FIT5216: Modelling Discrete Optimization Problems
Assignment 2: Nurse Rostering

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).
- Submit your model (copy and paste the contents of the .mzn file) and report using the Moodle assignment.

You have to submit by the due date (26th April 2024, 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% of the available marks per day. Submissions are not accepted more than 7 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**. You may not use **large langauge models** for any part of this assignment. If in doubt, contact your teaching team with any questions!

2 Problem Statement

Your task is manage the roster a busy hospital

There are 5 kinds of shifts in the roster:

MORN the person works in the morning shift

DAY the person works on the day shift

EVEN the person works on the evening shift

NIGH the person works on the night shift

OFF the person has the day off

Your aim is to build a roster over a give number of days for a set of nurses, that satifies various constraints, about shift sequence, that has enough people available for each shift type. You must also assign nurses to wards, and satisfy constraints about wards assigned.

Input data is given in MiniZinc data format:

```
NURSE = ⟨ The set of people to roster ⟩;
nday = ⟨ The number of days to roster for ⟩;
rostered_off = ⟨ For each nurse and day have they been already granted a day off ⟩;
maxweek = ⟨ The maximum work shifts allowed in any 7 day period ⟩;
```

```
maxnightfort = ⟨ The maximum number of NIGH shifts allowed in any 14 day period ⟩;
minfort = ⟨ The minimum work shifts allowed in any 14 day period ⟩;
minshift = ⟨ For each shift and day the minimum nurses rostered to each shift ⟩;
shift_cost = ⟨ The cost for rostering on each nurse for one day ⟩;
WARD = ⟨ The set of wards to assign ⟩;
dummy = ⟨ The ward representing a dummy (no ward) assignment ⟩;
minward = ⟨ For each ward and day the minimum nurses rostered to each ward ⟩;
maxward = ⟨ The maximum wards any nurse can staff in the roster period ⟩;
SKILL = ⟨ The set of advanced skills nurses may have ⟩;
skill = ⟨ For each nurse the set of advanced skills they have ⟩;
desired = ⟨ For each ward the advanced skills they want ⟩;
emergency = ⟨ Which ward is the emergency ward if there is one ⟩;
```

Note that the `emergency` data can be omitted meaning there is no emergency department in the roster problem.

Here is a sample data set (given in `nroster00.dzn`):

```
NURSE = { A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P };
nday = 14;
rostered_off = [| true, false, false, false, false, false, false,
                  true, false, false, false, false, false, false
                | .. too big to show .. |];
maxweek = 5;
minfort = 8;
maxnightfort = 4;
minshift =  [| 1, 3, 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 2, 1
             | 2, 2, 6, 2, 4, 3, 6, 2, 6, 2, 2, 4, 3, 2
             | 1, 5, 3, 2, 1, 2, 1, 3, 1, 1, 2, 1, 2, 1
             | 2, 2, 2, 2, 6, 2, 2, 4, 2, 2, 5, 4, 4, 2
             | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             |];
shift_cost = [ 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 10, 10 ];
WARD = { GEN, EME, HRT, CAN, '.' };
dummy = '.';
minward =   [| 2, 5, 2, 5, 0, 0, 2, 2, 5, 2, 5, 0, 0, 2
             | 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 2, 2, 4, 4
             | 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1
             | 1, 1, 5, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1
             | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             |];
maxward = 2;
SKILL = { SENIOR,  EMERG, CARDIO, RADIO };
senior = SENIOR;
skill = [ {}, {}, {}, {}, {}, {}, {EMERG}, {EMERG}, {CARDIO}, {RADIO}, {SENIOR},
          {SENIOR}, {EMERG,CARDIO}, {EMERG,RADIO}, {EMERG, SENIOR}, {CARDIO, SENIOR} ];
```

```
desired = [ {}, { EMERG }, { CARDIO }, {RADIO}, {} ];
emergency = EME;
```

This data requires building a roster for 16 employees over 14 days.

The decisions of the roster are

```
array[NURSE,DAYS] of var SHIFT: sh; % shift for each nurse on each day
array[NURSE,DAYS] of var WARD: wd; % ward rostered for each nurse
var int: total_cost; % the total cost of the roster
```

The assignment is in stages, please attempt them in order. Note that if you do not complete all stages you cannot get full marks.

Stage A - Basic Rostering

Create a model `nroster.mzn` that takes data in the format specified above and decides on shifts for each person. For Stages A – C you can just set the `wd` to `dummy` for all nurses. For Stages A – B you can just set `total_cost` to 0.

