离散优化建模：作业十二
安置难民

1 问题描述

由于悟空忙于扑灭火焰山上的火势，一处野火最终摧毁了很多山上的村庄。所有村民逃跑到山谷的安全地，他们将在那里作为难民生活。由于拥挤的情况，神农希望把难民分到 k 个不同的难民营里面。目的是使得难民营尽可能地和谐，所以他希望每一个人都与在同一个难民营中熟悉。显然来自同一个家庭的人们应该安排到同一个营地，已知互相不喜欢的人应该分到不同的营地。注意到如果能更和谐，神农不会使用所有的营地。

2 数据格式说明

安排难民的输入数据包含在名为 `data/refugee_`p`.dzn` 的文件当中，其中 p 是问题序号。数据文件定义了以下数量:

- $PROBLEM\_STAGE$ 是对于每一个测试例子的一个独特的编号，而且只会用于评分，（你可以忽略它）
- p 是问题数字,
- n 是人口的数量,
- k 是可以使用的营地数量,
- $maxsize$ 是在任意营地可以容纳人数的最大数量,
- E 是一个二维数组代表已知信息—每一行代表两个人认识彼此，
- ML 是一个二维数组— 每一行代表两个人来自同一个家庭, 和
- CL 是一个二维数组，维度大小与人口数量相同 — 每一行代表两个人非常不喜欢对方。

目标是得出一个赋值 x 表示每个人被安排到不同的营地里。

数据声明和主要的决策变量如下:

```
int: PROBLEM_STAGE;  % can be ignored for modeling purposes


int: n;
```

```
set of int: PERSON = 1..n;
int: k;
set of int: CAMPSITE = 1..k;
int: maxsize;
array[int,1..2] of PERSON: E;
set of int: EDGE = index_set_1of2(E);
array[int,1..2] of int: ML;
array[int,1..2] of int: CL;
set of int: FAMILY = index_set_1of2(ML);
set of int: DISLIKE = index_set_1of2(CL);

array[PERSON] of var CAMPSITE: x;
```

每一个阶段输出要求有以下形式

```
x = [对于每个人营地的安排];
obj = 安排的和谐度;
```

神农关于如何构建最和谐的安排有很好的想法。如果 m 表示两者互相认识的关系的数量，和 d_i 表示第 i 个人认识的人。

$$Harmony = \sum_{i=1}^{n}\sum_{j=1}^{n}(2m((i,j) \in E) - d_i d_j)(x[i] = x[j])$$

另外一个计算方法是

$$Harmony = \sum_{c=1}^{k}\left(4m \sum_{(i,j)\in E}(x[i] = c \wedge x[j] = c) - (\sum_{i=1}^{n} d_i(x[i] = c))^2\right)$$

目的是在满足每一对来自同一家庭的人被安排在同一个营地里以及每一对互相讨厌的人安排在不同的营地里这两个约束的基础上，最大化和谐度。

比如，对于数据文件

```
PROBLEM_STAGE = 99; % please ignore
n = 11;
k = 4;
maxsize = 6;
E =  [| 1,2  | 1,3  | 1,6
      | 1,10 | 2,3  | 4,5
```

```
      | 4,6  | 4,10 | 5,6
      | 7,8  | 7,9  | 7,10
      | 8,9  | 8,11 |];
ML = [| 1,2  | 3,10 | 6,7  |];
CL = [| 3,5  | 7,10 | 3,11 |];
```

包含 11 个人和 4 个最大可容纳 6 人的营地。

一个最优解是

```
x = [1, 1, 1, 2, 2, 2, 2, 3, 3, 1, 3];
obj = 226;
```

用了 4 个营地中的 3 个。注意到家庭成员都在同一个营地里 ($\{1,2\} \subseteq 1$, $\{3,10\} \subseteq 1$, $\{6,7\} \subseteq 2$), 不喜欢的人在不同的营地 ($3 \in 1, 5 \in 2$), ($7 \in 2, 10 \in 1$) 和 ($3 \in 1, 11 \in 3$)。

你可能需要根据输入的大小来改变你的模型如何工作。有一些输入是很小的，有一些会很大。

你甚至会使用完全不同的方法来求解这个问题。可以通过创建一个定制的局部搜索求解器并记录它能找到的最好的解, 并使用一个“假的”Minizinc 模型来回答这个问题实例。(会在后面解释细节)

为了能使你的局部搜索求解器更方便读入数据文件，我们提供了一个名为 `dzn2ascii.mzn` 的模型。它可以将数据文件以 ASCII 数字的方式写出，忽略了 `PROBLEM_STAGE` 参数。格式按顺序是 n, k, $maxsize$, m（相互认识的关系的数量），$2m$ 个数字表示认识的关系，ff（家庭成员关系的数量），$2ff$ 个数字表示家庭成员关系，dd（相互不喜欢的数量），和 $2dd$ 个数字表示不喜欢的关系。比如上述数据文件会被翻译成

```
11
4
6
14
1 2 1 3 1 6 1 10 2 3 4 5 4 6 4 10 5 6 7 8 7 9 7 10 8 9 8 11
3
1 2 3 10 6 7
3
3 5 7 10 3 11
```

3 指引

你可以编辑 `refugee.mzn` 模型文件来解决上述优化问题。你可以使用在本课程中学到的任何关于建模和搜索的技巧。你实现的模型 `refugee.mzn` 可以用提供的数据文件进行测试。在 MiniZ-

INC 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 ./refugee.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。

这次作业的**所有**测试都是答案提交。所以，你应该使用 MiniZinc 模型或者你定制的局部搜索求解器来为每一个数据文件求得最好的解，然后建立一个单独的假的模型来专门用作答案提交。

假的模型只会输出你可以对数据文件找到的最好的解，利用 `PROBLEM_STAGE` 参数来识别。以下是在`dummy.mzn` 里提供的假模型，还有假的最优解。

```
int: PROBLEM_STAGE;
int: n;
set of int: PERSON = 1..n;
int: k;
int: maxsize;
array[int,1..2] of int: ML;
array[int,1..2] of int: CL;
array[int,1..2] of PERSON: E;
solve satisfy;    % dummy solve
output [
  if PROBLEM_STAGE == 1 then
  "x = [1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 3];\n" ++
  "obj = 174;\n"

  elseif PROBLEM_STAGE == 2 then
  "x = [1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 3];\n" ++
  "obj = 174;\n"

  ...

  elseif PROBLEM_STAGE == 12 then
  "x = [1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 3];\n" ++
  "obj = 174;\n"

  else
```

```
  "ERROR: Did you modify the PROBLEM_STAGE in any .dzn file?"
  endif
];
```

注意对于所有答案提交，你只需要一个假模型。

这也就是说你可以用任意方法来求解问题。你可能希望用某一种局部搜索算法来完成它。或者你可以使用同一个 MiniZinc 模型来生成部分或者所有答案。

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 12 个答案提交部分。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。

在 MiniZinc IDE，点击 *Submit to 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.X (`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]问题解的提交并没有次数限制。