try to find an algorithm that has a short running time and acceptable accuracy.

In fact, this is an NP problem (source: Prof Linh Huynh).

$\longrightarrow$ We will try to find an algorithm that has a short running time and acceptable accuracy.

Fortunately, the above idea gives us such an algorithm!

Decode

With csv file of n courses with course code, lecturer, capacity, and csv file of m classrooms with classroom name, and available time, we decode our input as follows:

- *Course_list*: List of all courses code with its capacity.
- *Conflicting_dict*: {faculties'sname: [timeslots]}.
- *Back_up_Classroom_schedule*: List of backup slots with classroom name, available time.
- *Classroom_schedule*: List of slots with classroom name, available time.

```
// Function to find a suitable time slot for a given course. Return
   its time slot (2 sessions, classroom/day/time) if possible,
   otherwise, return empty
Function arrangeSchedule(classroom_schedule, course, Conflicting_dict[faculty]):
    for classroom in Classroom_schedule do
        if the course's capacity ⩾ the class's capacity then
            continue
        else
            Available_slot = Time slots of classroom - Conflicting_dict [faculty]
            for time in office time do
                if there is only one day contains this time then
                    continue
                else
                    timeslot1 = the earliest day that contains this time
                    timeslot2 = the next day that contains this time so that two
                     days are at least 1 day apart
                    schedule = timeslot[day1,time], timeslot[day2,time]
                    return schedule
                end
            end
            for day in office time do
                if there are 2 consecutive timeslots on this day then
                    schedule = timeslot[day,time1], timeslot[day,time1]
                    return schedule
                end
            end
        end
    end
    return []
end
```

Algorithm 1: Course scheduling algorithm

Input : *Course_list, Conflicting_dict, Back_up_Classroom_schedule, Classroom_schedule*

Output: True and a schedule, or False and suggestion.

```
course_schedule = [] // List of schedule corresponding with Course_list
Arrange = True
for each course in Course_list do
    schedule = arrangeSchedule(Back_up_Classroom_schedule, course,
     Conflicting_dict[faculty])
    if schedule is empty list then
        Arrange = False
        schedule = arrangeSchedule(Back_up_Classroom_schedule, course,
         Conflicting_dict[faculty])
        if schedule is empty list then
            Print: "We do not have enough classroom"
            continue
        else
            Print: "We should add two time slots" and schedule
        end
        Update course_schedule, Conflicting_dict with schedule
    end
end
Format course_schedule
return Arrange, course_schedule
```

Algorithm 1: Course scheduling algorithm

Input : *Course_list, Conflicting_dict, Back_up_Classroom_schedule, Classroom_schedule*

Output: True and a schedule, or False and suggestion.

```
1  course_schedule = [] // List of schedule corresponding with Course_list
2  Arrange = True
3  for each course in Course_list do
4      schedule = arrangeSchedule(Back_up_Classroom_schedule, course,
         Conflicting_dict[faculty])
5      if schedule is empty list then
6          Arrange = False
7          schedule = arrangeSchedule(Back_up_Classroom_schedule, course,
             Conflicting_dict[faculty])
8          if schedule is empty list then
9              Print: "We do not have enough classroom"
10             continue
11         else
12             Print: "We should add two time slots" and schedule
13         end
14         Update course_schedule, Conflicting_dict with schedule
15     end
16 end
17 Format course_schedule
18 return Arrange, course_schedule
```

Time complexity: $O(mn)$

Classroom Schedule: Create a list of m classrooms with the capacity that follows the capacity ratio

$$10 : 30 : 45 : 50 = 1 : 2 : 2 : 6.$$

Classroom Schedule: Create a list of m classrooms with the capacity that follows the capacity ratio

$$10 : 30 : 45 : 50 = 1 : 2 : 2 : 6.$$

Faculty list: Create a list of faculties in which:

- Most of the faculties teach 2 courses.
- Some of the faculties teach 1 course.

Classroom Schedule: Create a list of m classrooms with the capacity that follows the capacity ratio

$$10 : 30 : 45 : 50 = 1 : 2 : 2 : 6.$$

Faculty list: Create a list of faculties in which:

- Most of the faculties teach 2 courses.
- Some of the faculties teach 1 course.

Course list: Create a list of n courses with

- Capacity: Randomly assigned.
- Faculty: Randomly assigned from the faculty list.

