Borrels Scheduling

Algorithms for Decision Support 2024/2025

September 5, 2024

Contents

1 Introduction

This is the description of the group project for the course *Algorithms for Decision Support* 2024/2025. In this project, you have to do an experiment, write a scientific paper, and give a presentation. The scientific paper should include at least one algorithm, at least two theoretical results, and a discussion of the experiment.

Throughout the term, there are six tasks about this project:

- September 13 (Friday): Hand in a list of your group member
- September 13 (Friday): Hand in a test instance for the offline problem
- September 27 (Friday): Hand in a proposal of your project
- October 11 (Friday): Hand in a paper outline that includes your algorithm
- October 25 (Friday): Hand in your complete scientific paper and program
- October 29 (Tuesday) or October 31 (Thursday): Group presentation

For details of each task, see Section 3.

If you complete every task properly, you get at least 5 points out of 10. If your scientific paper indeed contains the demanded algorithm, (at least two) theoretical results, and the program, you get another 1 point out of 10. For more detailed grading rules, see Section 4.

Also, note that this is a project, not an exam. Do not expect that you will get a full mark.

2 Borrel!

Imagine that you work for the student organization and would like to plan several *borrels* for the freshers. However, the students have many obligations, e.g., exams, deadlines, group meetings, etc. These obligations are flexible and can be rescheduled or even preempted, but they need to be finished eventually for each student. Students with conflicting schedules between their obligations and a borrel will prioritize their obligations over the borrel. In this project, the goal is to schedule borrels such that as many students attend the borrels as possible.

2.1 Formal problem definition

Timeline is partitioned into t integral *time slots*. The time interval from time 0 (inclusively) to time 1 (exclusively) is time slot 1, and the time interval from time $i-1$ (inclusively) to time i (exclusively) is time slot i for all $i \geq 1$.

There are m borrels, and each borrel h has a length of b_h time slots. There are n students. Each student i has a set of obligations $\mathcal{I}_i = \{(I_{i,1}, p_{i,1}), (I_{i,2}, p_{i,2}), \cdots, (I_{i,\ell_i}, p_{i,\ell_i})\}$,

where $I_{i,j} = [r_{i,j}, d_{i,j}]$ is a time interval from time slot $r_{i,j}$ to time slot $d_{i,j}$, and $p_{i,j}$ is the time slots needed by the obligation j that has to be finished within $I_{i,j}$. A *feasible schedule* for a student i of obligation j is a schedule $S_{i,j}$, which is a set of time slots, such that $S_{i,j} \subseteq I_{i,j}$ and $|S_{i,j}| = p_{i,j}$. Moreover, for a student i and two of their obligations j and j', $S_{i,j} \cap S_{i,j'} = \varnothing$. Note that we assume $p_{i,j}$ are integral for all i and j, and we cannot assign a time slot partially to an obligation.

As a borrels planner, you aim to schedule each borrel h starting at time slot s_h and lasting until time slot $s_h + b_h - 1$, such that as many students attend the borrels as possible. More formally, you want to find the start time s_h of each borrel h such that there exists a set of feasible schedules $S_{i,j}$ for every pair of student i and obligation j such that the *conflict* among the borrels and the students' schedules, $[s_h, s_h + b_h - 1] \cap S_{i,j} \neq \varnothing$ for some i and j, is minimized. We also assume that borrels start at integral time (that is, the beginning of a time slot).

Offline setting. In the *offline setting*, all the parameters are known to the algorithm from the very beginning. That is, the algorithm knows t, m, b_h for all $h \in [1, m]$, n, and $\mathcal{I}_i$ for all $i \in [1, n]$. Once the algorithm receives the information, it should suggest schedules for all borrels.

Online setting. In the *online setting*, the online algorithm knows the number of time slots (t), the number of students (n), the number of borrels (m), and all b_h for each borrel h in the beginning but only learns about $I_{i,j} = [r_{i,j}, d_{i,j}]$ and $p_{i,j}$ at the beginning of time slot $r_{i,j}$ (that is, at time $r_{i,j} - 1$). The online algorithm has to decide if a borrel will start at time slot $x + 1$ right at time x (that is, the beginning of time slot $x + 1$). Formally, at time x (that is, the beginning of time slot $x + 1$), the algorithm first learns the obligations that have release time at time slot $x + 1$ and then decides if it wants to schedule a borrel starting from time slot $x + 1$.

