Vaccination Scheduling

Algorithms for Decision Support 2021

September 24, 2021

1 This assignment

This is the description of the group project for the course *Algorithms for Decision Support* 2021/2022.

2 Dates

- September 14, 2021: Groups should be formed.
- September 28, 2021: Hand in a test instance for the offline problem and a test instance for the online problem
- October 26, 2021: Group presentation
- October 29, 2021: Hand in scientific paper and software

3 Groups

Each group consists of four, five, or six participants of the course. We prefer that you form groups with five students. From groups with six participants, a better project is expected for the same grade. Groups with four participants will not be given a benefit.

You must inform the teachers of the group formation via Blackboard (see the respective assignment) ultimately on September 14, 2021.

4 Grading

The project has a number of different deliverables. The grade for a project depends on the following:

- The project has a number of minimum requirements. If these all are done 'reasonably well', then the grade will be 6.

- You can get a better grade by doing something additional, or performing in the minimum requirement tasks in a (very) good / excellent manner. More information is given below.
- In case the distribution of work among group members was very uneven, you can contact the teachers, and we can have a different grade for different members of the same project.

5 Minimum requirements

Each group must fulfill the following deliverables:

1. Two *test instances*.
2. A *project presentation*. At October 26, 2021, each group gives a presentation of 12 – 15 minutes of the project.
3. A *scientific paper*. The scientific paper is written with a specific format (described below), and handed in as a pdf-file.
   - The paper must follow a format, described below.
   - In the paper, you give at least one proof: for at least one strategy for the online problem, you give a proof of a competitive ratio.
   - The paper is at least eight pages long, but can be longer. There is no real upper limit to the size (but try to keep it below 50 pages.)
4. Two *programs*. One program gives exact solutions for (relatively) small instances of the problem, and one gives a strategy for the online version.

It is possible to *extend* the project, for a better grade (see below).

6 Problem definition

In an unnamed small country the population is to be vaccinated for an unnamed disease. In this assignment, the planning for giving the vaccination to the patients is to be made. There are a number of different aspects and constraints that are taken to be into account. Note that there are similarities and differences from the current practices of corona vaccination.

The vaccination is with *two-phase vaccine jabs*. Each vaccine jab has two doses, and a certain time gap is required between the two doses. Each dose must be given by a hospital.

After the time slot a patient having a dose, the patient may be required to stay further for examinations for a number of additional time slots, directly after receiving the dose. The time slot of having the dose together with the ones of staying for examinations form a contiguous and unsplittable time interval. The length of the time interval is the *processing time* of this dose. (For instance, if taking the dose takes 1 time slot, and the patient has to be examined for 4 time slots, then the processing time of the dose is 5 time slots.)

Each hospital can only take care (either give a dose or examine) of at most one patient per time slot.

A patient can submit an interval of time slots that he/she is willing to have the first dose. Our goal is to assign the time slots of the two doses for each patient (where both the preferences of the patients and the time gap between the two doses for every jabs are satisfied) with the minimum number of hospitals to operate the vaccine tasks.

We have the following values for an instance.

- Global parameters, which are the same for all patients:
  - the processing time of the first dose $p_1 \geq 1$,
  - the processing time of the second dose $p_2 \geq 1$,
  - and the time gap between the first and the second doses g.
- A set of jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$. Each job represents one patient. Each job has the following information:
  - For job $J_i \in \mathcal{J}$:
    * The first feasible interval $I_{i,1} = [r_{i,1}, d_{i,1}]$ for the first dose (given by the patient)
    * The first dose is scheduled at start time $t_{i,1} \in I_{i,1}$ such that $t_{i,1} + p_1 - 1 \leq d_{i,1}$. (This value has to be determined by the program.)
    * The patient-dependent delay x_i where $x_i \geq 0$.
    * The patient-dependent (second) feasible interval length ℓ_i where $\ell_i \geq p_2$
    * The second feasible interval $I_{i,2} = [t_{i,1} + p_1 + g + x_i, t_{i,1} + p_1 + g + x_i + \ell_i - 1]$ for the second dose. Note that this interval depends on the start time for the first dose as was determined by the program, and the given values g, x_i, and ℓ_i.)
    * The second dose is scheduled at start time $[t_{i,2}, t_{i,2} + p_2 - 1] \in I_{i,2}$. (This value also has to be determined by the program.)
- Machine (hospital): at any time step, there can be at most one job executing. I.e., at each time step, each hospital can have at most 1 patient who received a dose or is in observation.
- Feasible schedule: For any job J_i, the first dose is scheduled at time interval $[t_{i,1}, t_{i,1} + p_1 - 1] \subseteq I_{i,1}$, and the second dose is scheduled at time $[t_{i,2}, t_{i,2} + p_2 - 1] \subseteq I_{i,2}$.
- Objective: Minimize the number of machines (i.e., hospitals).

