离散优化建模：作业十一
厨房值班

1 问题描述

通天教主除了是一个恶棍，他也是一个食物和美酒的行家。通天教主的首席厨师有一个大厨师团队来准备老板的严格要求的食物。由于通天教主会经常在他不满意食物的时候赶走员工，首席厨师现在有一个很严格的值班排表来保证所有的食物是最高的品质，而没有任何员工太累而导致犯错。这次的目标是建一个员工的轮值表，使得首席厨师设下的规则可以被满足，而且在每一天每一个班次都有足够的人手值班，然后使他们能休息的时间最大化。

2 数据格式说明

厨房值班的输入包含在 `data/kroster_`p`.dzn` 的文件里，其中 p 是问题的序号。数据文件定义了以下数量:

- $week\_length$ 是排班表中一周的长度，
- $nb\_workers$ 是需要排班的员工的数量，
- $min\_daysoff$ 是一个员工需要的连续休息的最小天数，
- $max\_daysoff$ 是一个员工允许的连续休息的最大天数，
- min_work 是一个员工需要连续工作的最小天数，
- max_work 是一个员工可以连续工作的最大天数，
- nb_shifts 是班次的类型数量，
- $shift\_name$ 是每一种班次的名字（一个字母的字符串），
- $shift\_block\_min$ 是一个员工在下一种班次之前必须值班的最小天数，
- $shift\_block\_max$ 是一个员工在下一种值班类型之前可以工作的最大天数，
- $nb\_forbidden$ 是不被允许的连续排班种类序列的数量，
- $forbidden\_before$ 是一个表示禁止序列的第一种值班类型的数组，
- $forbidden\_after$ 是一个表示禁止序列的第二种值班类型的数组，而最后还有

- *temp_req* 是一个二维数组表示每一天每一班需要的最小员工数量。

目标是生成一个二维数组 *plan*，表示一个其 $nb\_workers + 1$ 周，每周长度为 *week_length* 的循环排班表。

数据声明和主要的决策变量如下：

```
int: week_length;   % Length of the schedule
set of int: DAY = 1..week_length;
int: nb_workers;    % Number of the workers
int: min_daysoff;   % Minimal length of days-off blocks
int: max_daysoff;   % Maximal length of days-off blocks
int: min_work;      % Minimal length of work blocks
int: max_work;      % Maximal length of work blocks
int: nb_shifts;                              % Number of shifts
set of int: SHIFT = 1..nb_shifts;            % Set of shifts
int: OFF = nb_shifts+1;                      % off shift representation
set of int: SHIFT0 = 1..nb_shifts+1;         % Set of shifts + off shift
array[SHIFT] of string: shift_name;          % Name of shifts
array[SHIFT] of int:    shift_block_min;     % Minimal length of blocks
array[SHIFT] of int:    shift_block_max;     % Maximal length of blocks
int: nb_forbidden;                           % Number of forbidden shift sequences
set of int: FORBIDDEN = 1..nb_forbidden;     % Set of forbidden shift sequences
array[FORBIDDEN] of int: forbidden_before;% The shift before in the forbidden sequence
array[FORBIDDEN] of int: forbidden_after; % The shift after in the forbidden sequence
array[SHIFT, DAY] of int: temp_req;     % Temporal requirements


array[WEEK,DAY] of var SHIFT0: plan;
```

输出需要有如下格式

```
plan = 2 维数组表示在 nb_workers+1 星期里的每天排班;
obj = 在排班计划表中有多少休息的排班;
```

幸好首席厨师已经建了一个模型，叫 `kroster.mzn`。这个项目的目的是对同一个问题建立一个线性模型。输入和输出需要跟原来的 `kroster.mzn` 相同，不过模型只能包含线性约束。也就是说，当用 Gecode 编译时，FlatZinc 只能有 `int_lin_eq` 和 `int_lin_le`。你可以发现纯线性模型在使用 COIN OR CBC 求解器的时候可以比原来的模型更高效。

例如对于给定的数据文件

```
week_length = 7;
nb_workers = 3;
min_daysoff = 1;
max_daysoff = 2;
min_work = 2;
max_work = 5;
nb_shifts = 3;
shift_name = ["D", "A", "N"];
shift_block_min = [2, 1, 2];
shift_block_max = [5, 4, 4];
nb_forbidden = 1;
forbidden_before = [1];
forbidden_after = [2];
temp_req = [| 1, 1, 1, 0, 1, 1, 0
            | 1, 1, 1, 1, 2, 1, 0
            | 0, 1, 1, 0, 0, 0, 0];
```

我们 3 个员工，在一周 7 天三种不同排班种类。只有一种禁止的连续排班序列，D 之后是 A。一个最优解是

```
% D D D . A A .
% . N N A A . .
% A A A . D D .
% D D D . A A .
%
plan = [| 1, 1, 1, 4, 2, 2, 4
        | 4, 3, 3, 2, 2, 4, 4,
        | 2, 2, 2, 4, 1, 1, 4,
        | 1, 1, 1, 4, 2, 2, 4];
obj = 7;
```

注意到计划是四个星期的，尽管只有 3 个员工。这是因为为了保证这个计划是合法的周期性计划，我们有额外的一周是第一周的副本。这样当约束满足时可以保证从最后一周到第一周的转变是允许的。这个周期性的排班表每三周对于第一个员工的来说是 "$DDD.AA..NNAA..AAA.DD.$", 对第二个员工是 "$.NNAAA..AAA.DD.DDD.AA.$", 而对第三个员工是 "$AAA.DD.DDD.AA..NNAA..$"。注意到每一次排班满足最小和最大排班序列的约束，而且不能包含禁止的连续排班序列。而每个员工在周期排班中有 `obj = 7` 的休息日。

原来的 `kroster.mzn` 模型输出一个注释表示这个排班，你的提交不需要这样做。不过你需要对于一个正确的`.dzn` 输出 `plan` 和 `obj` 变量。

3 指引

你可以编辑 `kroster.mzn` 模型文件来解决上述优化问题。你实现的模型 `kroster.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

```
mzn2fzn ./kroster.mzn ./data/<inputFileName>
```

来创建 `kroster.fzn` 和检查是否只包含线性约束。你可以使用

```
mzn-cbc -G std ./kroster.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 COIN OR CBC 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 3 个答案提交部分和 6 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质量。对于模型提交部分，我们将会提交你的模型文件 (.mzn) 然后用一些未公开的数据文件来做进一步检查。

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 求解器和 COIN OR CBC。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。但是，**模型提交部分**只能提交有限次。