Scheduling to Save Your Life

Algorithms for Decision Support 2024/2025

September 1, 2025

Contents

1 Introduction

This is the description of the group project for the course *Algorithms for Decision Support* 2025/2026. 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 for this project:

- September 5 (Friday): Hand in a list of your group members
- September 12 (Friday): Hand in a test instance for the offline problem
- September 26 (Friday): Hand in a proposal for your project
- October 10 (Friday): Hand in a paper outline that includes your algorithm
- October 24 (Friday): Hand in your complete scientific paper and program
- October 28 (Tuesday) or October 30 (Thursday): Group presentation

For details of each task, see Section 3.

If you complete every task *correctly*, 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 to get a full mark.

2 Scheduling to save your life!

As a Master's student, your life is an ongoing sequence of overlapping demands: attending lectures, writing final reports, catching up on two weeks of unread slides, attending a birthday party, doing laundry, grocery shopping, calling home, applying for internships, or going on a sudden "we-need-to-talk" date.

Each of these tasks requires a certain amount of time to complete and must be done within a specific time window. Some tasks are flexible, like reviewing slides between 1 a.m. and 3 a.m., while others are less negotiable, such as showing up to a 9 a.m. lecture (physically and spiritually).

Completing a task brings a reward: a sense of pride, higher grades, momentary peace, or even continued romantic stability. Skipping a task may lead to penalties: public shame, awkward group meetings, poor academic performance, or a broken heart. Fortunately, some tasks can be paused and resumed later if you remember to resume them.

Given that your cognitive bandwidth allows you to focus on only one task at a time, your goal is to schedule your life in a way that maximizes total reward and minimizes total penalty. In other words, schedule wisely, or suffer the consequences.

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 is one single machine, and the machine can execute at most one job at a time. There is a set N of $n \geq 1$ jobs $\{1, 2, \cdots, n\}$. Each job $j \in N$ has release time r_j, deadline d_j, and processing time p_j. For simplicity, we assume that r_j, d_j, and p_j are integers for all i. If job j is completely executed (for a length of p_j, not necessarily consecutive) within the interval from r_j to d_j, the schedule wins a reward of w_j. Otherwise, the schedule has to pay a penalty ℓ_j, where w_j and ℓ_j are at least 0. Note that the jobs can be preempted. That is, an executing job can be interrupted and resume execution later (which does not need to be on the same machine). However, a job cannot execute on different machines at the same time. Namely, parallelization is not allowed.

As a scheduler, you aim to schedule some of the jobs j at a set of time slots. Let S_j be the set of time slots where job j is executed. A *feasible* schedule respects the feasible intervals. That is, $S_j \subseteq \{r_j, r_j + 1, \cdots, d_j\}$. A job j is *finished* by the schedule if it occupies at least p_j time slots within the interval from r_j to d_j.

The goal of the scheduler is to find a feasible schedule of the jobs that maximizes the total reward minus the total penalty. Formally, let F be the jobs that finished by the scheduler, the total profit of the schedule is $\sum_{j \in F} w_j - \sum_{j \in N \setminus F} \ell_j$

Offline setting. In the *offline setting*, all the parameters are known to the algorithm from the very beginning. That is, the algorithm knows n, and r_j, d_j, p_j, w_j, and ℓ_j for all job $j \in [1, n]$. Once the algorithm receives the information, it should suggest schedules S_j for all jobs j.

Online setting. In the *online setting*, all the information (number of jobs, release times, deadlines, etc.) is unknown to the online algorithm initially. The online algorithm gets all information about job j at its release time r_j, including processing time p_j, deadline d_j, reward w_j, and penalty ℓ_j. The online algorithm has to decide if a job will execute at time slot $x+1$ (on a machine i) right at time x (that is, the beginning of time slot $x+1$). Note that the schedule is not obliged to finish any job that is in execution. That is, if a job j is executed but not finished, the scheduler can give up this job (perhaps due to another, more profitable job). In this case, the schedule pays the penalty for ℓ_j and gains no reward.

Note that the optimal solution can have a negative total profit. In this case, the competitive ratio of algorithm ALG is defined by $\max_I \frac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)}$. Overall, the competitive ratio of algorithm ALG is defined by $\max_I \{\max\{\frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}, \frac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)}\}\}$ ($\mathrm{OPT}(I)$ is considered to be $-\varepsilon$ when $\mathrm{OPT}(I) = 0$).

2.2 Input and output formats for the experiment

The general problem parameters are:

- n: the number of jobs,
- r_j: the release time of job j (that is, the first time slot that job j can be executed),
- d_j: the deadline of job j (that is, the last time slot that job j can be executed),
- p_j: the processing time of job j,
- w_j: the reward of job j when it is finished, and
- ℓ_j: the penalty of job j when it is not finished.

Example of input. The following is an example where there are 5 jobs:

```
5
1, 9, 4, 10, 10
3, 5, 2, 10, 15
2, 7, 5, 10, 5
6, 7, 2, 1, 100
2, 3, 1, 3, 1
```

The first line is the number of machines (2). The second line is the number of jobs (5). Each of the following 5 lines j contains the release time r_j, deadline d_j, processing time p_j, reward w_j, and penalty ℓ_j. In this example, the first job has release time 1, deadline 6, processing time 3, reward 9, and penalty 4. The second job has release time and deadline both at time slot 2, processing time 1, reward 2, and penalty 11.