For stage A we just need to ensure we make acceptable roster schedules for each nurse.

A1 First we need to ensure that every nurse is rostered OFF for each day where `rostered_off` is true.

A2 Next we need to ensure that the model decides on shifts for each person that satisfy the shift regulation constraints, which are:

- No nurse works more than 3 NIGH shifts in a row
- No nurse works a MORN shift directly after a NIGH shift
- No nurse works a DAY shift directly after a NIGH shift
- No nurse works a MORN shift directly after an EVEN shift
- No nurse has more than 2 OFF shifts in a row
- No nurse that works a DAY shift has less than 2 DAY shifts in a row (except the last shift can be a lone DAY shift)

A possible `sh` solution for the data above to stage A is:

```
A: . E E E E E N . E E N E E D
B: M . E E E E E . . E E E E D
C: D D . E E E . M M . E E E D
D: E N N . E E E E D D . E E D
E: E E E . . E E E E E . . E D
F: E E E E N . E E E E . M . M
G: E E E E . M . E E E E . M .
H: . E E E N . M . E E E D D .
```

```
I: M . E E E E . . N . E E D
J: E N . N . E E E E . E E E D
K: E E N . E E E E E N . . E D
L: E E E . . D D . E N N . E D
M: . E E E N . E E E E E E . M
N: E E E E . M . N . E E N N .
O: . E E N E E E E E E E N N .
P: M . E E E E E E E E E N . .
```

Where M = MORN, D = DAY, E = EVEN, N = NIGH and . = OFF. Note how the days off for nurse A align with where they are rostered off (days 1 and 8). No nurse roster violates any of the constraints, e.g. we dont see three days OFF in a row, or less than two DAY shifts in a row, except on the last day.

Note there are global constraints that can create very efficient models for this stage.

Stage B - Minimum Rostering Levels

Clearly the schedule above has many EVEN shifts. This is because without making sure each shift has enough people on it we get silly solutions.

B1 In this stage we need to ensure that each SHIFT type s for each day d has at least `minshift[`s`,`d`]` nurses rostered to it.

B2 We also need to enforce long term constraints on the shift sequences of nurses:

- No nurse works more than `maxweek` shifts (shifts other than OFF) in any 7 day period.
- No nurse works less than `minfort` shifts in any 14 day period.
- No nurse works more than `maxnightfort` NIGH shifts in any 14 day period.

Use global constraints where possible to encode these constraints.

A possible `sh` solution for the data above to stage A is:

```
A: . D D . D D D . E . M D D M : 10
B: . . E E N N . E . N N . E E : 9
C: E E . M M . D D D . D D . N : 10
D: M M N . N . D D D . . D D M : 10
E: N E D D . M . E D D N . M . : 10
F: M N . N N . E . D D D M . D : 10
G: D D D . N N . D D . E D D . : 10
H: . M . M E D D . N . N N N . : 9
I: M . M . D D D N . M . N E N : 10
J: M M . N . E D D M . N . N E : 10
K: N N E . M M . M D D . . M M : 10
L: D D D E . . N . M E E . M D : 10
M: . E N . N . M N E . E N . . : 8
```

```
   N: . E D D D . . N . M M M N . : 9
   O: . E E . N E . E N E . N N . : 9
   P: . . D D D . N N . N N E . . : 8
MORN: 4 3 1 2 2 2 1 1 2 2 2 2 3 3
DAY:  2 3 6 3 4 3 6 4 6 3 2 4 3 2
EVEN: 1 5 3 2 1 2 1 3 2 2 3 1 2 2
NIGH: 2 2 2 2 6 2 2 4 2 2 5 4 4 2
OFF:  7 3 4 7 3 7 6 4 4 7 4 5 4 7
```

Where now we show the number of working shifts in the 14 day period for each nurse on the right. Note how in each 7 day period no nurse works more than 5 shifts, and no one works less than 8 shifts in the full 14 day roster. No nurse works more than 4 NIGH shifts over the 14 day roster.

The number of nurses on each shift for each day is shown below the roster. Note how each reaches the min required by `minshift`.

Stage C - Optimising Cost

Now we need to calculate the total shift cost, and minimize it. For every shift when a nurse is rostered on (not OFF) we need to pay the shift cost for that nurse given by `shift_cost`. The total cost for the roster above is 484. But we can do better. Compute the total cost of the roster in variable `total_cost` and minimize it.