(n, m)	Running time	Result
$(10, 2)$	0.01s	True
$(85, 11)$	0.035s	False
$(95, 11)$	0.039s	False
$(5000, 565)$	48.4s	True

The second and third test cases use the data as the courses and classrooms for the Fall 2022 and Spring 2023 semesters at our university.

Suggestions and the corresponding schedules for these two cases are:

Fall 2022:

- Need to add slot 4:45 - 6:15 on Mon and Wed for classroom 302 for course VS204_Fall2022_S01_20.
- Need to add slot 4:45 - 6:15 on Mon and Wed for classroom 454 for course VS210_Fall2022_S01_40
- Need to add slot 4:45 - 6:15 on Tue and Fri for classroom 302 for course VS214_Fall2022_S01_25
- Need to add slot 6:30 - 8:00 on Mon and Wed for classroom 302 for course VS215_Fall2022_S01_30
- Need to add slot 6:30 - 8:00 on Tue and Fri for classroom 302 for course VS303_Fall2022_S01_30

As the number of courses is increase compare to Fall 2022, for Spring 2023, the algorithm suggests we add 15 more slots to get a suitable schedule.

The suggested schedules can be found here

- Aach course requires two distinct time slots.
- Aach classroom can only handle twenty time slots per week.

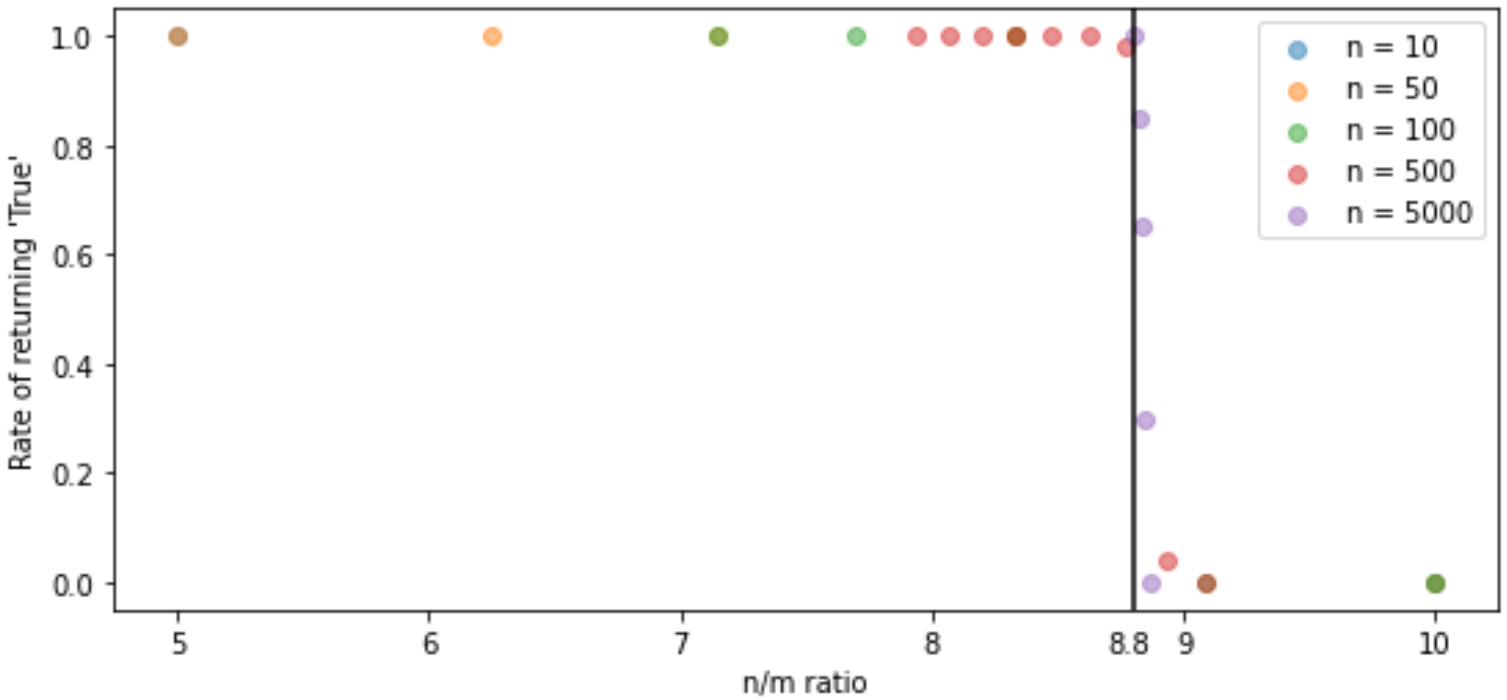
$\longrightarrow$ A classroom may accommodate a maximum of 10 courses.