In the offline version of the problem, the global parameters p_1, p_2, g, and the set of jobs with for each job the first feasible interval, second feasible interval, patient-depending delay, and patient-depending second interval length are given. The task is to find for each patient the time slot for the first and for the second dose such that all constraints are fulfilled, and the number of hospitals is as small as possible.

The online problem and online algorithm work as follows. At the start, the global parameters p_1, p_2, and g are given. Then, we have for each patient one round. At round i, we obtain all information for the ith patient: these are four integers, where the first two give the interval for the first dose, the third the delay for this patient, and the fourth the length of the second feasible interval. The program then has to schedule this patient: give the time and hospital when and where the first dose is given, the time and hospital when and where the second dose is given. These should fulfill the conditions as explained earlier. After this, we start the next round with the next patient.

6.1 Input and output

The general problem parameters are: the processing time of the first dose, the processing time of the second dose, and the minimal gap between the first and the second doses. The problem is: what is the minimum number of hospitals we need to schedule all the patients. You should also provide a feasible schedule for the patients using the claimed number of machines.

6.1.1 The offline problem

For the offline problem, you know the number of patients and their preferred feasible intervals. For each patient, the input consists of a feasible interval of the first dose (that is, the first and the last time slot of the feasible interval), a patient-dependent delay, and a patient-dependent length of the second dose.

The output. You have to give the minimum number of machines needed to serve all the patients. Moreover, there should be a feasible schedule provided. That is, for each patient, you should indicate when (the time slot) and where (the machine/hospital) the patient is given the vaccines.

Example. For the case where the processing time of the first dose is 2, the processing time of the second dose is 3, gap is 1, and there are 3 patients, the input will be:

```
2
3
1
3
2, 7, 5, 4
1, 2, 3, 3
2, 4, 0, 5
```

Here the first line is the processing time of the first dose, the second line is the processing time of the second dose, the third line is the minimum gap between the first and the second dose, and the fourth line is the number of patients. After these four lines, each line is the input for one patient that consists four integers: the first available time slot of the first dose, the last available time slot of the first dose, the patient-dependent delay between the first and the second doses, and the length of the second dose feasible interval. The first patient is the one shown in Figure 1.

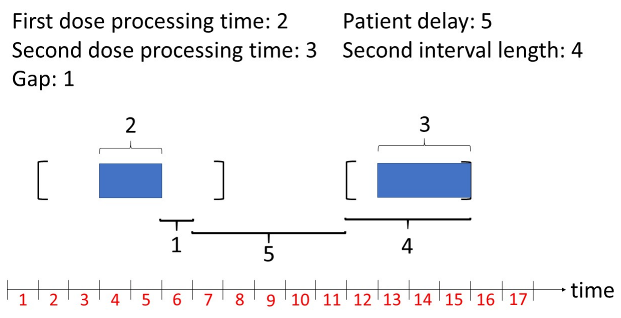
First dose processing time: 2
Second dose processing time: 3
Gap: 1
Patient delay: 5
Second interval length: 4
2
3
1
5
4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
time

Figure 1: A patient with delay 5 and second feasible interval length 4.
The first feasible interval is $[2, 7]$ and $t_{i,1} = 4$.
The second feasible interval is $[4 + 2 + 1 + 5, 4 + 2 + 1 + 5 + 4 - 1] = [12, 15]$.

For this instance, the output can be:

```
4, 1, 13, 1
1, 1, 7, 2
3, 2, 6, 1
2
```

In the output, the first n lines are the schedule of the patients. In the end, there is another line stating the minimum number of hospitals needed to serve all the patients. Each line of the patient schedule is a four-tuple $(t_{i,1}, m_{i,1}, t_{i,2}, m_{i,2})$, where $t_{i,k}$ and $m_{i,k}$ are starting time and hospital of the k-th dose of the i-th patient, respectively.