2.2 Input and output formats for the experiment

The general problem parameters are:

- t: the number of time slots.
- m: the number borrels,
- b_h: the h-th borrel's length (that is, number of time slots),
- n: the number of students,
- $I_{i,j} = [r_{i,j}, d_{i,j}]$: the time interval for obligation j of student i, and
- $p_{i,j}$: length of obligation j of student i.

Example of input. The following is an example where there are 10 time slots, 2 borrels, and 4 students:

```
10
2
1, 3
4
2, 1, 4, 3, 5, 10, 6
1, 2, 5, 1
1, 4, 10, 3
3, 1, 3, 1, 1, 6, 5, 3, 9, 2, 8, 8, 1
```

The first line is the number of time slots (10). The second line is the number of borrels (2). The third line contains the length of each of the borrels. In this case, $b_1 = 1$ and $b_2 = 3$. The fourth line is the number of students (4). Each of the following 4 lines i starts with the number of obligations of student i. For the rest of the line, every three numbers represent the start time slot, deadline, and length of an obligation of i. In this example, the first student has 2 obligations. The first one starts from time slot 1, has a deadline at time slot 4, and needs 3 time slots to finish. The second one starts from time slot 5, has a deadline at time slot 10, and needs 6 time slots to finish.

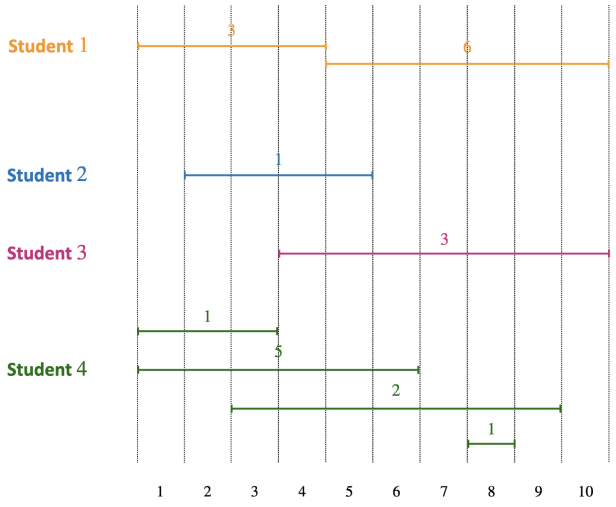

Figure 1: An instance

Example of output. In the output, there should be $1 + \ell_1 + \ell_2 + \cdots + \ell_n + 1$ lines. (Recall that ℓ_i is the number of obligations of student i.) The first line consists m numbers, which are the starting times of borrel 1, 2, $\cdots$, and m. The $(1 + j)$-th line, where $j = 1, 2, \cdots, \ell_1$, has $p_{1,j}$ numbers, each is a time slot that student 1 works on their j-th obligation. Similarly, the $(1 + \ell_1 + j)$-th line with $j = 1, 2, \cdots, \ell_2$ has $p_{2,j}$ numbers, each is a time slot that student 2 work on their

j-th obligation, and so on. The last line is the total number of attendance of the borrels.

For the instance mentioned above, the output can be:

```
10, 5
1, 2, 3
5, 6, 7, 8, 9, 10
2
4, 8, 9
1
2, 3, 4, 5, 6
7, 9
8
5
```

Note that the last line indicates that the objective value is 5, which comes from the fact that students 2 and 3 attend both borrels, and student 4 attends the borrel with a length of 1 time slot. The following figure illustrates the output. The time slots assigned for borrels are marked in gray.

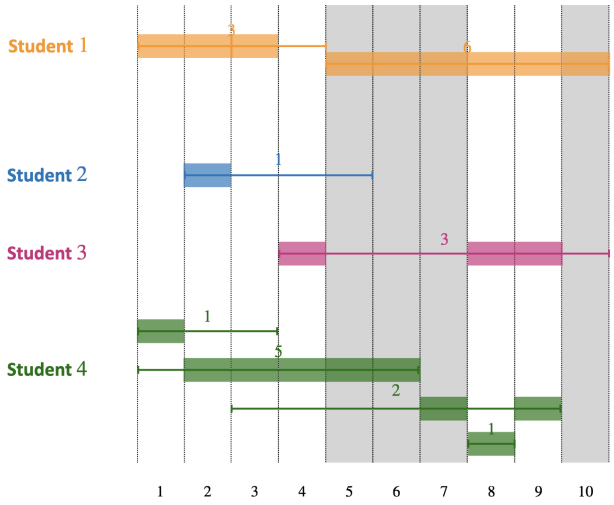

Figure 2: Output of the example

3 Requirements

3.1 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 by **September 13**.