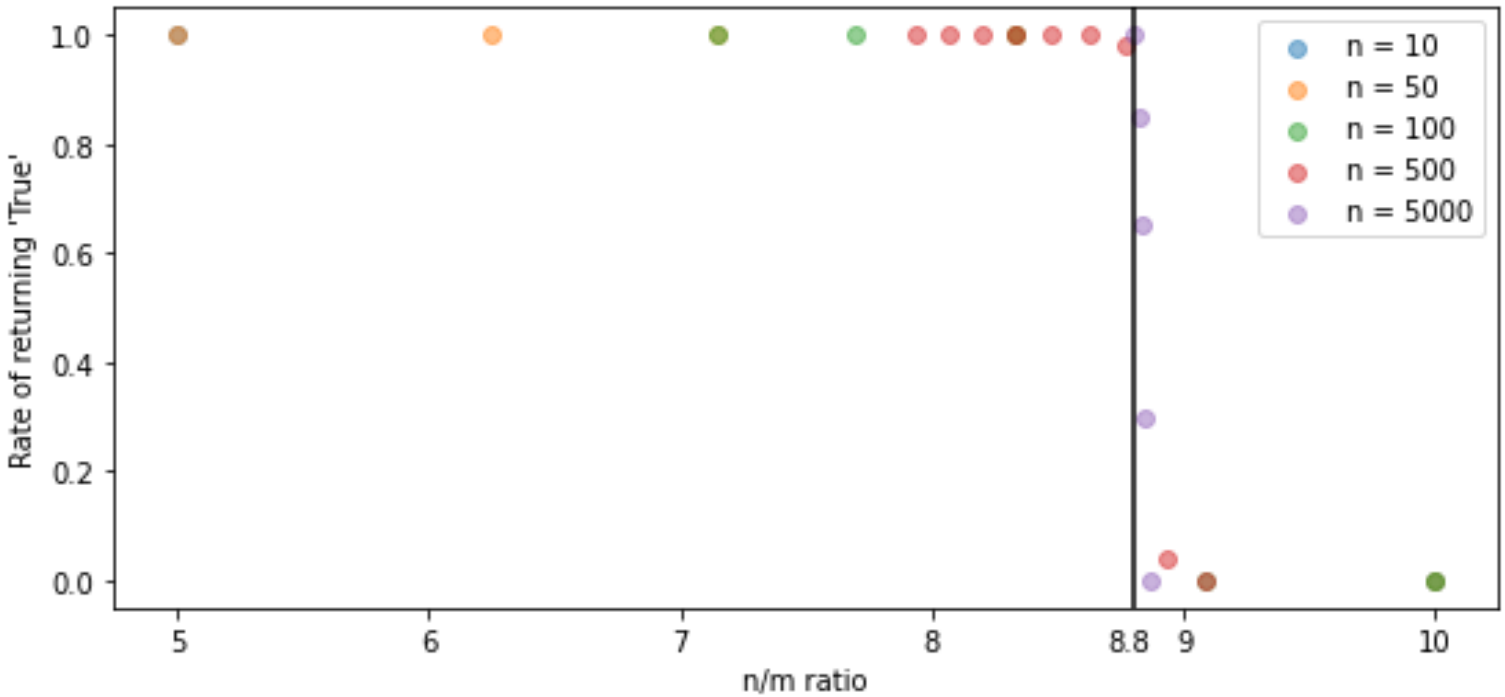
$\longrightarrow$ To establish a timetable, we must ensure that $\frac{n}{m} \leqslant 10$.

However, if $\frac{n}{m} \approx 10$, it is extremely difficult to create a schedule that accommodates course conflicts, and not every classroom can be utilized optimally (for example: classes with a smaller capacity will be likely to have more free time slot because there are fewer courses that can use this classroom)

$\longrightarrow$ If our algorithm can properly schedule the cases of n/m = k, with a large enough k (for instance, k = 8), then we may infer that our approach produces acceptable results.

Experimenting with different values of n, m, each pairs several times, record the corresponding rate of returning *True*, we have the following scatter plot





Conclusion: The experiment suggests that for all pair (n, m) such that $n/m \leqslant 8.8$, we are very confident that our algorithm will return a correct result (return True, and the corresponding schedule).

Thank you for listening!!

Special thanks to Nhật Tân for helping us with this project.