离散优化建模：作业七
追求者来访

1 问题描述

刘备深深地被孙权的妹妹，孙尚香所吸引。在他成功地为她谱写出一首词曲之后，孙尚香要求刘备帮忙计划招待来拜访的追求者, 当然这些追求者里有刘备。不过她实际的目标是在她母亲不知道的情况下安排尽量多的功夫课。而刘备的目的是要保证在孙尚香的行程中有尽可能多的时间跟他在一起，不过这也是要在保证安排中能有最多的功夫课程的前提下。

孙尚香有一帮追求者，而她的母亲希望她可以花尽量多的时间跟追求者在一起。除了某些时段别人在用房间以外，她可以用金房，红房和蓝房来招待不同的追求者。每一个追求者有固定的拜访的次数，而拜访的时间有一个上限和下限，这取决于拜访时使用的房间。这个安排需要考虑到这个时间的限制。孙尚香希望尽早结束这些拜访，从而挤出时间来上功夫课程。不过即便是没有功夫课的日程也必须考虑时间限制。这份计划也必须考虑到从一个房间移动到另外一个房间的所需时间。

刘备为孙尚香准备了两份日程安排，分别要考虑到了有无功夫课程的情况。这些日程的首要目标是功夫课程的数量最多，次要目标则是令孙尚香与刘备一起的时间最多。

2 数据格式说明

追求者来访问题的输入是名为 `data/suitor_schedule_`p`.dzn` 的文件，其中 p 是不同问题的序号。$SUITOR$ 是一个包括 $LiuBei$ 的枚举类型变量。n 则是需要会面的总数。$suitor$ 代表每场会面中出席的追求者。$mintime$ 是一个二维数组，代表某个的追求者在某个房间所对应的会面的最少的时数（你可以假设这最少是 1）。$maxtime$ 也是一个二维数组，代表某个追求者在某个房间所对应的会面可允许的最大时数（你可以假设 $maxtime[s,r] \geq mintime[s,r]$）。$move$ 是一个二维数组，代表从一个房间到另外一个房间所需要的时间（你可以假设移动到相同房间的时间为 0）。$ndays$ 代表需要安排的时间的天数。$earliest$ 是每天最早的可以开始会面的时间，$latest$ 是每天最晚结束会面的时间。$minsep$ 是每一节功夫课结束到下一节功夫课开始所需要的最少的间隔时间。$usedstart$ 代表每一个房间每天被其他人使用的开始的小时，$useddur$ 代表每一个房间每天用于其他方面的小时数。最后因为此次作业是分阶段的，我们加入了一个 $stage$ 参数来表示这一个数据文件是属于哪一个阶段的。

因此数据与决策变量的声明如下：

```
% scheduling suitors
enum SUITOR;
SUITOR: LiuBei; % which suitor is LiuBei
```

```
int: n; % number of meetings
set of int: MEETING = 1..n;
array[MEETING] of SUITOR: suitor;

enum ROOM = { Red, Gold, Blue };
array[SUITOR,ROOM] of int: mintime;
array[SUITOR,ROOM] of int: maxtime;
constraint forall(s in SUITOR, r in ROOM)
                 (assert(mintime[s,r] >= 1 /\ maxtime[s,r] >= mintime[s,r],
                  "error in mintime/maxtime at [\(s),\(r)]\n"));

array[ROOM,ROOM] of int: move;
constraint forall(r in ROOM)(assert(move[r,r] = 0,"move[\(r),\(r)] != 0\n"));

int: ndays; % number of days
set of int: DAY = 1..ndays;
int: earliest; % time for earliest meeting start
int: latest;  % time for end of latest meeting

int: lessontime; % time for kung fu lesson;
int: minsep;     % minimum time between consecutive kung fu lessons

array[ROOM] of int: usedstart;  % time others use each room each day
array[ROOM] of int: useddur;    % durations of others use

enum STAGE = {A,B,C};
STAGE: stage;

set of int: TIME = 0..24*ndays;

array[MEETING] of var TIME: start;
array[MEETING] of var TIME: dur;    % duration in false schedule
array[MEETING] of var ROOM: room;
set of int: KUNGFU = 1..n;
array[KUNGFU] of var int: kungfu;  % start time for each lesson
```

```
                                    % unused lessons have start times after 24*ndays
```

