School of Data and Computer Science, Sun Yat-sen University

P03 Planning and Uncertainty

Due: 11:59pm, Saturday, Nov. 28, 2020

Contents

1 STRIPS planner

In this part, you will implement a simple STRIPS planner. The input of your planner is a PDDL domain file and a problem file in the STRIPS restriction, that is, preconditions of actions and the goal are conjunctions of atoms, and effects of actions are conjunctions of literals. The output of your planner is a sequence of actions to achieve the goal.

1. Describe with sentences the main ideas behind computing the heuristic for a state using reachability analysis from lecture notes. (10 points)
2. Implement a STRIPS planner by using A* search and the heuristic function you implemented.(20 points)
3. Explain any ideas you use to speed up the implementation. (10 points)
4. Run you planner on the 5 test cases, and report the returned plans and the running times. Analyse the experimental results. (10 points)

2 Diagnosing by Bayesian Networks

2.1 Variables and their domais

```
(1)PatientAge:['0-30','31-65','65+']
(2)CTScanResult:['Ischemic Stroke','Hemmorraghic Stroke']
(3)MRIScanResult: ['Ischemic Stroke','Hemmorraghic Stroke']
(4)StrokeType: ['Ischemic Stroke','Hemmorraghic Stroke', 'Stroke Mimic']
(5)Anticoagulants: ['Used','Not used']
(6)Mortality:['True', 'False']
(7)Disability: ['Negligible', 'Moderate', 'Severe']
```

2.2 CPTs

Note: [CTScanResult, MRIScanResult,StrokeType] means:

P(StrokeType='...' | CTScanResult='...' ∧ MRIScanResult='...')

```
(1)
[PatientAge]


['0-30', 0.10],
```

```
['31-65', 0.30],
['65+', 0.60]

(2)
[CTScanResult]

['Ischemic Stroke',0.7],
[ 'Hemmorraghic Stroke',0.3]

(3)
[MRIScanResult]

['Ischemic Stroke',0.7],
[ 'Hemmorraghic Stroke',0.3]

(4)
[Anticoagulants]

[Used',0.5],
['Not used',0.5]

(5)
[CTScanResult, MRIScanResult,StrokeType])

['Ischemic Stroke','Ischemic Stroke','Ischemic Stroke',0.8],
['Ischemic Stroke','Hemmorraghic Stroke','Ischemic Stroke',0.5],
[ 'Hemmorraghic Stroke','Ischemic Stroke','Ischemic Stroke',0.5],
[ 'Hemmorraghic Stroke','Hemmorraghic Stroke','Ischemic Stroke',0],

['Ischemic Stroke','Ischemic Stroke','Hemmorraghic Stroke',0],
['Ischemic Stroke','Hemmorraghic Stroke','Hemmorraghic Stroke',0.4],
[ 'Hemmorraghic Stroke','Ischemic Stroke','Hemmorraghic Stroke',0.4],
[ 'Hemmorraghic Stroke','Hemmorraghic Stroke','Hemmorraghic Stroke',0.9],
```

```
['Ischemic Stroke','Ischemic Stroke','Stroke Mimic',0.2],
['Ischemic Stroke','Hemmorraghic Stroke','Stroke Mimic',0.1],
[ 'Hemmorraghic Stroke','Ischemic Stroke','Stroke Mimic',0.1],
[ 'Hemmorraghic Stroke','Hemmorraghic Stroke','Stroke Mimic',0.1],

(6)
[StrokeType, Anticoagulants, Mortality]

['Ischemic Stroke', 'Used', 'False',0.28],
['Hemmorraghic Stroke', 'Used', 'False',0.99],
['Stroke Mimic', 'Used', 'False',0.1],
['Ischemic Stroke','Not used', 'False',0.56],
['Hemmorraghic Stroke', 'Not used', 'False',0.58],
['Stroke Mimic', 'Not used', 'False',0.05],

['Ischemic Stroke',  'Used' ,'True',0.72],
['Hemmorraghic Stroke', 'Used', 'True',0.01],
['Stroke Mimic', 'Used', 'True',0.9],
['Ischemic Stroke',  'Not used' ,'True',0.44],
['Hemmorraghic Stroke', 'Not used', 'True',0.42 ],
['Stroke Mimic', 'Not used', 'True',0.95]

(7)
[StrokeType, PatientAge, Disability]

['Ischemic Stroke',   '0-30','Negligible', 0.80],
['Hemmorraghic Stroke', '0-30','Negligible', 0.70],
['Stroke Mimic',        '0-30', 'Negligible',0.9],
['Ischemic Stroke',     '31-65','Negligible', 0.60],
['Hemmorraghic Stroke', '31-65','Negligible', 0.50],
['Stroke Mimic',        '31-65', 'Negligible',0.4],
['Ischemic Stroke',     '65+'  , 'Negligible',0.30],
```

```
['Hemmorraghic Stroke', '65+'   , 'Negligible',0.20],
['Stroke Mimic',        '65+'   , 'Negligible',0.1],

['Ischemic Stroke',     '0-30'  ,'Moderate',0.1],
['Hemmorraghic Stroke', '0-30'  ,'Moderate',0.2],
['Stroke Mimic',        '0-30'  ,'Moderate',0.05],
['Ischemic Stroke',     '31-65','Moderate',0.3],
['Hemmorraghic Stroke', '31-65','Moderate',0.4],
['Stroke Mimic',        '31-65','Moderate',0.3],
['Ischemic Stroke',     '65+'   ,'Moderate',0.4],
['Hemmorraghic Stroke', '65+'   ,'Moderate',0.2],
['Stroke Mimic',        '65+'   ,'Moderate',0.1],

['Ischemic Stroke',     '0-30'  ,'Severe',0.1],
['Hemmorraghic Stroke', '0-30'  ,'Severe',0.1],
['Stroke Mimic',        '0-30'  ,'Severe',0.05],
['Ischemic Stroke',     '31-65','Severe',0.1],
['Hemmorraghic Stroke', '31-65','Severe',0.1],
['Stroke Mimic',        '31-65','Severe',0.3],
['Ischemic Stroke',     '65+'   ,'Severe',0.3],
['Hemmorraghic Stroke', '65+'   ,'Severe',0.6],
['Stroke Mimic',        '65+'   ,'Severe',0.8]
```