6.1.2 The online problem

Input example. In the online problem, we get as input almost the same as the offline problem, except that there is no information about the number of patients:

```
2
3
1
2, 7, 5, 4
1, 2, 3, 3
2, 4, 0, 5
```

`x`

At some point, the program receives an x and that is the end of the input.

As soon as the program receives a line about a patient, it must assign the patient to a timeslot and a hospital. That is, your algorithm should work as the following:

- Read the first line for the processing time of the first dose (p_1)
- Read the second line for the processing time of the second dose (p_2)
- Read the third line for the time gap (g)
- Read a line of four integers representing the first patient's preference: the first available time slot of the first feasible, the last available time slot of the first feasible interval, the patient's delay, and the length of the second feasible interval ($r_{1,1}, d_{1,1}, x_1, \ell_1$)
- **The algorithm should schedule the first patient. That is, deciding the time step $t_{1,1}$ when his first dose $t_{1,1}$ is administered, the id of the hospital of his first dose $m_{1,1}$, the time step $t_{1,2}$ when his second dose is administered $t_{1,2}$, the id of the hospital of his second dose, $m_{1,2}$.**
- Read a line of four integers representing the second patient's preference: the first available time slot of the first feasible, the last available time slot of the first feasible interval, the patient's delay, and the length of the second feasible interval ($r_{2,1}, d_{2,1}, x_2, \ell_2$)
- **The algorithm should schedule the second patient. That is, deciding the time step $t_{2,1}$ when his first dose $t_{2,1}$ is administered, the id of the hospital of his first dose $m_{2,1}$, the time step $t_{2,2}$ when his second dose is administered $t_{1,2}$, the id of the hospital of his second dose, $m_{2,2}$.**
- Read a line of four integers representing the third patient's preference: the first available time slot of the first feasible, the last available time slot of the first feasible interval, the patient's delay, and the length of the second feasible interval ($r_{3,1}, d_{3,1}, x_3, \ell_3$)
- **The algorithm should schedule the third patient. That is, deciding the time step $t_{3,1}$ when his first dose $t_{2,1}$ is administered, the id of the hospital of his first dose $m_{3,1}$, the time step $t_{3,2}$ when his second dose is administered $t_{1,2}$, the id of the hospital of his second dose, $m_{3,2}$.**
- $\cdots$
- The algorithm reads a line consisting a single `x`
- **The algorithm stops.**

7 Test instances

At September 28, 2021, you should hand in two test instances:

- An instance for the offline problem
- An instance for the online problem

Both should be correct, in the sense that they fulfill the description given above. You are allowed to hand in instances where your own programs work well.

Hand in two files, in normal ASCII text, via Blackboard, at or before September 28, 2021.

It is allowed to hand in more test instances.

8 The scientific paper

Each group writes a scientific paper on the project. The paper must have a specific format. This format is provided via the MS Teams site of the course.

The format includes the following:

- The paper is written in LaTeX, following the Lipics style file as provided. The paper has author names, an abstract, keywords, main text (see below), proper formatted reference list, and optionally an appendix. (More specifically, the lipics-v2019 style is used. You can download the necessary files from the Teams site, or from `https://submission.dagstuhl.de/documentation/authors#lipics`
- The main text must have the following sections:
  - Introduction. In the first section, you describe the problem, possibly describe related scientific literature, and describe in brief sentences (or as formatted theorems) your results.
  - Preliminaries / Definitions. In the second section, you give definitions of mathematical notions that you use or need. If you use well known scientific results / lemmas / theorems / algorithms as subroutine, you can also mention these here.
  - One or more chapters that: describe the algorithms you use, prove a competitive ratio for at least one case of an online strategy, and optionally give more theoretical results.
  - One or more chapters that: give the results from the experiments. Describe the setting (what computer(s) did you use to test, what programming language did you use, what OS, etc.), and give the results of the experiments. Try to draw some conclusions from your experiments. These should report on all or some of the provided test instances. You can use more test instances that you made or found yourself, if you want.

    - Conclusions. In a (possibly short) last chapter, you repeat the main findings of the project, and possibly give ideas for further research, open problems, etc.