3.2 Test (offline) instances

Each group should hand in at least one test instance for the offline problem, together with the correct optimal answer (that is, the decisions on each day). The submitted instance should be different from the example in subsection 2.2.

Note that the instance(s) should be correct in the sense that they fulfill the description given above. It also needs to be *feasible*. That is, there should be a schedule for each student to finish their obligations. You are allowed to hand in instances where your own programs work well.

By **September 13**, the instance(s) and the corresponding answers should be handed in via Blackboard in one normal ASCII-text file. You are allowed to hand in more test instances.

3.3 Proposal

Each group should submit a short proposal by **September 27**. The proposal should include the potential cases you want to work on (see subsection 4.2 for examples) and the way you plan to tackle them. Optionally, you can add references to the papers you find for your project.

The proposal can be extended to the Introduction section of your final paper. However, it should not be too long at this stage. One page is enough.

Note that a proposal is not a promise. If you find a more promising direction later, you can change your plan.

3.4 Paper outline

By **October 11**, each group should submit a paper outline. The outline should include the introduction, preliminaries, your algorithm(s), and a plan of your expected results.

- **Introduction.** This should contain the general problem description, a literature review, and the problems/special cases you want to solve. If you have any cool techniques that you use in your research, you can also advertise them here. You can use your proposal in this part.

- **Preliminaries/Definitions.** This should contain the definitions of mathematical notions that you use or need. If you use well-known scientific results/lemmas/theorems/algorithms as subroutines, you can also mention these here.
- **Algorithm(s).** The most important task at this stage is to have some algorithm(s) well-defined now. You might need to refer to the notations defined in the preliminaries.
- **Expected results.** For example, what theorems are you going to argue? What intermediate lemmas do you need to prove the main theorems?
  - It is encouraged to have your section titles and rough contents ready. As an example, Section 1 is the Introduction, Section 2 is the Preliminaries, Section 3 is the offline algorithm and analysis, Section 4 is the online algorithms and analysis, and Section 5 is the Conclusion. Note that according to your expected results and targets you made in the proposal, you may have a different structure for your paper. For example, your Sections 3, 4, and 5 may be for different special cases, and Section 6 is the conclusion.

3.5 The scientific paper

Each group should write a scientific paper on the project. The paper should be written in LaTeX, following the LIPIcs style file as provided. The paper has author names, an abstract, keywords, a main text (see below), a properly formatted reference list, and optionally an appendix. More specifically, the LIPIcs-v2021 style is used. You can download the necessary files from `https://submission.dagstuhl.de/series/details/LIPIcs#author`. When you submit your paper, please **keep the line numbers**. You hand in the paper by **October 25** via blackboard.

- **Main text.** For completeness, we repeat some parts mentioned in Section 3.4. The main text must have the following sections:
  - **Introduction.** In the first section, you describe the problem and related scientific literature and describe your results in brief sentences (or as formatted theorems).
  - **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 subroutines, you can also mention these here.
  - One or more chapters include **at least one algorithm** and **at least two theoretical results**. It can describe the *algorithm(s)* you use, prove a *competitive ratio* for some case(s) of an online strategy, prove NP-completeness, prove polynomial time solvability, etc.

  - One chapter that gives the results from the **experiments**. Describe the setting (what computer(s) you used to test, what programming language you used, 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.
  - **Conclusion.** In a (possibly short) last chapter, you repeat the project's main findings 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 properly formatted, following the standard scientific writing style.

You should follow the rules of scientific integrity. This includes that you cannot use images 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.

3.6 Experiments

By **October 25**, the program should be made visible to the teacher. You can do this by uploading to the specific Blackboard assignment or to the URL of a public-visible code.

The program should be well written and commented.

3.7 Presentations

On **October 29** or **October 31**, each group gives a presentation of at most 10 minutes of the project.

Given the short time, you are encouraged to present the result that you are proud of the most. The presentation can be given by one, some, or all group members.

