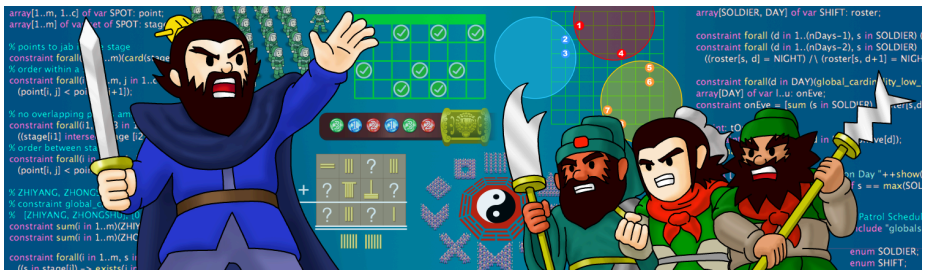
The Royal Hunt

Basic Modeling for Discrete Optimization: Assignment 4



Problem Statement

Emperor Xian desires a hunt to be organized by Liu Bei to celebrate the emperor's birthday. To make the hunt a success, Liu Bei must carefully match the court members to their horses so that everyone enjoys the day. There may be more court members than horses, in which case some court members will not ride, but each horse must be used. There may be more horses than court members in which case each court member must ride. But there are certain unwritten rules the hunt must follow:

- The emperor must enjoy the day more than anyone else.
- All court members must ride unless there is no horse left available,
- If a court member holds a higher rank than another, then either (a) the beauty of their horse can be no less than that assigned to the other, (b) the lower rank member does not ride, or (c) both court members do not ride.
- If a horse is faster than another then either (a) the rider of the faster horse has no less riding ability than that of the slower horse, (b) the faster horse has no rider, or (c) both horses have no rider.

The aim is to maximize the total enjoyment of the hunt over all riders. In truth the constraints are too hard to satisfy usually. So the last constraint can be violated for a penalty of 100 for each violation to the objective.

Data Format Specification

The input form for the Royal Hunt is a file named `data/royalhunt_p.dzn`, where p is the problem number, n being the number of court members (the first court member is the emperor), `rank` is an array mapping court members to rank (the higher, the better), `ability` is an array mapping court members to riding ability (the higher, the better), `m` is the number of horses available, `beauty` is an array mapping horses to their beauty (the higher, the better), `speed` is an array mapping horses to their speed, and `enjoy` is

a two-dimensional array mapping court members and horses to the enjoyment of the rider on that horse. If an entry in the array is negative, then the horse cannot be assigned to the court member.

The data declarations are hence:

```
int: n; % number of court members
set of int: COURT = 1..n;
int: emperor = 1;
array[COURT] of int: rank;
array[COURT] of int: ability;

int: m; % number of horses
set of int: HORSE;
array[HORSE] of int: beauty;
array[HORSE] of int: speed;

array[COURT,HORSE] of int: enjoy;
```

An example data file is

```
n = 6;
rank =    [8,5,5,4,2,2];
ability = [0,1,0,1,0,1];
m = 5;
beauty =  [1,5,3,8,8];
speed =   [6,3,5,4,4];

enjoy = [| 3,4,5,7,3
         | 1,6,3,4,8
         | 2,3,4,3,3
         | 9,6,2,3,4
         | 4,-1,3,3,3
         | 4,4,4,4,4 |];
```

which considers 6 court members and five horses. Your model's output should give the horse assigned to each rider (or 0 if they are assigned no horse), and the objective value. A solution for this data file (which is not necessarily the best) is given by the following assignment.

```
horse = [4,2,5,3,1,0];
objective = -178;
```

There are two violations of the horse constraint: horse 1 is faster than horse 2 but its rider 5 has less ability than horse 2's rider 2; and similarly horse 1 is faster than horse 3, but its rider has less ability than horse 3's rider 4. The total enjoyment is $7 + 6 + 3 + 2 + 4 = 22$.

As the **interface to the grader**, your model should contain variable declaration for `horse`, and it must calculate the correct objective value. Note that

you are allowed to use `::output_only` variable declarations, if you want to use a different point of view.

Instructions

Edit the provided `mzn` model files to solve the problems described above. Your implementations can be tested locally by using the **Run + Check** icon in the MiniZinc IDE. Once you are confident that you have solved the problem, submit your solution for grading.

Technical Requirements To complete this assignment you will a new version of the MiniZinc Bundle (`http://www.minizinc.org`). Your submissions to the Coursera grader are currently checked using MiniZinc version 2.5.5.

Handin This assignment contains 7 solution submissions and 1 model submissions. For solution submissions, we will retrieve the best/last solution the solver has found using your model and check its correctness and quality. For model submissions, we will retrieve your model file (.mzn) and run it on some hidden data to perform further tests.

From the MiniZinc IDE, the **Submit to Coursera** icon can be used to submit assignment for grading. Follow the instructions to apply your MiniZinc model(s) on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions.[1]. You can track the status of your submission on the **programming assignments** section of the course website.

[1] Please limit your number of submissions and test on your local machine first to avoid congestion on the Coursera servers