CS4210-A: Algorithms for Intelligent Decision Making
Part II: Game Theory and Mechanism Design
A social scheduling problem

Mathijs de Weerdt
M.M.deWeerdt@tudelft.nl

March 2020

This assignment is required. You may perform it in groups of two students. Assignment prepared together with Arthur Guijt and Max van Deursen

Read This First

The idea of this assignment is to experience the difference between regular optimisation and optimisation for situations with multiple agents. More concretely, we consider a case where multiple agents have preferences over a schedule, i.e., a sequence of a given set of jobs, and where these preferences are given by a most preferred schedule. The objective we focus on in this assignment is then to find a collective schedule which minimises the sum of tardiness (Σ-T) of all agents and all jobs compared to their preferred schedules, but which is also fair and is Condorcet-consistent. The problem is explained in more detail in a recent scientific publication by Pascual et al.

> Fanny Pascual, Krzysztof Rzadca, and Piotr Skowron. 2018. Collective Schedules: Scheduling Meets Computational Social Choice. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018*, IFAAMAS. `https://dl.acm.org/citation.cfm?id=3237482`

Provided code An exact implementation for minimizing Σ-T through a MiniZinc model is provided in `model.mzn`. This file takes in MiniZinc datafiles of the same format as `example.dzn`. Note that this datafile is simply to demonstrate the format of the datafiles and only serves this purpose. The parameters mean the following:

- `numJobs`: The number of jobs.
- `numAgents`: The number of agents.
- `processingTimes`: The processing times of each of the jobs.

- `preferences`: The order preference of the jobs of each agent. This means that for agent a, `preferences[a,1]` is the job that they wants scheduled first, `preferences[a,2]` as the second job, and `preferences[a, numJobs]` as the job which is scheduled last.

Your Task

As you may have noticed already, running the provided MiniZinc model is quite costly and may be undoable for large numbers of jobs. Additionally, it may result in schedules violating the PTA-Condorcet principle.

1. Find another voting rule to your liking, and provide pseudocode. Explain in your report why you chose this one, and cite the literature (or definition from the book) that you used.
2. Analyse the runtime efficiency and implement it.
3. Generate (at least) three benchmark sets with each at least 30 problem instances and motivate your choices:
   (a) one representative of "realistic" schedules,
   (b) two with extreme conditions on the instances (e.g. regarding similar/non-similarity of preferences or processing time distribution)
4. For each benchmark set, compare the results of "your" voting rule to the provided MiniZinc model that minimises Σ-T, on each of the following criteria. Write down your hypotheses about each criterion (possibly dependent on the properties of the respective benchmark set).
   (a) fairness (Gini index, you may use `stackoverflow.com/questions/31321810/gini-coefficient-in-julia-efficient-and-accurate-code`)
   (b) percentage of violations of the PTA-Condorcet principle
   (c) Σ-T
   (d) percentage of Pareto-efficient solutions[1]
   (e) runtime
5. Write down your observations on the results of these experiments.
6. Write a brief report about all your steps. If possible, draw a conclusion. Also indicate what you consider to be relevant next steps for this line of work.
7. Submit instances, code and a brief report explaining your approach and analysis through BrightSpace.

[1]Hint: instead of the Pareto-efficiency definition from the paper, you may use the provided MIP model for establishing Pareto-optimality of the individual sum-of-tardiness per agent.

Submission Instructions

All task answers, other than source code, must be in a single report in PDF format. Further:

- Identify the team members and state the group number inside the report. Your group number is assigned when you self-enrol to a group under category 'Assignment groups' on BrightSpace.
- Address each task of each problem, using the numbering and the ordering in which they appear in this assignment statement.
- Take the instructions of the relevant demo report on BrightSpace as a strict guideline for the structure and content of a model description, model evaluation, and task answer in the report, as well as an indication of the expected quality of the report.
- Write clear task answers, source code, and comments.
- Any graph you include should be in high resolution (clearly readable), using for example: gnuplot, or R (`www.r-project.org`); screenshots are not allowed.
- Justify all task answers, except where explicitly not required.
- State any assumptions you make that are not in this document.
- Please thoroughly proof-read, spell-check, and grammar-check your report.
- Upload all model files to BrightSpace in a single ZIP file without folder structure.
- Write a paragraph, which will not be graded, describing your experience with this assignment: Which aspects were too difficult or too easy? Which aspects were interesting or boring? This will help us improve the course in the coming years.
- Remember that when submitting you implicitly certify (a) that your report and all its uploaded attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, (b) that each teammate can individually explain any part starting from the moment of submitting your report, and (c) that your report and attachments are not freely accessible on a public repository.

Please submit two files on BrightSpace: one PDF file which is your report, and one ZIP file which contains (in the root of the ZIP file) all your model files. Submit once only, as a group on BrightSpace. For any BrightSpace questions, please contact `brightspace@tudelft.nl`.

Grading Rubric

We will check both your report as well as your code on correctness, clarity, quality and completeness with respect to the requirements. You will be graded based on the following point distribution. Note however that additional points can be subtracted if any of the requirements is not met.

(1p) Pseudocode of your voting rule and proper motivation and referencing.

(1p) Theoretical runtime analysis of your voting rule.

(2p) Implementation of the voting rule (e.g. in Java). Fewer points in case of mismatches between the pseudocode and implementation, or inefficient choices.

(1p) Benchmark sets and their motivation.

(3p) Experimental analysis.

(2p) Expectations of the experiments, a discussion of the actual results, and your ideas for conclusions and future work.

Grades and feedback will be given through BrightSpace.

Good luck!