

Modeling for Discrete Optimization Assignment: Latin Killer

1 Problem Statement

A latin square is an $n \times n$ square filled with the numbers $1..n$ where each number appears exactly once in each column and each row. Latin squares are a well-studied object in mathematics. Less studied is the *latin killer square*. A latin killer square has each position in the $n \times n$ matrix assigned to a group a_{ij} , $1 \leq i, j \leq n$ between 0 and m . Each group g from $1..m$ is given a sum s_g , and the values assigned to positions in each group $g \in 1..m$ must sum up to s_g and be all different. The values assigned to positions in group 0 are not further constrained. The aim is to find a solution to a latin killer problem.

2 Data Format Specification

The input form for Latin Killer is a file named `data/latinkiller- p .dzn`, where p is the problem number with n the size of the square, m the number of groups, s an array $1..m$ of sums, and a a two dimensional matrix of assignments to groups.

For example a 6×6 problem with 9 groups might be defined by

```
n = 6;
m = 9;
s = [20, 11, 18, 16, 11, 18, 4, 6, 6];
a = [| 0, 1, 1, 2, 2, 3
      | 1, 1, 2, 2, 4, 3
      | 1, 5, 5, 4, 4, 3
      | 6, 6, 5, 7, 4, 3
      | 6, 6, 6, 8, 8, 8
      | 0, 0, 9, 0, 9, 9 |];
```

Your model's output should be in the form of a MINIZINC 2D array:

```
x = [| 1, 6, 4, 2, 3, 5
      | 3, 2, 1, 5, 6, 4
      | 5, 3, 2, 1, 4, 6
      | 2, 1, 6, 4, 5, 3
      | 6, 4, 5, 3, 1, 2
      | 4, 5, 3, 6, 2, 1 |];
```

Alternatively you can use a one dimensional array, and MINIZINC array coercion syntax (note that white space is irrelevant):

```
x = array2d(1..6,1..6, [ 1, 6, 4, 2, 3, 5,
                          3, 2, 1, 5, 6, 4,
                          5, 3, 2, 1, 4, 6,
                          2, 1, 6, 4, 5, 3,
                          6, 4, 5, 3, 1, 2,
                          4, 5, 3, 6, 2, 1 ]));
```

The template file `latinkiller.mzn` is provided to demonstrate reading the input data and reporting the solution.

3 Instructions

Edit `latinkiller.mzn` to solve the optimization problem described above. Your `latinkiller.mzn` implementation can be tested with the command,

```
mzn-gecode ./latinkiller.mzn ./data/<inputFileName>
```

Resources You will find several problem instances in the `data` directory provided with the hand-out.

Handin Run `submit.py` with the command, `python ./submit.py`. 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. However, it may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *feedback* section of the assignment website.

4 Technical Requirements

For completing the assignment you will need MINIZINC 2.0 (<http://www.minizinc.org/2.0/>) and the gecode solver (<http://www.gecode.org/download.html>). To submit the assignment for grading, you will need to have python 2.7.x installed on your system (installation instructions, <http://www.python.org/downloads/>).