Here is an optimal solution for Stages A – C. Note that the total cost is substantially reduced

```
   A: . M D D D . M . D D . N N N : 10
   B: N . N . E D D E . . E D D M : 10
   C: E N . E N N . E N . E D D . : 10
   D: N N E . N E . M N E . D D . : 10
   E: . D D . . E D D D . M . M D : 9
   F: D D D . N . D D D . N E . . : 9
   G: . E D D N . . N E . . M N . : 8
   H: . M D D D N . . M . D D . . : 8
   I: . . M . D D D N . N . M E . : 8
   J: . E . N . . D D D . N . E D : 8
   K: . E E . N . E E . N . . M N : 8
   L: . E N E . M N . M . N . . E : 8
   M: . M . M M . N . D D D N . . : 8
   N: D D D . M M . N . M . . N . : 8
   O: . E . . D D D N . . N N N . : 8
   P: M . E N N . . D D . N N . . : 8
MORN: 1 3 1 1 2 2 1 1 2 1 1 2 2 1
DAY:  2 3 6 3 4 3 6 4 6 2 2 4 3 2
EVEN: 1 5 3 2 1 2 1 3 1 1 2 1 2 1
NIGH: 2 2 2 2 6 2 2 4 2 2 5 4 4 2
OFF: 10 3 4 8 3 7 6 4 5 10 6 5 5 10
total_cost = 436;
```

Stage D - Ward Constraints

Each nurse that is rostered on has to be assigned to a WARD where their principal responsibilities lie. We have minimum staffing requirements for each ward on each day. Now you must decide the `wd` assignment of each nurse on each day to each ward. First for consistency we require that any nurse that is rostered OFF should be assigned to the `dummy` ward and vice versa. Note these constraints should always be turned on in stage F.

You model should satisfy the following ward constraints:

D1 For each ward w and each day d at least `minward[`w`,`d`]` nurses should be assigned to that ward.

D2 There is one more restriction on ward assignments: No nurse can be assigned to more than `maxward` different wards (including the dummy ward) over the roster period.

Again use global constraints where possible to encode the constraints.

A solution that satisfies these constraints is:

```
   A: . E D D D . N . M M . M D D : 10
   B: N . . N E D D N . . D D M M : 10
   C: N N . M M M . M M . N N . M : 10
   D: D D D . N N . E D D . . N E : 10
   E: D D D E . . M N E . E . N N : 10
   F: . E E . N . D D D . N E . N : 9
   G: . M D D D N . . D D N N . . : 9
   H: . N . . N E N . N . D D M . : 8
   I: . . N . M M . E . N . D D M : 8
   J: M E . N . . D D D . . D D . : 8
   K: . M M . D D D . N N . . E . : 8
   L: . . N E . D D . D D N . N . : 8
   M: . E D D N . . N . E . N . D : 8
   N: . M D D D . . N . . M M N . : 8
   O: . E E . N . D D D . E . E . : 8
   P: E . E . N E E E . . N N . . : 8
MORN: 1 3 1 1 2 2 1 1 2 1 1 2 2 3
 DAY: 2 2 6 4 4 3 6 3 6 3 2 4 3 2
EVEN: 1 5 3 2 1 2 1 3 1 1 2 1 2 1
NIGH: 2 2 2 2 6 2 2 4 2 2 5 4 4 2
 OFF: 10 4 4 7 3 7 6 5 5 9 6 5 5 8
total_cost = 440;
  A:   . EME EME EME EME   . EME   . EME EME   . EME EME EME
  B: GEN   .   . GEN EME EME EME GEN   .   . GEN EME GEN GEN
  C: EME GEN   . GEN EME EME   . GEN GEN   . GEN EME   . EME
  D: EME GEN GEN   . GEN EME   . GEN GEN GEN   .   . GEN EME
  E: GEN GEN GEN GEN   .   . GEN EME GEN   . GEN   . GEN GEN
  F:   . GEN CAN   . GEN   . GEN GEN GEN   . GEN GEN   . CAN
  G:   . GEN GEN GEN GEN CAN   .   . GEN CAN GEN CAN   .   .
```

```
  H:   . GEN   .   . EME EME EME   . GEN   . EME EME EME   .
  I:   .   . EME   . EME EME   . EME   . EME   . EME EME EME
  J: CAN GEN   . GEN   .   . GEN GEN GEN   .   . GEN GEN   .
  K:   . GEN GEN   . GEN GEN GEN   . GEN GEN   .   . GEN   .
  L:   .   . CAN HRT   . CAN CAN   . CAN HRT HRT   . HRT   .
  M:   . HRT CAN CAN CAN   .   . HRT   . HRT   . CAN   . HRT
  N:   . EME CAN EME CAN   .   . EME   .   . EME EME EME   .
  O:   . CAN CAN   . EME   . EME CAN EME   . CAN   . CAN   .
  P: HRT   . HRT   . HRT HRT HRT HRT   .   . HRT HRT   .   .
GEN:   2   8   4   5   4   1   4   5   8   2   5   2   5   2
EME:   2   2   2   2   6   5   4   3   2   2   2   6   4   4
HRT:   1   1   1   1   1   1   1   2   0   2   2   1   1   1
CAN:   1   1   5   1   2   2   1   1   1   1   1   2   1   1
  .:  10   4   4   7   3   7   6   5   5   9   6   5   5   8
```

