# Project Work
# Combinatorial Decision Making & Optimization 2024/2025

**Topic**: Modelling & solving the Sports Tournament Scheduling (STS) problem
**Authors**: Zeynep Kiziltan and Roberto Amadini
**Date**: May 21, 2025

This document describes the project work of the Combinatorial Decision Making and Optimization course for the academic year 2024/2025, which is about modelling and solving the Sports Tournament Scheduling (STS) problem. The students can propose another problem from the literature provided that they get our approval before starting work on it. The project work involves approaching the problem using (i) Constraint Programming (CP), (ii) propositional SATisfiability (SAT) and/or its extension to Satisfiability Modulo Theories (SMT), and (iii) Mixed-Integer Linear Programming (MIP).

The students **must form a group** of 2-3 members, and it suffices for point (ii) to develop a SAT or an SMT solution. If they do both, they obtain bonus points. The students can also form a group of 4 members, but in that case they need to develop both SAT and SMT solutions. A student who cannot form a group must get our approval with convincing motivation.

## 1 Problem description

Organizing sports tournaments is a strategically important, yet complex task. As sports continue to grow in commercial value and global popularity, the demand to efficiently generate equitable tournament schedules has become increasingly critical. As the combinatorial decision and optimization expert, students are asked to solve the Sports Tournament Scheduling (STS) problem defined as follows. A tournament involving $n$ teams is scheduled over $n - 1$ weeks, with each week divided into $n/2$ periods (assuming that $n$ is even), and each period consists of two slots. In each period of every week, a game is played between two teams, with the team assigned to the first slot playing at home, while the team in the second slot playing away. The goal is to decide the home and away teams of all the games in a way that:

- every team plays with every other team only once;
- every team plays once a week;
- every team plays at most twice in the same period over the tournament.

An example schedule with 6 teams is shown in Table 1.

|  | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 |
|---|---|---|---|---|---|
| Period 1 | 2 v 4 | 5 v 1 | 3 v 6 | 3 v 4 | 6 v 2 |
| Period 2 | 5 v 6 | 2 v 3 | 4 v 5 | 6 v 1 | 1 v 4 |
| Period 3 | 1 v 3 | 4 v 6 | 2 v 1 | 5 v 2 | 3 v 5 |

Table 1: A schedule of 15 games over 5 weeks.

**Optional optimization** Optionally, in addition to the decision problem stated above, an optimization version can be implemented. In this case, the goal is to balance the number of home and away games of each team, to ensure fairness in the tournament. For instance, Table 1 shows a balanced schedule for $n = 6$.

## 2 Project work

The purpose of the project work is to model and solve the STS problem using (i) Constraint Programming (CP), (ii) propositional SATisfiability (SAT) and/or its extension to Satisfiability Modulo Theories (SMT), and (iii) Mixed-Integer Linear Programming (MIP). The students will decide themselves the decision variables, the problem constraints and the optional objective function around the problem description. The project work includes building models as well as conducting an experimental study using different search strategies (where applicable) to assess the performance of the solvers. The experimental study should consider progressively larger values of $n$ for which a solution can be obtained within the time limit.

**ATTENTION Since this is a well-studied problem in the literature, we encourage the students to explore the established approaches. It is important, however, to thoroughly understand and clearly explain the chosen method, cite it appropriately, and implement it independently.**

### 2.1 Project groups

The students **must form a group** of 2-3 members, and it suffices for point (ii) to develop a SAT or an SMT solution. If they do both, they obtain bonus points. The students can also form a group of 4 members, but in that case they need to develop both SAT and SMT solutions.

The names of the group members and their e-mail addresses must be indicated in the following sheet before project submission: `https://shorturl.at/rSfdR`. The sheet can be used also to write down individual names in search for partners and to form groups. A student who cannot form a group must get our approval with a convincing motivation before start working on the project.