2.3 Tasks

1. Briefly describe with sentences the main ideas of the VE algorithm. (10 points)

   VE algorithm 在计算给定证据下某些变量的结果的时候，我们要应用链式法则和全概公式来计算，其中链式法则将表达式拆解成乘积的形式，而全概公式将表达式拆解成求合的形式。再通过限制住某些元素得到取值来得到我们最终想要的结果。VE 算法通过 restrict(限制变量的取值)，multiply(进行链式法则实现概率之间的相乘)，sumout(全概公式求合) 三个操作，将无关变量通过相乘求合等操作消去，留下在证据下我们需要查询的变量的概率。

2. Implement the VE algorithm (C++ or Python) to calculate the following probability values: (10 points)

(a) p1 = P(Mortality='True' ∧ CTScanResult='Ischemic Stroke' | PatientAge='31-65')

(b) p2 = P(Disability='Moderate' ∧ CTScanResult='Hemmorraghic Stroke' | PatientAge='65+' ∧ MRIScanResult='Hemmorraghic Stroke')

(c) p3 = P(StrokeType='Hemmorraghic Stroke' | PatientAge='65+' ∧ CTScanResult='Hemmorraghic Stroke' ∧ MRIScanResult='Ischemic Stroke')

(d) p4 = P(Anticoagulants='Used' | PatientAge='31-65')

(e) p5 = P(Disability='Negligible')

实现思路 有了实验八的基础，这一部分的代码只需要在实验八的基础上稍作修改即可。首先我们添加了一个名为 valueMap 的字典，里面讲每一个变量的可取值映射为 0，1，2……这样做可以方便后续的数值选取。其次我们将原本传入的列表类型 queryVariables 换成了字典类型变量，并在其中记录每一个查询变量的查询取值，这样在输出结果的时候，只需要输出查询的取值结果而不需要输出全部的结果，相应地在 printInf 函数中做出一些格式化的调整。下面我们分别给出我们的程序计算的概率结果数值和使用 pomegranate 进行计算得到的结果，经过比对可以发现二者在误差可允许的范围内结果一致。会出现微小的误差是由变量消除的顺序导致的，因为消除的顺序不同所以计算的结果运算的顺序也不同。

```
our code's result:
RESULT:
0.41639499999999996
max eliminate width: 6
Elimination order :  ['StrokeType', 'Disability', 'MRIScanResult', 'Anticoagulants']

RESULT:
0.057
max eliminate width: 6
Elimination order :  ['StrokeType', 'Anticoagulants', 'Mortality']

RESULT:
0.4
max eliminate width: 3
Elimination order :  ['Disability', 'Mortality', 'Anticoagulants']

RESULT:
0.5
max eliminate width: 6
Elimination order :  ['StrokeType', 'Disability', 'Mortality', 'CTScanResult', 'MRIScanResult']

RESULT:
0.38977
max eliminate width: 6
Elimination order :  ['StrokeType', 'Mortality', 'PatientAge', 'Anticoagulants', 'CTScanResult', 'MRIScanResult']
```

Figure 1: our code's result

```
pomegranate result:
p1 =  0.41639499999999974
p2 =  0.05700000000000005
p3 =  0.3999999999999996
p4 =  0.5
p5 =  0.38977
```

Figure 2: pomegranate result

3. Implement an algorithm to select a good order of variable elimination. (10 points)

PolyTree 根据给出的概率表，我们可以知道我们的这个问题是一个多叉树，变量之间的相互作用关系如下所示。对于贝叶斯网络中的多叉树来说，在每一步我们都消掉一个单一连接的节点 (因为是多叉树，所以一定能保证在每一步消去的时候都有单一连接的节点)，那么树的因子的大小将一直不会增加，也就是说，初始时因子中变量数目最大的一个即为树的宽度。