Here we can see the ward allocations for each nurse and each day. Note how the dummy ward ' .' lines up with days OFF exactly. The minimum ward allocations for each ward and day are satisfied. No nurse is assigned to more than two wards over the period.

Stage E - Skills + Slices [Challenge Stage]

Stage E adds the concepts of SKILLs and SLICEs to the roster. It is designed to be challenging, you can get most of the marks without handling this stage. The enumerated type `SKILL` stores the set of skills. Each nurse can have a set of additional skills, which may be empty. One of these skills is `senior` which means the nurse has broad experience over a number of years. Each ward has a set of `required` skills. There are 4 slices in a day PREDAWN, AM, PM, LATE which account for the hours 0-6, 6-12, 12-18, 18-24 of the day. Each shift covers 2 slices: MORN covers { PREDAWN, AM }, DAY covers { AM, PM }, EVEN covers { PM, LATE } and NIGH covers LATE and PREDAWN of the next day.

The stage E constraints are:

- On any day when a ward requires rostered on nurses, for each of its required skills at least one nurse assigned to the ward has that skill.
- For each slice there must be a nurse with skill `senior` rostered to cover that slice, somewhere in the hospital.
- If there is an emergency ward defined, then some nurse must be rostered to emergency that covers each transition from one slice to the next, i.e. there must a single nurse covering both PREDAWN and AM in emergency, and (different) single nurse covering both AM and PM, also PM and LATE, and LATE and PREDAWN of the next day. This is to ensure smooth handoff of patients as the personnel in emergency change over the day. Note you can check whether an emergency ward is defined using MiniZinc like

  ```
  if occurs(emergency) then ... endif
  ```

A solution satisfying the stage E constraints as well is

```
   A: . M D D D N . . D D N N N . : 10
   B: N . D D D M . N . . E D D E : 10
   C: . E . E D D D . M . N E N N : 10
   D: M N E . N E . E D D . M M . : 10
   E: M E E N . . D D D . N . N . : 9
   F: D D D M N . . N N . D D . M : 10
   G: D D . M M M . M M . E N E . : 10
   H: . M D D N . N . D D . . E . : 8
   I: . . D D D . M E . N . M M N : 9
   J: N E . E . D D N N . N . . D : 9
   K: . M M . M . D D D E . . N N : 9
   L: E E N . . N N . E . M . D D : 9
   M: . E E . N . E . D D D N . M : 9
   N: . N N . N E . E . N . D D . : 8
   O: . . D D N . D D . M N N . . : 8
   P: M . . N E D D N . . D D . . : 8
MORN: 3 3 1 2 2 2 1 1 2 1 1 2 2 2
 DAY: 2 2 6 5 4 3 6 3 6 4 3 4 3 2
EVEN: 1 5 3 2 1 2 1 3 1 1 2 1 2 1
NIGH: 2 2 2 2 6 2 2 4 2 2 5 4 4 3
 OFF: 8 4 4 5 3 7 6 5 5 8 5 5 5 8
total_cost = 458;
  A:   . GEN CAN GEN GEN GEN   .   . GEN GEN GEN GEN GEN   .
  B: EME   . EME GEN GEN GEN   . EME   .   . GEN EME EME EME
  C:   . GEN   . EME GEN GEN EME   . GEN   . GEN EME EME GEN
  D: EME EME CAN   . EME EME   . EME EME EME   . CAN EME   .
  E: GEN GEN CAN GEN   .   . GEN GEN GEN   . GEN   . GEN   .
  F: GEN GEN GEN GEN GEN   .   . EME EME   . GEN GEN   . GEN
  G: EME EME   . EME EME EME   . EME EME   . EME EME EME   .
  H:   . EME GEN GEN GEN   . GEN   . GEN GEN   .   . GEN   .
  I:   .   . HRT HRT EME   . EME HRT   . EME   . EME HRT HRT
  J: CAN CAN   . CAN   . CAN CAN CAN CAN   . CAN   .   . CAN
  K:   . GEN EME   . EME   . GEN EME GEN EME   .   . EME EME
  L: EME EME EME   .   . EME EME   . EME   . EME   . EME EME
  M:   . HRT EME   . HRT   . EME   . HRT HRT HRT EME   . EME
  N:   . GEN CAN   . CAN CAN   . GEN   . CAN   . CAN CAN   .
  O:   .   . CAN EME EME   . CAN CAN   . EME EME EME   .   .
  P: HRT   .   . EME EME EME HRT EME   .   . EME EME   .   .
GEN:   2   6   2   5   5   3   3   2   5   2   5   2   3   2
EME:   4   4   4   4   6   4   4   6   4   4   4   7   6   4
HRT:   1   1   1   1   1   0   1   1   1   1   1   0   1   1
CAN:   1   1   5   1   1   2   2   2   1   1   1   2   1   1
  .:   8   4   4   5   3   7   6   5   5   8   5   5   5   8
```