**ATTENTION While group members may divide the implementation and report-writing tasks, each member remains fully responsible for the project development decisions.**

## 2.2 Project development

When developing the project, students should pay attention to the following:

- Start with the instance parameter, decision variables, domains, and then constraints.
- For the optional optimization model, consider constraining the objective function with tight lower and upper bounds.
- Though a trivial model is a good starting point, focus on the peculiarities of the underlying optimization approach and try to come up with the best model so as to be able to tackle the instances in the best possible ways.
- Investigate if there are any implied constraints that you can benefit from.
- Use global constraints to impose the main problem constraints and the implied constraints in your CP model.
- Optimize the model/encoding size and minimize redundancy.
- Observe if any symmetry can be exploited to improve the model by adding symmetry breaking constraints. When you add multiple symmetry breaking constraints, they should not conflict with each other (i.e., there must be at least one solution satisfying all the symmetry breaking constraints).
- Investigate the best way to search (where applicable) which does not conflict with the symmetry breaking constraints.
- The CP model **must** be implemented in MiniZinc and run using at least Gecode, whereas the SAT model **must** be implemented using at least Z3. For both SMT and MIP models, students can use their favourite theories, solvers and languages. Bonus points are considered if the problem is modelled with a solver-independent language, so as to try different solvers without rewriting the model. MiniZinc should be avoided here: models automatically derived from a MiniZinc model will not be considered. Students are welcome to exploit different solver parameter configurations.
- Set a time limit of 5 minutes (300 secs) to interrupt solving, as some instances may be too difficult to solve. Students are encouraged to apply best practices to obtain a solution and, in the case of the optimization version, to improve the quality of the solution, within the given time limit.
- If any *pre-solving* technique is used before invoking a solver then include the time taken for pre-solving in the reported total runtime.
- All solvers should be used in their sequential version; solvers cannot be run using multiple cores.

## 2.3 Project report

The students should describe clearly their work in a report by explaining the modelling choices, presenting the experimental study and commenting on results. While a MiniZinc notation could be acceptable to describe a CP model, it is required in general to use propositional logic, first-order logic and appropriate mathematical notations. **Do not copy and paste code!** The experimental results should show the total time took to obtain a solution for solved instances, and in the optimization case the solution quality obtained by the time limit.

**ATTENTION It is mandatory to produce one single report following the template provided in the Virtuale page. The report should be written in LaTeXusing `\documentclass{article}` without changing fonts, font size and margins. The page limits is 12 pages with three optimization approaches and 15 pages with all the approaches, excluding the authenticity and the author contribution statement and the references.** You can use `www.overleaf.com/` to produce a shared LaTeXdocument.

## 2.4 Solution format

Collect all the solutions under a `res` folder, by creating specific sub-folders (`res/CP`, `res/SMT`, `res/MIP`, etc.) to store the results of each optimization technology. For all the instances solved by a given model, you must present a corresponding `.json` file. For instance, the results of your CP model on instance #6 should be placed under the path `res/CP/6.json`. The keys of each `.json` file are the approach names presented in the experimental results of the report. Use meaningful names and be coherent with what you present in the report.

Each approach must be documented with its *runtime* (`time`, an integer), *optimality state* (`optimal`, a Boolean true iff the instance is solved for the decision version, or solved to optimality for the optimization version), *objective function value* (`obj`, a positive integer), which is set to `None` without an objective function, and the actual *solution* (`sol`). For the `time` field, use the *floor* of the actual runtime (e.g., if the actual runtime is 42.7s then `time` = 42). If an approach unexpectedly terminates before the timeout (300s) without solving to optimality, then set the `time` field to 300. In this way, `time` $= 300 \iff$ `optimal` $= false$.

The `sol` field must be an $(n/2) \times (n-1)$ matrix, represented as a list of lists, where each matrix entry is a list of two teams playing at home and away, in a given period and week. For example, the solution shown in Table 1 is represented as:

```
[
 [[2 , 4] , [5 , 1] , [3 , 6] , [3 , 4] , [6 , 2] ]
 [[5 , 6] , [2 , 3] , [4 , 5] , [6 , 1] , [1 , 4] ]
 [[1 , 3] , [4 , 6] , [2 , 1] , [5 , 2] , [3 , 5] ]
]
```

An example `.json` file looks as follows:

```
{
   "gecode":
   {
       "time": 200,
       "optimal": true,
       "obj": 1,
       "sol" : [ [[2 , 4] , [5 , 1], ...
   },
   "chuffed":
   {
       "time": 300,
       "optimal": false,
       "obj": 3,
       "sol" : [ [[1 , 3] , [4 , 6], ...

   },
   "gecode_symbreak":
   {
       "time": 100,
       "optimal": true,
       "obj": None,
       "sol" : [ [[1 , 3] , [4 , 6], ...

   }
}
```

Before the project submission, the correctness of the computed solutions must be checked with the solution checker (`check_solution.py`), which will be available on the Virtuale page. You are not allowed to make any modifications to the checker.