- In the (optional) appendix, you could give additional information, e.g., additional long tables with results from experiments, etc.
- The reference list must be proper formatted, following standard style of scientific writing.
- You should follow rules of scientific integrity. This includes that you cannot use images that are copyrighted by others, and give a reference for non-copyrighted images you use. It is always better to make images yourself. Of course, you should never commit fraud (e.g., changing data) or plagiarism; we follow the rules of Utrecht University here.

You hand in the paper at or before October 29, 2021 via blackboard.

9 A competitive ratio

Your scientific paper should at least contain a description of one online strategy and an competitiveness analysis.

- The competitiveness analysis can be:
    - an upper bound of the competitive ratio of your algorithm,
    - a lower bound of the competitive ratio of your algorithm, or/and
    - a general lower bound of the problem.
- For the theory part, you can focus on an online algorithm for special inputs (see Section 12).
    - The smaller gap between the upper bound and lower bound of your algorithm you prove, the higher grades you get.
    - The more general instance you can provide competitiveness guarantee, the higher grades you get.

10 Programs

At or before October 29, 2021, the programs should be made visible to the teachers. You can do this by uploading in the specific Blackboard assignment the URL of a public visible code.

The programs should be well written, and commented.

The online program is *not allowed to cheat*: it should output the assignment of a group directly after the group is given, and not (try to) read sizes of future groups. If we find that a project has a cheating online strategy, then we divide the project score by 2.

11 Presentations

At October 26, 2021, each group gives a presentation of at least 12 and at most 15 minutes of the project. These will be given on the campus. There are four parallel sessions.

The presentation can be given by one, some or all group members.

Presence at the presentation of your groups, and the other groups in your session is obligatory, except when you have a good reason not to come. Fear of corona-infection is regarded as a good reason. If you cannot come to the presentation session, you should inform the teachers BEFORE the presentation session, when possible, and otherwise you must inform the teachers after the session, with the reason of absence. We can have a deduction from a grade if the reason for absence is not considered to be valid by the teachers, or came too late. Failing to contact the teachers in case of absence is regarded as an uncompleted course.

12 Extensions

For a better grade, you can add additional features to the project. These can include:

- Additional theoretical results. Can you prove (maybe with additional conditions), the exact problem NP-complete? Can you give polynomial time approximation algorithms with a proven approximation ratio? Can you give polynomial time algorithms for special cases? Can you give an time bound of the form $O(c^n)$ for some c of the exact algorithm? Etc.
- Better algorithms for special cases, either in theory, or in your program, or both. The special inputs include but not limit to:
  - Unit-length doses: $p_1 = p_2 = 1$
  - Restricted-length doses: $p_1 = 1$ and $p_2 = 2$
  - Less-flexible feasible intervals: $|I_{i,1}| = 2 \cdot p_1$ and $\ell_i = 2 \cdot p_2$ for every job J_i
  - Empty-gap: $g = 0$
  - Uniform-length feasible intervals: $|I_{i,1}| = c_1$ and $I_{i,2} = c_2$ for all jobs J_i, where c_1 and c_2 are constants
  - One hospital.
- Very good algorithms that give better results than that of other groups on our test instances. (Better can be: closer to optimal, faster algorithms, program can handle larger inputs, etc.)
- But maybe you have other ideas for extensions that are interesting? When in doubt, ask the teachers via MS Teams.

13 Rules of conduct

- Do not commit fraud or plagiarism.
- Give all your sources. It is allowed to use free software from the internet, but you should mention this in the report.
- Do the project in your group. Do not use the help of others, except the staff of this course. If in doubt, consult the teachers. Do not cooperate with other groups.
- Have a fair division of work within the group. If at the end of the project, you find that the division of work was not fair, you should report this to the teachers.
- Do not start this course and this project if there is a large chance you cannot complete it. Not completing the course makes the life for the other students much harder, and is unkind, impolite and unfair towards them. Do a fair share of the work of the team, start in time, and keep your promises to the other team members.
- Start in time; as a group, do a fair division of work, and make a good planning.
- If you have questions, you can contact the teachers/staff of this course via the MS Teams group of this course. Handing in materials is via Blackboard; other communication to the teachers via MS Teams team of the project.