其中一个样本数据文件如下：

```
SUITOR = { S1, S2, S3 };
LiuBei = S1;
n = 2 + 2 + 2 ;
suitor = [ S1, S1, S2, S2, S3, S3 ];

mintime = [|  1, 2, 3
           |  2, 2, 1
           |  3, 3, 3 |];

maxtime = [| 2, 3, 4
           | 2, 4, 3
           | 3, 4, 3 |];

move = [|  0, 1, 1
        |  1, 0, 2
        |  1, 2, 0 |];

ndays = 2;
earliest = 8;
latest = 18;

lessontime = 1;
minsep = 2;

usedstart = [ 10, 12, 14 ];
useddur   = [ 1, 2, 1 ];
```

上面的问题需要考虑 3 个追求者而每个追求者需要在两天内有两次会面的情况。

3 A 阶段

在 A 阶段你的模型应该忽略功夫课程和在房间之间转移的时间。你只需要计划如何让孙尚香会见所有追求者。

你的模型在所有部分的输出都应该给出决策变量和目标数值。目标数值取决于 100 与功夫课程的乘积，加上与刘备的会面的小时数。在这一阶段，功夫课程的时间可以全部设为大于 $24 \times ndays$。比如对于以上的数据，如果加入 $stage = \text{A};$, 你的程序可能输出：

```
start = [32, 8, 16, 12, 13, 36];
dur =[4, 4, 1, 1, 3, 3];
room = [Blue, Blue, Blue, Blue, Red, Red];
kungfu = [49, 49, 49, 49, 49, 49];
obj = 8;
```

这计划了 6 场会面，其中刘备跟孙尚香的会面有 8 个小时。注意到这个计划在考虑到移动时间之后是不可能的，因为在 13 时有一场会面在蓝房结束而在红房又马上开始了另外一场会面。

```
      1   1   1   1   1
  8   0   2   4   6   8
 |B B B B|B|R R R|B|. .
 |   LB  |2|  3  |2|

  3   3   3   3   4   4
  2   4   6   8   0   2
 |B B B B|R R R|. . . .
 |  LB   |  3  |
```

4 B 阶段

在 B 阶段，你应该考虑到不同房间之间的移动时间，不过仍然可以忽略功夫课程。

对于上述的样本数据，如果加入 $stage = \text{B};$, 你的程序可能的输出如下：

```
start = [32, 8, 40, 16, 37, 13];
dur =[4, 4, 2, 2, 3, 3];
room = [Blue, Blue, Red, Red, Red, Red];
kungfu = [49, 49, 49, 49, 49, 49];
obj = 8;
```

现在你可以看到在蓝房和红房之间的移动会导致有一个小时的延迟。

```
      1   1   1   1   1
  8   0   2   4   6   8
```

```
|B B B B|.|R R R|R R|.
|   LB  | |  3  | 2 |

 3   3   3   3   4   4
 2   4   6   8   0   2
|B B B B|.|R R R|R R|.
|   LB  | |  3  | 2 |
```

5 C 阶段

现在你可以处理考虑了功夫课程的情况。对于上述的样本数据，如果加入 $stage = \mathrm{C};$, 你的程序可能的输出如下：

```
start = [8, 16, 36, 32, 13, 38];
dur = [4, 2, 2, 3, 3, 3];
room = [Blue, Red, Red, Gold, Red, Red];
kungfu = [11, 17, 34, 49, 52, 55];
obj = 304;
```

需注意这并不是最优解!

在 11 时有我们计划了三个功夫课程，分别在 11 时（在与刘备第一次会面的结束前一个小时），17 时（在与刘备第二次会面结束前一个小时）和 34 时（在与 S2 的第一次会面结束前一个小时）。需注意到（实际上）功夫课程的时间应该有不递减的顺序。而且要注意到虽然在计划上显示是 6 小时，但刘备跟孙尚香共处的时间实际上是 4 小时。这是因为有两个小时是用在了功夫课上。

```
     1   1   1   1   1
 8   0   2   4   6   8
|B B B B|.|R R R|R R|.
|   LB  | |  3  | LB|
       K           K

 3   3   3   3   4   4
 2   4   6   8   0   2
|G G G|.|R R|R R R|. .
|  2  | | 2 |  3  |
     K
```

按照 Coursera 评分系统的要求，你需要建构一个模型 `suitor_schedule.mzn` 去完成全部三个阶段。在你的模型中，你需要以 `stage` 参数去判断应该考虑哪些约束以符合不同阶段的要求。

6 指引

你可以编辑 `suitor_schedule.mzn` 模型文件来解决上述优化问题。你实现的模型 `suitor_schedule.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 6 个答案提交部分和 3 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。

7 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。