CS24011：实验室2：约束满意度

伊恩普拉特哈特曼
弗朗西斯科洛博

学术课程： 2022-23

Git标签：24011-lab2-S-约束

介绍

本实验室采用了古斯塔沃·尼迈耶在Python约束模块中实现的约束满足求解器。它由两个练习组成。

- 在第一个课程中，您将解决本课程入门讲座中遇到的“逻辑难题”（关于到达机场的旅客）。本练习的目的是练习将一个用英语非正式表达的问题转化为一个约束满足问题（CSP）。如果您成功了，您将看到Python约束满足包工作得非常好。
- 在第二个练习中，我们将相同的约束满足包应用于娱乐数学中的一个更抽象的问题。问题的问题是生成全对角魔法方块。（我们将在下面解释一下这些内容。）本练习的目的是说明CSP求解器的一些局限性——特别是当给出的问题实例的规模增大时。

有关约束模块的文档，请参见

https://labix.org/python约束

试试这个。首先，您将需要安装Python约束模块。只要在命令shell中输入它就足够了

```
$ pip安装蟒约束
```

逻辑谜题

让我们回顾一下在入门讲座中遇到的逻辑难题。Four travellers — Claude, Olga, Pablo and Scott — are returning from four countries — Peru, Romania, Taiwan and Yemen — departing at four different times — 2:30, 3:30, 4:30 and 5:50 (one hopes: in the afternoon).没有两个旅客从同一个目的地返回，也没有两个旅客同时出发。*我们得到了以下特殊的约束条件：*

1. 奥尔加比从也门起飞的旅行者早2个小时离开。
2. 克劳德要么是下午2点半离开的人，要么是下午3点半离开的旅行者。
3. 下午2点30分离开的人正从秘鲁起飞。
4. 从也门起飞的人比从台湾起飞的人更早离开。
5. 这四名旅行者分别是巴勃罗，从也门起飞的旅行者，下午2：30离开的人和下午3：30离开的人。

我们需要使用Python约束模块来设置这个设置。我是这样做的：

```
from constraint import Problem, AllDifferentConstraint
problem= Problem()

people= ["claude", "olga", "pablo", "scott"]
times= ["2:30", "3:30", "4:30", "5:30"]
destinations= ["peru", "romania", "taiwan", "yemen"]

t_variables= list(map(lambda x: "t_"+x, people))
d_variables= list(map(lambda x: "d_"+x, people))

problem.addVariables(t_variables, times)
problem.addVariables(d_variables, destinations)

problem.addConstraint(AllDifferentConstraint(),t_variables)
problem.addConstraint(AllDifferentConstraint(),d_variables)
```

这个代码应该是不言自明的。请注意一个聪明的捷径，它可以说某些变量集必须有不同的值。（必须手工写出来会很烦人。）如果没有任何特殊的约束，我们可以得到如下的解。

```
解决=问题。getSolutions ()
打印（索恩斯）
```

你会注意到有相当多这样的人。

现在使用Python约束模块表示上述四个特殊约束。为了帮助你，我们已经为你做了第一个了。

```
# Olga is leaving 2 hours before the traveller from Yemen.
for person in people:
    problem.addConstraint(
        (lambda x,y,z: (y != "yemen") or
            ((x == "4:30") and (z == "2:30")) or
            ((x == "5:30") and (z == "3:30"))),
        ["t_"+person, "d_"+person, "t_olga"])
```

注意这里使用了lambda表达式来表示约束。这是三个变量x，y和z的函数，它们根据这些变量的值返回一个布尔值。（您应该能够读取该功能的主体。）当，在for循环中，人们计算为-说-claude时，附加约束方法将问题约束应用于三个变量t_claude、d_claude、t_olga，允许解决方案只分配函数返回true的各自值x、y、z的元组。

其他一些特殊的约束也有类似的处理方法。也许最令人困惑的是第四个。作为一个暗示，你可以把它解读为一组陈述的集合，大意是某些描述不共同引用：巴勃罗不是从也门起飞，在2：30和3：30离开；无论从也门起飞的，也不会在2：30也不在3：30离开。当你让一切都能正常工作时，你会发现实际上只有一个解决方案：

```
[{'t_olga': '2:30', 'd_olga': 'peru',
    't_claude': '3:30', 'd_claude': 'romania',
    'd_pablo': 'taiwan', 'd_scott': 'yemen',
    't_scott': '4:30', 't_pablo': '5:30'}]
```

印刷的顺序有点随机，但不用担心。

如果你去掉第一个约束（关于奥尔加比从也门旅行者离开两个小时），你会看到有更多的解决方案。..

