离散优化建模：作业五
远走荆州

1 问题描述

刘备预见到袁绍无论采取什么行动都终将被曹操打败。他告诉袁绍他将前往荆州请增援回来帮助袁绍，不过实际上他计划逃去荆州。

他需要通过一段复杂的地段，在这里用一个方格阵来表示，而曹操的士兵则在这个方格阵中巡逻。在方格阵中，地段的地形有平地，山地，森林，城市和河流。他需要找到一条通往荆州的路，这条路需要满足以下约束：

- 他不会进入任何山地，因为山地太冷，他无法穿越这一地形。
- 他可以途径（进入）最多一座城市。当他进入一座城市时，他肯定会被认出，而当他进入另外一座城市时，他就会被曹操的士兵所抓住。
- 他的旅程需要尽量简短。否则，曹操的大部队会到达而且抓住他。
- 他的旅程不能有太多步数，否则他会迷路。

他的目标是令他旅程中遇到的士兵的数目尽量的少，因为这样才能令到他逃走的成功率最大化。

2 数据格式说明

远走荆州的输入文件是 `data/to_jin_`p`.dzn`，其中 p 是问题的序号。$nrow$ 是方格阵的行数，$ncol$ 是方格阵中的列数。$start\_row$ 是刘备起始位置的行数，$start\_col$ 是刘备起始位置的列数。$delay$ 是一个一维数组，表示穿越不同地形的方格所需要的最大天数。$timelimit$ 是刘备整个旅程的天数。$terrain$ 是一个二维数组，表示每一个方格与其对应的地形。Jin 是一个二维布尔型数组，代表每一个方格是否属于荆州。$soldier$ 是一个二维数组，代表每一个方格中的士兵中队的数目。$maxstep$ 则是在旅程中最大的步数。

刘备已经匆忙地建造了一个模型，它能在数据规模较小时得出解，不过对于实际需要求解的问题的数据量它好像还不够好。他建立的模型是（在文件 `to_jin.mzn` 中）：

```
int: nrow;
set of int: ROW = 1..nrow;
int: ncol;
```

```
set of int: COL = 1..ncol;

% Plains, Mountain, Forest, City, River
enum TERRAIN = { P, M, F, C, R };
array[TERRAIN] of int: delay;
int: timelimit;

array[ROW,COL] of TERRAIN: terrain;
array[ROW,COL] of int: soldier;
array[ROW,COL] of bool: Jin;

int: start_row;
int: start_col;

int: maxstep;
set of int: STEP = 1..maxstep;
set of int: STEP0 = 0..maxstep;

var STEP: steps;
array[ROW,COL] of var STEP0: visit;

% start at start position
constraint visit[start_row,start_col] = 1;

% only use steps moves
constraint sum(r in ROW, c in COL)(visit[r,c] >= 1) <= steps;
% reach Jin province
constraint exists(r in ROW, c in COL)(Jin[r,c] /\ visit[r,c] >= 1);

% visit at most one city
constraint not exists(r1,r2 in ROW, c1,c2 in COL)
                     ((r1 != r2 \/ c1 != c2)
                      /\ terrain[r1,c1] = C /\ terrain[r2,c2] = C
                      /\ visit[r1,c1] >= 1 /\ visit[r2,c2] >= 1);
```

```
% can't enter Mountain
constraint not exists(r in ROW, c in COL)(terrain[r,c] = M /\ visit[r,c] >= 1);

% visit only one place in every step
constraint forall(r1,r2 in ROW, c1,c2 in COL)
                 (r1 != r2 \/ c1 != c2
                  -> (visit[r1,c1] = 0
                      \/ visit[r2,c2] != visit[r1,c1]));
% steps form a path
constraint forall(s in 1..steps-1)
                 (exists(r1, r2 in ROW, c1, c2 in COL)
                        (abs(r1-r2) + abs(c1-c2) = 1
                         /\ visit[r1,c1] = s /\ visit[r2,c2] = s+1));
% no shortcuts on path
constraint forall(r1,r2 in ROW, c1,c2 in COL)
                 (abs(r1-r2) + abs(c1-c2) = 1 ->
                  visit[r1,c1] = 0 \/ visit[r2,c2] = 0 \/
                  abs(visit[r1,c1] - visit[r2,c2]) = 1);

% not too much delay
constraint time <= timelimit;
var int: time = sum(r in ROW, c in COL)(delay[terrain[r,c]]*(visit[r,c] >= 1));

% minimize the number of soldiers traversed
solve minimize obj;
var int: obj = sum(r in ROW, c in COL)((visit[r,c] > 0)*soldier[r,c]);

array[TERRAIN] of string: ter = [".", "#", "^", "C", "~"];

output
       ["%"] ++
       [ " " ++ ter[fix(terrain[r,c])] ++ if c = ncol then "\n%" else "" endif
       | r in ROW, c in COL ]
       ++ ["\n%"] ++
       [ if soldier[r,c] > 0 then show_int(2,soldier[r,c]) else " ." endif
```

```
        ++ if c = ncol then "\n%" else "" endif
      | r in ROW, c in COL ]
      ++ ["\n"] ++
      ["%"] ++
      [ if fix(visit[r,c]) > 0 then show_int(2,visit[r,c]) else " ." endif
        ++ if c = ncol then "\n%" else "" endif
      | r in ROW, c in COL ]
      ++ ["\nvisit = array2d(ROW,COL,\(visit));\nsteps = \(steps);\n" ++
          "time = \(time);\nobj = \(obj);"]
      ;
```