Example of output. In the output, there should be $n + 1$ lines. The first line consists of all the time slots in which job 1 is executed. The second line consists of all the time slots in which job 2 is executed. More generally, the j^{th} line consists of all the time slots that job j. The last line is the total profit (that is, total reward minus the total penalty).

For the instance mentioned above, the output can be:

```
1, 3, 8, 9
4, 5
null
6, 7
2
19
```

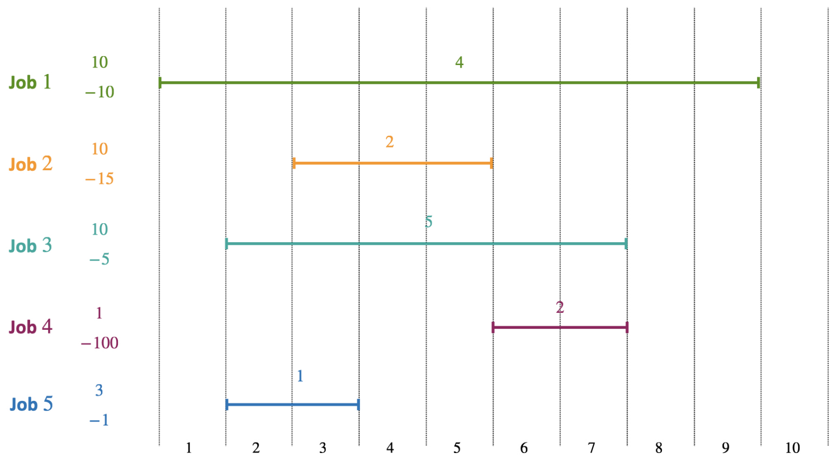

Figure 1: An instance

Note that the last line indicates that the objective value is 16, which comes from the fact that all jobs but the third one are finished. The following figure illustrates the output.

3 Requirements

3.1 Groups

Each group consists of four, five, or six participants in 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 BrightSpace (see the respective assignment), ultimately by **September 5**.

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 12**, the instance(s) and the corresponding answers should be handed in via BrightSpace in one normal ASCII-text file. You are allowed

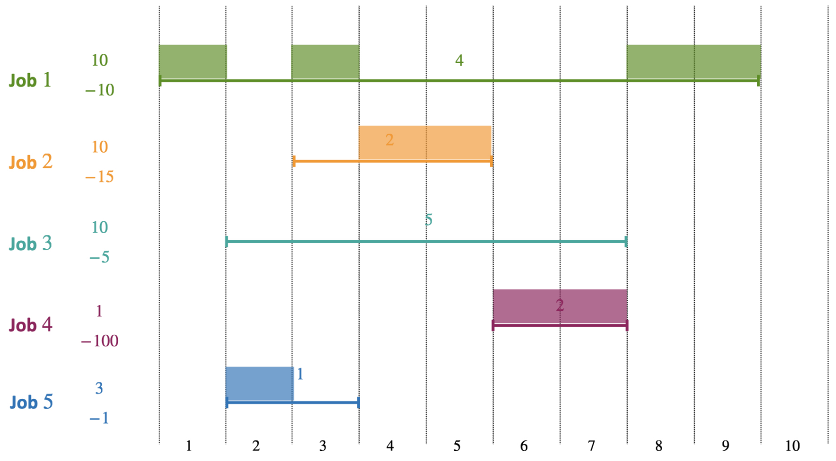

Figure 2: Output of the example

to hand in more test instances.

3.3 Proposal

Each group should submit a short proposal by **September 26**. 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 10**, 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 24** via BrightSpace.

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

The program should be well written and commented.

3.7 Presentations

On **October 28** or **October 30**, 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 all these are done *correctly*, 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 fill 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 12:** At least ONE feasible **test instance** (for the offline version).
2. **By September 26:** A **proposal** about your plan on the project.
3. **By October 10:** An **outline of your paper**.
4. **By October 24:** 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 24:** 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 28 or October 30:** 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 of the problem. In special instances, you can impose restrictions on the number of machines, the processing of each job, the ratio between the processing time and the length of the feasible interval of any job, etc. The special instances include, but are not limited to:

- All jobs arrive at the same time: $r_j = 1$ for all jobs j.
- All jobs have the same deadline: $d_j = d$ for all jobs j, where d is fixed.
- Uniform-processing time jobs: $p_j = p$ for all job j. (It is super easy when $p = 1$.

- Bounded flexibility of jobs: For all jobs j, $d_{i,j} - r_{i,j} + 1 \geq c \cdot p_j$ for some (fixed) constant c.
- Bounded ratio between the reward and the penalty: For all jobs j, $w_j = c \cdot \ell_j$ for some (fixed) constant c.
- (only for the online version) The optimal solution always has a positive total profit.

In such special cases, if there is a parameter c, the competitive ratio might be a function of c, which is fine.

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

There can also be more generalized versions. For example, you can consider that there are multiple machines. You can also investigate different objectives. For example, maximizing the total profit while the total penalty is within a given budget of penalties.

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 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.
- Every written deliverable will only be read once (or one version). Further changes/discussions should be conducted during office hours.
- 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 BrightSpace; other communication with the teachers via the MS Teams team of the course.