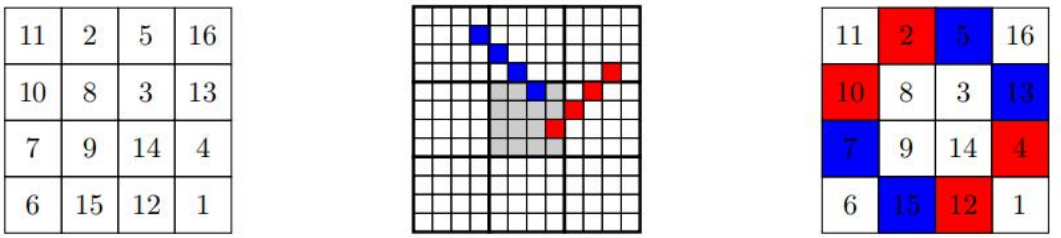
任务1：编写一个函数旅行者（列表），以获取一对对的列表。每对时间都由一个旅行者和一个时间，e组成。g.[“奥尔加”，“2：30”]，被解释为奥尔加在2：30旅行的约束。您的函数需要设置具有上面的特殊约束2到5的逻辑难题，以及作为参数列表传递的额外约束，并返回使用获得的解决方案列表限制问题。得到系统的解（）。

对于困难，呼叫旅行者（[[“olga”，“2：30”]]）应该返回列表与上面给出的一个解决方案（可能与问题变量的不同顺序），而呼叫旅行者（[[“olga”，“5：30”]]）应该返回一个空列表，因为没有解决方案，等等。

评估：本次练习的任务有6个分数。您的功能将被测试
各种测试用例，以及完全正确的返回值的比例将决定你的分数（在6分中）。如前所述，对于某些测试用例，可能没有解决方案（在这种情况下，您的函数应该返回一个空列表）。

泛对角魔法方块

设n是一个正整数。一个大小为n的魔法平方是一个n×n个列表，它的条目是整数1到n^2排列方式是，每一行、每一列和两个对角线的和是相同的。图1(a)显示了一个n = 4的例子，其中每行、每一行、每一列和两个对角线的项加起来是34。图1(b)显示了相同的4 4 个方块重复在一个3×3



(a) A magic square. (b) Two diagonals. (c) Broken diagonals.

图1：魔法方方形和断裂对角线。

超电网。考虑中心的4×4正方形（阴影），并从任何边界单元格开始，假设我们沿对角线（在四个方向中的任何一个）进行4个正方形的距离：两条这样的对角线路径显示（蓝色和红色）。由于中心正方形是重复的，这两条对角线路径实际上是原始正方形的破碎对角线，如图所示。1(c).注意这两个主要对角线只是断裂对角线的特殊情况。一个简单的检查显示，图中平方的断裂对角线。1(a)加起来并不是全部是34。*大小为n的全对角魔法平方是一个n×n列表，其条目是整数1到n^2*排列方式是，每一行、每一列和每个断裂的对角线的和是相同的。有些令人惊讶的是，泛对角线的魔法广场的存在：图。2显示了n = 4的泛对角线魔法方块。

13	12	7	2
8	1	14	11
10	15	4	5
3	6	9	16

图2：一个大小为4的泛对角线的魔法广场

任务2：大小为n的泛对角线魔法平方中的每行、列和断对角线的公共和必须相同。编写一个python函数CommonSum (n)来计算这个数字。

评估：本次练习的任务有2个分数。您的功能将被自动测试-
在许多情况下，完全正确的返回值的比例将决定你的分数（在2分中）。

下面的Python代码建立了查找大小为n的魔法方块的基本框架。

```
from constraint import Problem,
    AllDifferentConstraint, ExactSumConstraint
problem= Problem()
problem.addVariables(range(0,n*n), range(1,n*n+1))
```

因此，我们有n^2变量（编号为0到n-1），每个变量的取值范围为1到n^2包括的为了明确起见，我们假设逐行编号的变量。0，...，n-1沿底部一行，n，。..，2n-1沿着下一行，以此类推，与n^2- n，...，n-1on^{2}最上面的一行。我们需要说，这些变量都取不同的值。请记住，约束包包含了一个执行此操作的捷径：

```
问题附加约束（一个（），范围（0，n * n））
```

接下来，我们需要约束，说明必须添加所有行、所有列和两个对角线
增加相同的数字。为了帮助您，这里是对行的约束。

```
for row in range(n):
    problem.addConstraint(ExactSumConstraint(CommonSum(n)),
        [row * n + i for i in range(n)])
```

写相应的约束，列必须加到相同的数字，并且
两个主对角线必须加起来等于这个数字的约束条件。现在把所有的东西都放在一起，这样你就可以计算出任何给定大小的魔法方块的总数。你可能会想要使用像

```
解决=问题。getSolutions ()
打印（索恩斯）
```

为了使我们可以测试您的代码，我们需要添加一些额外的约束条件。..