数据文件的例子如下：

```
nrow = 5;
ncol = 5;

start_row = 5;
start_col = 5;

delay = [ 1, 9, 3, 1, 2 ];
timelimit = 8;

terrain = [| P, P, P, P, M
           | P, C, M, P, P
           | P, P, C, P, P
           | P, R, P, C, F
           | M, R, F, P, P |];

Jin = [|  true,  true,  true, false, false
       |  true, false, false, false, false
       |  true, false, false, false, false
       | false, false, false, false, false
       | false, false, false, false, false |];

soldier = [| 3,1,4,8,1
           | 2,1,9,5,4
```

```
          | 6,1,4,8,1
          | 3,1,7,1,2
          | 6,1,2,4,1 |];

maxstep = 8;
```

它表示了一个 5 乘 5 的地图 (在文件 to_jin_0.dzn 中)。刘备的模型可以准确地找出这个数据文件对应的最优解。它在注释中还输出了地形图（“.'' 代表平原, “#” 代表山地, “^” 代表森林, “C” 代表城市，“~” 代表河流），士兵分布图，和路径图。对于以上数据它的输出是：

```
% . . . . #
% . C # . .
% . . C . .
% . ~ . C ^
% # ~ ^ . .
%
% 3 1 4 8 1
% 2 1 9 5 4
% 6 1 4 8 1
% 3 1 7 1 2
% 6 1 2 4 1
%
% . . . . .
% . . . . .
% 7 6 . . .
% . 5 4 3 .
% . . . 2 1
%
visit = array2d(ROW,COL,[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 6, 0, 0,
0, 0, 5, 4, 3, 0, 0, 0, 0, 2, 1]);
steps = 7;
time = 8;
obj = 21;
----------
==========
```

这表示一条长为 7 步，需时 8 天的路径图和途中将遇到 21 个士兵中队。

这次作业的目标是优化已经给出的模型，从而令它更有效率。这有可能需要重写约束，甚至是更改决策变量。不过数据文件的输入格式不能更改，还有输出也必须跟刘备的模型相同（除了被注释的部分）。建立一个可以解决规模最大的数据文件的模型是很困难的，因此只作为对高水平的学生的要求。

3 指引

你可以编辑 `to_jin.mzn` 模型文件来解决上述优化问题。你实现的模型 `to_jin.mzn` 可以用提供的数据文件进行测试。在 MiniZinc IDE 中，你可以通过点击 *Run* 按钮在本地测试和运行。或者在命令行中输入

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

进行本地测试和运行。两种情况下，你的模型都是用 MiniZinc 进行编译同时用 GECODE 求解器求解。

参考资料 你可以在 `data` 文件夹下找到讲义中的几个问题实例（的数据文件）。

提交作业 这次的作业包含有 5 个答案提交部分和 1 个模型提交部分。对于答案提交部分，我们将会提交求解器求解你的模型所得到的最好 / 最后的答案，然后检查它的正确性和得分。对于模型提交部分，我们将会提交你的模型文件 (.mzn) 然后用一些隐藏的数据文件来做进一步检查。

在 MiniZinc IDE，点击 *coursera* 图标可以用于提交作业。若采用命令行方式，`submit.py` 可以用于提交作业。无论采用那种方法，你都需要根据本指引中的要求完成作业各部分的 MiniZinc 模型。你可以多次提交，最终作业分数是你的最高的一次。[1]作业的打分过程可能需要几分钟，请耐心等候。你可以在课程网站上的*编程作业* 版块查看你的作业提交状况。

4 软件要求

为了完成作业,你需要安装MiniZinc 2.1.x和GECODE 5.0.x 求解器。这些软件都会包含在MiniZinc IDE 2.1.2 (`http://www.minizinc.org`)的集成版本中。如果你需要通过命令行提交作业，你需要安装Python 3.5.x。

[1]答案提交部分并没有次数限制。但是，模型提交部分只能提交有限次。