Presence. 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. If you cannot come to the presentation session, you should inform the teachers BEFORE the presentation session. We can have a deduction from a grade if the reason for absence is not considered to be valid by the teacher, or came too late. Failing to contact the teacher in case of absence is regarded as an uncompleted course.

4 Grading

Overall, the project will be graded according to the Master thesis rubrics and conference paper guidelines. There are a number of different deliverables. The grade for a project depends on the following:

- The project has a number of minimum requirements (see subsection 4.1). 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 the minimum requirement tasks in a (very) good/excellent manner. In general, a paper is appreciated if it consists of the following:
  - studies of the general case or interesting/critical special cases that fills research gaps,
  - finding surprising results, and/or
  - usage of advanced techniques.

  More information is given below in subsection 4.2.
- 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.

4.1 Minimum requirements

Each group must fulfill the following deliverables:

1. **By September 13:** At least ONE feasible **test instance** (for the offline version).
2. **By September 27:** A **proposal** about your plan on the project.
3. **By October 11:** An **outline of your paper**.
4. **By October 25:** A complete **scientific paper** contains specific theoretical elements (described in Section 3.5) and is handed in as a pdf-file. Note that:
   - In the paper, you give at least ONE algorithm, which can be an offline or an online algorithm.
   - You have to give at least TWO proofs. A valid proof can be a time-complexity analysis, a correctness proof, a competitive analysis of an online algorithm, a lower bound of the competitive ratio of an online algorithm, a lower bound of the competitive ratio of the online problem, an NP-hardness proof, etc.

- The paper is at least six pages long but can be longer. There is no real upper limit to the size (but try to keep it concise and below 15 pages.) Note that a longer paper does not always get a higher grade, but a concise one does.

5. **By October 25:** One **program** for the offline version, which should give exact solutions for (relatively) small instances of the problem. Note that your program should be able to handle any feasible instance.

6. **On October 29 or October 31:** A *project presentation* of the project for 10 minutes. The exact date will be decided in the week of October 24.

It is possible to *extend* the project for a better grade (see subsection 4.2).

4.2 Possible variations

There are various directions of theoretical results. The following is a list of examples of the directions you can try.

- Prove that the problem is NP-complete (for some special instances or for more general instances).
- Offline optimal algorithm and its proof of correctness
- Prove that the problem is polynomial-time solvable (for some special instances or for more general instances).
- Give an online algorithm and show that it is at most constant-competitive.
- Give an online algorithm and show that it is constant-competitive for some special case.
- Give an online algorithm and show that it is at least c-competitive for some constant $c > 1$.
- A constant-competitive online algorithm for some special case (where the algorithm knows that the instance must be in the form of the special case)
- No online algorithm can be constant-competitive

You can have some combo attacks:

- Show that there exists an online algorithm that is exactly c-competitive for some $c > 1$ (for some special cases)
- Show that some online algorithm is optimal (for some special cases)

Note that although your program should deal with general input instances, in the paper for the theoretical results, you can tackle special cases for the problem. In special instances, you can impose restrictions on the number of borrels, the length of each borrel, the number of obligations of each student,

the range of feasible time intervals ($I_{i,j}$), length $p_{i,j}$ of the obligations, the ratio between the length of feasible time intervals and the length of obligation, etc. The special instances include but are not limited to:

- Only one borrel: $m = 1$
- Unit-size borrels: $b_h = 1$ for any borrel h
- Unit-size obligations: $p_{i,j} = 1$ for any student i and obligation j
- Super flexible students: $r_{i,j} = 1$ and $d_{i,j} = t$ for all students i and obligation j.
- Bounded flexibility of obligations: For all i and j, $d_{i,j} - r_{i,j} + 1 = c \cdot p_{i,j}$ for some (fixed) constant c.
- Bounded number of obligations: For any student i, there are at most c obligations for some (fixed) constant c.

Note that a special case can be rather easy. You get a higher grade if you challenge yourselves and solve the more challenging case(s).

4.3 Rules of conduct

- Do not commit fraud or plagiarism.
- Give all your sources. Free software from the internet is allowed, 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 it is unkind, impolite, and unfair towards them. Do a fair share of the work of the team, start on time, and keep your promises to the other team members.
- Start in time; as a group, do a fair division of work and make 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 course.