## 2.5 Project submission

The project will be submitted via Virtuale, and it shall contain the report and the results (as described in Sections 2.3 and 2.4), as well as the source code under a `source` folder, by creating specific sub-folders (`source/CP`, `source/SMT`, etc.)

To be reproducible, the source code must be executable through *Docker*. Inside the code folder, there must be the Dockerfile to build the *Docker* image, along with a README containing the instructions to run each of your models. The instructions should concisely explain how to reproduce the result of a single model on a specific instance, without modifying the source code of your project; for instance, you could provide a script that takes as input arguments the instance number and/or the approach chosen. Moreover, you should also provide a way to run all the models on all the possible instances automatically, aiming to generate all the feasible results with a single operation. The choices for the instructions format and the implementation of the *Docker* environment are completely up to you.

Keep in mind that the solvers need to be executed inside the Docker container defined by your Dockerfile, and not just in your local machine. Therefore, it is strongly suggested to use free software only. If commercial software is used, ensure reproducibility through *Docker* on another machine; alternatively, provide an open source alternative that can reproduce results. If neither of these conditions are met, the solution will not be considered valid.

**ATTENTION All the sources needed to reproduce the experiments (e.g., `.mzn`, `.dzn`, `.smt2` files) must be uploaded. Any submission where the `.json` files containing the solutions:**

- **cannot be processed by the checker, or**
- **contain errors reported by the checker**

**will be automatically rejected.**

Name your submission folder as `CDMO_Proj_LastnameName1_..._LastnameNameK` and upload its `tgz` or `zip` compressed archive. When working as a group, it suffices that only one group member submits the project, provided that the following information is provided in the **submission comments**:

- all the members' names, lastnames and UniBo e-mail addresses;
- the impossible dates and time for discussion due to a hard constraint (like another exam, medical appointment, employment, urgent family matter).

## 3 Project discussion and course assessment

Group members will be invited for an oral discussion of the project with one of the course teachers. No additional presentation is required. At each exam session, there will be a single date for submission and multiple dates for discussions. The discussion schedule will be devised taking into account students' hard constraints and will be communicated some days after the submission.

### Important dates

- July 2025: Submission 06/07, discussions during the week of July 14
- September 2025: Submission 06/09, discussions during the week of September 15

There will be other exam sessions in December 2025 and February 2026. Those who have not yet discussed by February 2026 will be able to submit later with the risk of having to wait until the summer of 2026 for discussion. Discussions will take place online on Teams.

The project evaluation will be based on the students' ability to model and solve a combinatorial optimization problem using the approaches and the tools discussed in the course, as well as the ability to present their work. Particular attention will be given to the literature search, originality, technical quality, and clarity of presentation. While it suffices to produce either a SAT or an SMT encoding for groups of max 3 students, bonus points will be given to those who produce both. Groups of 4 students are required to produce both. Bonus points will be given also to the groups that implemented the optimization version in addition to the decision version.

As part of the course assessment, students will be asked oral questions on the course topics during the project discussion. A successful overall evaluation will result in a mark between 18 and 30L.

A group member may reject the proposed mark and re-submit the project *only once*. In that case, the project can be discussed again only by the re-submitting group members and no new members can be added to the project. The final mark after the re-assessment may be higher, the same, or lower than the original mark, depending on the quality and significance of the revision.

ATTENTION

- **If you have any specific deadline by which you need to register the mark (for instance due to a scholarship, graduation etc), please contact the course teachers well in advance! We cannot guarantee to meet personal deadlines which have not been communicated earlier.**
- **Students on a mobility leave (Eramus+ Study or Traineship, Overseas, Thesis Abroad etc) during the discussion can register their marks only after their return to UniBo.**

## 4 Academic integrity

Intellectual development requires honesty, responsibility, and doing your own work. Plagiarism is the use of someone else's work, words, or ideas as if they were your own. Any idea taken from any person or resource should be cited appropriately. Plagiarised work will not be evaluated.

ATTENTION **The use of AI-generated text in the report is allowed but must be disclosed. The sections using AI-generated text, and in general any part of the project developed with AI tools, must reference the AI system used.**

**It is forbidden to use AI-generated CP/SAT/SMT/MIP models.**

## 5 Questions

For any questions regarding the project work and report of possible bugs in the solution checker, the students should use the discussion forum after checking the previous messages, in case a similar question was posted and answered already.