FIT5216: Modelling Discrete Optimization Problems
Assignment 2 (extended): Testing Vaccines

1 Overview

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

- Submit your work 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.

You have to submit by the due date (1st May 2022, 11:55pm), 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

In the race to find an effective vaccine for COVID-19 for the many new strains developing the World Health Organization is a trialing different vaccines combinations.[1]. The vaccines are given as an enumerated type:

```
enum VACCINE;
```

To determine the effectiveness of vaccines and their combinations we need to test them on different subsets of the population, to determine efficacy and any other details. Its important to also test placebos, to see what effect is just from having any treatment. So we are guaranteed to a value in the set representing the placebo vaccine (an injection of sterile water)

```
VACCINE: placebo;
```

The placebo vaccine is guaranteed to occur before any real vaccine in the type:

```
constraint assert(forall(v in VACCINE)(placebo <= v),
                  "vaccine \(v) appears before placebo\n"));
```

The number of test populations which will undergo different vaccine combinations is defined by N.

```
int: N; % number of test populations
set of int: POP = 1..N;
```

[1]This is a **fictional scenario**, and completely made up! But this type of problem, called an *experiment design*, is quite common in practice.

The testing will happen over a W week trial.

```
int: W; % number of weeks
set of int: WEEK = 1..W;
```

We have to determine for each population a treatment plan, indicating what activity is planned for each week. The possibly treatments for every member in the population are: WAIT do nothing; VAX give a vaccination shot; PCR give a PCR test; RAT give a RAT test; and SAT give a SAT test. Each treatment has an associated cost. This is defined in MiniZinc as

```
enum TREATMENT = { WAIT, VAX, PCR, RAT, SAT };
array[TREATMENT] of int: cost;      % cost of a treatment to the population
```

The key decisions are for each population, what is the treatment plan and what are the set of vaccinations they get:

```
array[POP,WEEK] of var TREATMENT: schedule;
array[POP] of var set of VACCINE: vaccinations;
```

The constraints on the assignment are as follows:

1. Each vaccine is given to a group either 0 or 2 times. Hence the number of scheduled VAX treatments for each group should be 2 times the number of vaccines they are vaccinated with.
2. Each population gets a different set of vaccinations
3. One population gets no vaccines
4. One population gets only the placebo vaccine.
5. In each week at most $maxvax$ populations can get vaccinated

   ```
   int: maxvax; % maximum populations getting vaccinated each week
   ```

6. In each treatment plan we have to abide by minimum and maximum limits of each treatment: we must give every population the minimum of each treatment, and no more than the maximum.

   ```
   array[TREATMENT] of int: mintreat;  % minimum number for each treatment
   array[TREATMENT] of int: maxtreat;  % maximum number for each treatment
   ```

7. Every treatment plan must have a WAIT treatment the week after each VAX treatment
8. No population can have a PCR test if they havent had a RAT test earlier in their treatment plan
9. The first treatment cant be WAIT for any population, otherwise the population doesnt think the trial has started.

3 Stage A: Finding a Plan

For the first part, the aim is simply to find a treatment plan that works.

For example a sample data set is

```
VACCINE = { P, V1, V2 };
placebo = P;
N = 4;
W = 6;
cost = [ 0, 10, 4, 2, 6 ];
maxvax = 2;
maxtreat = [ W, 4, 3, 2, 1 ];
mintreat = [ 0, 0, 1, 1, 0 ];
```

which defines a problem with two actual vaccines and one placebo. There are 4 different populations in the treatment plan which goes for 6 weeks. The maximum number of populations that can be vaccinated in a week is 2. Each populations can have any number of rests, at least 1 PCR and RAT test, and maximum of 4 vaccinations, 3 PCR, 2 RAT and 1 SAT.

A solution is given by

```
cost = 90;
schedule = [| VAX, WAIT,  RAT,  PCR,  SAT,  VAX
 | RAT, WAIT, WAIT,  PCR, WAIT, WAIT
 | RAT, WAIT,  VAX, WAIT,  PCR,  VAX
 | RAT,  VAX, WAIT,  PCR,  VAX, WAIT
 |];
vaccinations = [{P}, {}, {V1}, {V2}];
```

Notice how each population has either 2 or 0 VAX scheduled consistent with the number of vaccinations. And all of the other constraints hold.

