FIT5216: Modelling Discrete Optimization Problems
Assignment 3: Crew Allocation

1 Overview

For this assignment, your task is to write a MiniZinc model for a given problem specification, and a report on your findings.

- Submit your model to the MiniZinc auto grading system (using the submit button in the MiniZinc IDE). You must submit via the IDE to be graded and receive marks.
- Submit your model (copy and paste the contents of the .mzn file) using the Moodle assignment.
- Submit your report using the Moodle assignment.

You have to submit by the due date (Tuesday 19th May 2020, 11:59pm), using MiniZinc and using the Moodle assignment, to receive full marks. You can submit as often as you want before the due date. Late submissions without special consideration receive a penalty of 10% per day. Submissions are not accepted more than 3 days after the original deadline.

This is an **individual assignment**. Your submission has to be **entirely your own work**. We will use similarity detection software to detect any attempt at collusion, and the **penalties are quite harsh**. If in doubt, contact your teaching team with any questions!

2 Problem Statement

Your task is to produce a staff roster, allocating staff to vehicles. For example, the scenario could be an airline allocating crew to their aircraft, or a construction company putting together teams of builders. Each staff member has a certain set of skills, with varying competence. For example, a captain would have very high competence in flying the plane, a co-pilot high competence, and all other staff no competence. For each week and each vehicle, a certain combination of skills at certain competence levels is required.

The staff members, skills and vehicles are given as enumerated types, and the number of weeks for the schedule is given as a set:

```
enum STAFF;
enum VEHICLE;
enum SKILL;
int: n_weeks;
set of int: WEEKS = 1..n_weeks;
```

The staff members competence levels and weekly salaries, and the required combined competence for each week, vehicle and skill, are given as arrays:

```
array[STAFF,SKILL] of int: competence;
array[STAFF] of int: salary;
array[WEEKS,VEHICLE,SKILL] of int: required_competence;
```

2.1 Basic Model

Write a model that assigns a crew to each vehicle and week, such that

- For each skill, the combined competence of the crew in that skill is at least as much as is required. Competence simply adds up (i.e. two people with competence 5 for a skill is the same as one with competence 10).
- Each staff member can be allocated to at most one vehicle per week.
- The overall cost of the schedule is minimised. The cost is simply the sum over the salaries of each allocated staff member, per vehicle and week.

Document your model with comments that briefly explain your modelling decisions.
Use the COIN-BC solver for testing your model. The auto-grading server is also configured to use COIN-BC for grading this assignment.

2.2 Crew Rotation and Rest

In a security-oriented industry, it is important to rotate and rest staff members. E.g. pilots and co-pilots will only work together for a short time, so that they are forced to obey proper protocols (rather than develop any shortcuts together). The number of weeks after which two staff members have to switch teams is given as an integer `max_together`. In addition, staff members need to take a week off (i.e., cannot be allocated to any vehicle) after `max_consecutive` weeks.

If `max_consecutive=0` or `max_together=0`, it means that there is *no* constraint, i.e., no restriction on crew rotation or rest weeks.

Extend your model so that any two staff members are only working together on the same vehicle at most `max_together` weeks in a row, and only work for at most `max_consecutive` weeks in a row.

Explain in the report how you implement the rotation and rest constraints (e.g. which viewpoint you chose, and whether you experimented with different approaches).

2.3 Secondary Objective

In addition to minimising cost, we want to also maximise the competence levels of our crews. Extend your model such that it will produce a solution that is minimal in cost, and for the minimal cost, also maximises the overall competence of the staff employed in each week.

For the submission, implement the balance between the two objectives by using a weight of 10 for the cost, and a weight of -1 for the overall competence.

For the report, consider the following questions:

- Write a short explanation about how you implement these two objectives.
- Experiment with different weightings for the two objectives. For example, how different are the solutions if you put most weight on maximising competence and little weight on minimising cost? Represent the trade-offs between competence and cost in a graphical way (for example, plot the maximum competence for each possible cost, or the other way around). You should use data files `c4.dzn` and `c5.dzn` for these experiments.

2.4 Wrap Around

In your model so far, you only produce a schedule for the given set of weeks, without taking into account what happened before the schedule starts, or what the allocation will look like in the week after the schedule ends.

Your task now is to modify the model so that it produces a *cyclic* schedule. For example, if the given number of weeks is 4, then in week 5 we will use the same allocation as in week 1, then continue again with week 2 and so on. Assuming that the required skills are the same every 4 weeks, we can therefore just produce a single schedule and then "copy-and-paste" it for the entire year.

But in order to do that, all constraints still have to hold for the whole year. In particular, if two staff members were working together in week 4, they may not be allowed to work together in week 1 (because week 1 is now also used as week 5). And the same holds for the week off after `max_consecutive` weeks.

Modify your model so that it produces correct cyclic schedules. Explain in the report how you modelled the cyclic aspects, and how the modification affects the cost of the produced schedules for different data files.

3 Instructions

Edit the provided `mzn` model files to solve the problems described above. You are provided with some sample data files to try your model on. Your implementations can be tested locally by using the *Run* icon in the MiniZinc IDE.

The report has to be submitted in PDF format. It does not have to follow any particular structure (e.g. you do not need an abstract, table of contents etc.). The explanations should be concise. The overall limit for the report is 3 pages, including all graphs, tables etc.

4 Marking

The auto-grading server is using the COIN-BC solver to grade this assignment. Grades are calculated based on correctness of the found solutions, not on the quality of solution that is achieved. However, your model needs to be able to find at least one solution for each instance within the 1 minute time-out in order to get any marks.

You can get a maximum of 30 marks for this assignment, which will be worth 15% of your overall unit marks.

The marks for the correctness of your model are automatically calculated. You will receive up to 5 marks for locally tested data and up to 10 marks for your model as tested on the auto-grading server. You will only get full marks if you implement all extensions.

The code comments and your report will be marked by your tutors, and they are worth the remaining 15 marks, with the following allocation:

Part 2.1 (code comments): 3 marks; Part 2.2: 3 marks; Part 2.3: 6 marks; Part 2.4: 3 marks.