任务3：编写一个函数msqList（n，对列表），当n传递一个整数n，pair列表传递一个（可能是空的）对列表[v，i]，与0≤v ＜ n^2和1≤≤^{2}，返回大小为n的魔法方块列表，这样，对于列表中的每一对[v，i]，位置v都用整数i填充。同样，您需要使用约束。问题。获取解决方案（）来获得这个解决方案列表。

记住，魔法方块中的位置假定从0到n^2-1，逐行排列。例如，msqList（4，[[0,13]，[1,12]，[2,7]]）应该返回一个列表
4个解决方案，所有这些都包括在内

```
{0:   13 ,   1:   12 ,   2:   7}
```

另一方面，msqList（3，[]）-i。e.使用一个固定位置的空列表-应该返回一个包含8个解决方案的列表。如果您的程序对于n ＞ 4和一个小的（或特别空的）额外约束列表非常慢，请不要担心；我们将测试应该在合理时间内运行的示例。

评估：本次练习的任务有6个分数。您的功能将被测试
各种测试用例，以及完全正确的返回值的比例将决定你的分数（在6分中）。请注意，对于某些测试用例，可能没有解决方案（在这种情况下，您的函数应该返回一个空列表）。

现在写两个约束条件，说断裂的对角线加起来必须是相同的数字。
这更棘手。作为提示，我们建议您创建一个列表，比如dd，这样dd的每个成员都是沿着断对角线的变量列表。

例如，如果n = 4，dd将包含列表[0,5,10,15]和[4,9,14,3]。

为了使我们可以测试您的代码，我们再次需要添加额外的约束。..

任务4：编写一个函数pmsList（n，对列表），当n传递一个整数n，对列表传递一个（可能是空的）对列表[v，i]，与0≤v < n^2和1≤≤^{2}，*返回大小为n的全对角线魔法方块的列表，这样，对于列表中的每一对[v，i]，位置v都用整数i填充*。您需要使用约束条件。问题。获取解决方案（）来获得这个解决方案列表。

同样，记住魔法方块中的位置被假为从0到n^2-1，逐行排列。因此，pmsList（4，[[0,13]，[1,12]，[2,7]]）应该返回的解决方案之一是图中的泛对角线魔法方块。2，即

```
{0: 13, 1: 12, 2: 7, 3: 2,
    4: 8, 5: 1, 6: 14, 7: 11,
    8: 10, 9: 15, 10: 4, 11: 5,
    12: 3, 13: 6, 14: 9, 15: 16}
```

尽管实际的Python输出不太可能像这里所做的那样对位置进行排序。顺便注意一下，pmsList（3，[]）应该返回[]，因为没有3个×3的泛对角线魔法方块。

另外，如果您的程序对于n > 4和一个小的（或特别空的）额外约束列表非常慢，请不要担心；我们将测试应该在合理时间内运行的示例。

评估：本次练习的任务有0个分数。您的功能将被测试各种测试用例，以及完全正确的返回值的比例将决定你的分数（在6分中）。请注意，对于某些测试用例，可能没有解决方案（在这种情况下，您的函数应该返回一个空列表）。

【共20分】

提交

请按照COMP24011中的自述文件。md说明操作 2022 Gitlab repo刷新实验室2分支的文件。您将找到一个名为约束-Lab的骨架源文件。在其中为您的解决方案编写Python代码。当你准备好推动你的约束功能时，ts-实验室。请记住，你需要用这个命令来标记你的提交

```
$ git标签24011-lab2-s-约束
```

您在提交中包含的任何其他文件都将被忽略。关于使用Gitlab提交课程作业的一般说明可以在CS手册中找到。

提交的截止日期是10月31日星期一的18：00。实验室将自动离线。标记程序将从Gitlab下载您的代码，并根据参考实现进行测试。（这需要一段时间，所以预计标记至少需要两周的时间。）重要提示：请注意，骨架约束是s实验室。Py结束于

```
# Debug
if __name__ == '__main__':
    print("debug run...")
```

您应该熟悉这种构造，它允许运行源文件进行调试

以及像Python模块一样导入[1].除了所需的函数定义和任何必要的导入语句外，您在约束实验室中添加的所有代码。警察必须这样保护起来。其原因是，自动标记程序将通过导入以下代码来测试您的解决方案

```
从约束实验室导入旅行者，公共，列表，列表
```

而任何未受保护的剩余代码都会影响对函数的测试。

lhttps://docs.巨蛇org/3/库/__主要__。html