A visualization of the solution is given below

```
       000000
       123456
pop  1:V.RPSV  32 {P}
pop  2:R..P..   6 {}
pop  3:R.V.PV  26 {V1}
pop  4:RV.PV.  26 {V2}
```

Where we see what each population does in each week, the total cost of treatments for each population and the set of vaccinations they receive.

Write a MiniZinc model `strains.mzn` which models the above problem.

4 Stage B: Minimal Expenditure

If you only finish the first stage you will be unable to acheive more than 50% marks for most instances.

In reality the aim is to minimize the overall cost of the treatment plan

Modify your model to to minimize the total treatment cost of the plan.

5 Stage C: Balanced Expenditure

To achieve full marks on mmnay instances you need to tackle this extension. Grading of the solutions will be done based on this "true form" of the objective. Note it does make solving much harder.

In reality we need to minimize the total cost of the treatment plan. But different population groups get disheartened if the expenditure on them is substantially different.

Add a constraint requiring that the cost of the treatment of each population group is no more than 2 times the cost of treatment of another population group. Clearly the solution above violates this constraint since population 1 has cost 32 which is more than twice the cost for population 2 of 6.

An example solution (not necessarily the best) with this constraint is

```
cost = 92;
schedule = [| VAX, WAIT, RAT,  PCR, WAIT,  VAX
 | RAT, WAIT, PCR,  VAX, WAIT,  VAX
 | SAT,  RAT, PCR, WAIT, WAIT,  RAT
 | RAT,  PCR, VAX, WAIT,  VAX, WAIT
 |];
vaccinations = [{P}, {V1}, {}, {V2}];
```

with visualization

```
       000000
       123456
pop  1:V.RP.V  26 {P}
pop  2:R.PV.V  26 {V1}
pop  3:SRP..R  14 {}
pop  4:RPV.V.  26 {V2}
```

Clearly the worst off population has 14 spend on it which is more that half of the spend on the other populations 16.

6 Stage D: Testing Effectiveness

To achieve full marks you need to tackle this extension. There will be a limited number of tests of stage D

In the current plan there is no guaranteee that the treatment plans are sufficiently rigourous. The vaccines have to handle new strains of the vaccine. The strains we are interested in are ALPHA. BETA, OMICRON, and MU. Each test has differing accuracy on each strain: false negative rate F^- (this is the chance that the test says someone does not have the strain, but they actually do) and false positive rate F^+ (this is the chance that the test says someone does have the strain but they actually dont.

Below is a table showing the false negative and false positive rates for each test and strain.

Test	ALPHA F^-	BETA F^-	OMICRON F^-	MU F^-
PCR	2%	3%	5%	30%
RAT	10%	12%	20%	16%
SAT	3%	4%	8%	10%

If we do different tests then the chances of an overall false negative are multiplied, making it increasingly unlikely. So for example if the schedule has a RAT followed by PCR followed by SAT, the chance of a false negative for MU is $16\% \times 30\% \times 10\% = 0.0048 = 0.48\%$.

Add constraints to your model to ensure that each treatment plan has no more than a 1% chance of a false negative overall for each strain at the end of the plan.

Hint: Multiplication is hard for solvers (and in fact COIN-BC will disallow most multiplication constraints), but there is a way to rewrite this multiplication constraint to only use addition!

The stageD constraint only applies for some of the instances. If your model includes

```
opt bool: stageD;
```

then if this is not given in the data files (the files for stages A to C), the constraints above do not need to be applied. Stage D instances will include

```
stageD = true;
```

You can use a constraint like

```
constraint if occurs(stageD) then ... endif;
```

to ensure that constraints (in the `...`) are only applied for stage D examples.

7 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 or by using,

```
minizinc ./modelname.mzn ./datafile.dzn
```

at the command line. Note the grader will make use of the solver COIN-BC, so we definitely suggest you use this solver when developing and testing your solution.

8 Marking

The marks are automatically calculated.

The submission has 14 marks for locally tested data and 18 for model testing, for a total of 32 marks.