Note that the example data `nroster00.dzn` is quite challenging in order to show all the complexities of the model. You should start testing on the easier data files `nroster01.dzn` ..

nroster05.dzn.

Stage F - Sensitivity Report

Each of the constraints of the problem serves to make it more difficult, but for different data files the constraints that make it hard to find solutions may differ. Using the solver `HiGHS 1.6.0` for each data file (except `nroster00.dzn`) in the `data` subdirectory make an exploration about how much each group of constraints affects the runtime of the model and how much it affects the objective. We consider 7 groups of constraints A1, A2, B1, B2, D1, D2 and E and hence 7 variants of the problem with one group turned off (removed from the model). Run your model with the objective function and all constraints except one group that is removed. Compare that with running the full model. Note that if you have not successfully implemented all stages explore it for each stage that you have implemented (e.g. if you only implemented to stage C you have 4 variants with all constraints except A1, A2, B1 and B2). Give a table of the best objective value found within a reasonable time limit (60 seconds say) for each variant. Give a table of runtimes (or timeout) to prove optimality within a reasonable time limit (60 seconds say) for each variant.

In the report, for each data file rank each set of constraints in order of importance, i.e. which constraints contribute most to raising the objective, in decreasing order. Explain why you think this is the case, from examining the data in that file, with a paragraph per data file. You may want to examine the solver statistics to help understand what is happening. Finally, in the last paragraph explain what are the most important constraints of the model overall, and justify your answer. The report should use no more than 4 A4 pages.

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+check* icon in the MiniZinc IDE. Note that the checker for this assignment will only test whether your model produces output in the required format, it does not check whether your solutions are correct. The grader on the server will give you feedback on the correctness of your submitted solutions and models. Global constraints such as `global_cardinality`, `nvalue`, `regular`, and `sliding_sum` can be helpful for this assignment, but are certainly not required.

4 Marking

You will get a maximum of 30 marks for this assigment which is worth 15% of the units overall marks. Part of the marks are automatically calculated. The submission has 8 marks for locally tested data and 8 for model testing, for a total of 16 marks by automatic calculation. You will only get full marks if you implement all stages.

The 14 remaining marks are given for code comments and the report marked by the tutors, with the following allocation: Code Comments (4 marks) Report (10 marks).

For the autograded part of the marking you can get most marks having implemented Stages A-D only. You will not get any marks unless you implement at least up to Stage C, and the consistency between OFF and dummy ward in Stage D. Stage E is optional, without implementing it there will be some instances where you can get a maximum of 3/4 of the marks available.

Code commenting should clear explain the role of each variable, constraint and objective defined in the model. You should explain how every constraint is modelled unless this is straightforward: e.g. adding that we use `alldifferent` to ensure all of a set are different is unecessary detail. You should use good identifier names for variables and procedures you define to help readability. The code and the comments must be formatted to be easy to read and comprehend.

The report requires two tables and a written discussion: 1 mark each is available for the data presentation in the two tables, which should be clear and precise. Make sure its easy for the reader to determine what everything in the table means. The written report is worth 8 marks and should answer the questions listed in part F. The explanations should be clear and easy to interpret and should be supported by the data in the table. You can make other assertions by referring to the data given in the data file or other data you collected during the execution to help your explanations.