Combinatorial Optimisation Lab

Fundamentals of Artificial Intelligence Programme (IFEEMCS520100) 2024/2025 Q1

The goal is to gain hands-on experience with modelling and solving combinatorial problems.

MiniZinc. We will be using the MiniZinc modelling language. Once you devise a model in MiniZinc, you may use many of the available solvers to find solutions to your problem. Please visit the webpage, download MiniZinc, and familiarize yourself with the software using the provided MiniZinc guide, which is located in the same folder as this document.

Lab questions. There are three questions. For each problem, you are given a skeleton model (.mzn file) that contains an incomplete model of the problem. Your task is to fill in the missing pieces to complete the model and find solutions to ten instances of the problem (.dzn files). Please leave the parameters, decision variables, objective function and model output definitions untouched. Furthermore, please write a single constraint for each sub question.

Tip #0. Read the problem text, and ensure the problem is crystal clear! This is essential before you start formally modelling the problem.

Tip #1. We recommend adding one constraint at a time and checking the changes in the solutions after adding a new constraint to the model. This will help you the understand how the model is built to reach the desired solutions and detect bugs early. To support you, we prepared a special piece of software ("checker"), for usage see the readme file in the 'checkers' folder.

Tip #2. By default, solvers in MiniZinc are configured to run without a timeout. However, for this lab you should set a timeout of 1 minute. You can do so by clicking "Show configuration editor", and under "Options" (make sure "Maintain these options across solver configurations" is checked) check the "Time limit" checkbox and set the time limit to 60 seconds. Ensure that, within the time limit, the instances for the first two questions are solved to optimality, and at least one feasible solution is found for the instances of the last question. Depending on your hardware, you may also consider increasing the time limit, e.g., to two minutes.

Tip #3. Recall that different solvers may exhibit different performance depending on the problem being solver. We invite you to experiment with the solver Chuffed, COIN-BC and Gecode. Note that at this stage it is not necessarily need to understand how these solvers work, it is enough to know that they implement different algorithms, and they can be used with MiniZinc.

Submission. Please submit a single ZIP archive containing your completed .mzn files. Your submission will be evaluated pass or fail, where a pass is awarded if the checker accepts your model. This allows you to assess your own submission before submitting! Currently the submission instructions are incomplete, we will update this section shortly. In the meantime you may still use the checker to evaluate your approach.

1. Weighted Set Cover

The set cover problem is a classical combinatorial optimisation problem, where the goal is cover a number of elements by selecting the least amount of sets. See the Wikipedia page for the definition[1].

A concrete example would to form a team of people, such that the team has a certain set of skills. Let us say a team needs to have the following skills: *Planning, Written communication, Programming, System Design, Presenting*. We can choose from the following pool of people:

Person	Skills
Anika	Planning, Written communication, Programming
Bao	Written communication, System Design
Cees	Programming, System Design
Dave	System Design, Presenting

In this situation, the smallest team to cover all skills would consist of Anika and Dave. This example is encoded in the small-1.dzn file.

a. **set-cover-explicit.mzn** models the above example. Using this file and **small-1.dzn**, write a constraint for each skill to satisfy the skill requirements.

   Using the model file **set-cover.mzn,**

b. Write a generic constraint for team skill requirements.
c. The set cover problem can be expanded to the weighted set cover problem, by attaching a cost to each of the people (their salary for example). The goal is to minimise the combined cost of the team. Write a constraint where the *total_cost* is the sum of the salaries of the selected people.

[1] https://en.wikipedia.org/wiki/Set_cover_problem

2. Graph Coloring

Consider the famous NP-complete graph coloring problem, where the question is whether the nodes of a graph can be colored with exactly k colors such that no two connected nodes share the same color. See the Wikipedia page[2] for more information.

Graph coloring has numerous applications. One such example is the timetabling problem, where classes are represented by the nodes, and two nodes share an edge if there is at least one student who enrolled in both corresponding classes. In this case, feasible graph coloring provides a timetable with no student conflict.

Using the model file **graph-coloring.mzn,** write a constraint that

a. A node can take only one color,
b. Neighboring nodes cannot have the same color,
c. k is the number of used colors.

[2] https://en.wikipedia.org/wiki/Graph_coloring#Applications

3. Sports Scheduling

The sports scheduling problem is a classical constraint programming problem, where the goal is to determine the order and location of games between teams. Each team plays one game per round, either at home or away, and the schedule is complete when every team has played against every other team exactly once. Team i incurs traveling cost $cost[i, j]$ when it plays an away game with team j. Furthermore, a team cannot have more than two consecutive home games or more than three consecutive away games.

Using the model file **sports_scheduling.mzn,** write a constraint that

a. A team never plays itself,
b. The opponent of a team's opponent is the team itself,
c. If a team plays home, its opponent plays away,
d. All teams play someone else within the same round,
e. All teams play each other once,
f. No team has more than two consecutive home games or more than three consecutive away games.
g. Objective is the total traveling cost incurred by the teams.