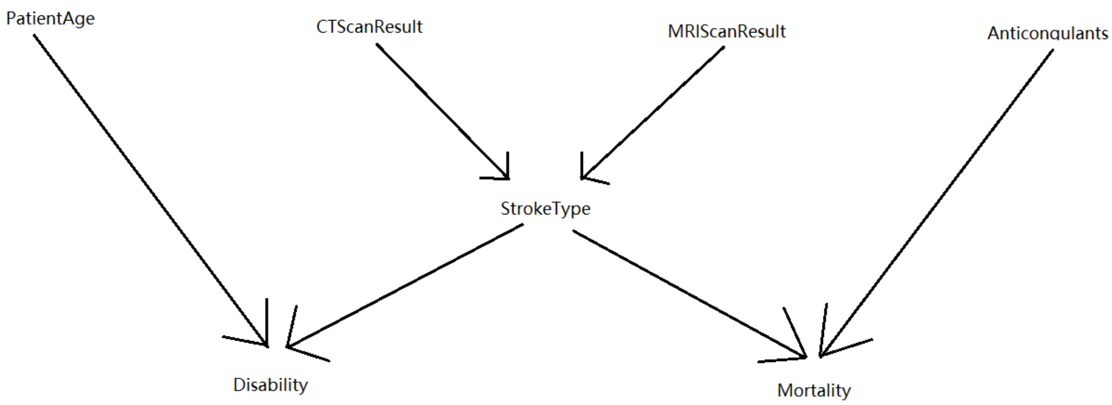

Figure 3: the polytree of the problem

Min-Fill Heuristic 我们使用的启发式策略是每次都消除产生最小因子的变量。这一算法在多树问题上是线性时间复杂度的。在多数问题中我们只需要在待消除变量中寻找单一连接的节点，然后在单一连接的节点中因子大小最小的变量先进行消除即可。这体现在我们的数据结构中只需要根据变量表的大小就可以判断。我们利用一个优先队列来帮助简化编程，具体代码如下所示。

```python
def orderedVariables(factorList, orderedListOfHiddenVariables: list):
    q = queue.PriorityQueue()
    for v in orderedListOfHiddenVariables:
        for f in factorList:
            if v == f.name:
                q.put((len(f.varList), v))
                break
    orderedListOfHiddenVariables.clear()
    while q.qsize():
        orderedListOfHiddenVariables.append(q.get()[1])
```

4. Compare the running times of the VE algorithm for different orders of variable elimination, and fill out the following table: For test cases p4 and p5, for each of the order selected by your algorithm and 5 other orders, report the elimination with, and the total running time of the VE algorithm. For each case, the first order of elimination should be the one chosen by your algorithm. Analyze the results. (20 points)

由于问题模型很简单，即使是使用最差的消除顺序运行时间也很短，为了使结果更加明显，我们将每个消除顺序都运行 5000 次后比对运行时间的差异。在测试时，注释掉程序的输出部分，将待消除变量保存到一个列表中，之后调用我们实现的 orderedVariables 函数寻找最优消除顺序，之后调用 random.shuffle(eliminatelist) 函数来对消除顺序进行随机重排后进行消除宽度和时间上的测试。

Test case	Elimination order	Elimination width	Total time
p4	3	['CTScanResult', 'MRIScanResult', 'Disability', 'Mortality', 'StrokeType']	1.23 s
p4	4	['MRIScanResult', 'Mortality', 'StrokeType', 'Disability', 'CTScanResult']	2.51 s
p4	4	['CTScanResult', 'MRIScanResult', 'StrokeType', 'Disability', 'Mortality']	2.84 s
p4	6	['StrokeType', 'Mortality', 'MRIScanResult', 'CTScanResult', 'Disability']	7.70 s
p4	5	['CTScanResult', 'StrokeType', 'MRIScanResult', 'Disability', 'Mortality']	4.74 s
p4	6	['StrokeType', 'Mortality', 'Disability', 'CTScanResult', 'MRIScanResult']	7.53 s
p5	3	['Anticoagulants', 'CTScanResult', 'MRIScanResult', 'PatientAge', 'Mortality', 'StrokeType']	2.10 s
p5	4	['MRIScanResult', 'PatientAge', 'StrokeType', 'Anticoagulants', 'Mortality', 'CTScanResult']	4.94 s
p5	5	['CTScanResult', 'StrokeType', 'PatientAge', 'Anticoagulants', 'Mortality', 'MRIScanResult']	12.89 s
p5	6	['StrokeType', 'Mortality', 'MRIScanResult', 'Anticoagulants', 'PatientAge', 'CTScanResult']	20.77s
p5	4	['Anticoagulants', 'PatientAge', 'StrokeType', 'MRIScanResult', 'Mortality', 'CTScanResult']	4.40 s
p5	6	['StrokeType', 'Anticoagulants', 'PatientAge', 'Mortality', 'CTScanResult', 'MRIScanResult']	22.61 s

```
count p4 = P(Anticoagulants='Used' | PatientAge='31-65')

our order of variable elimination:  ['CTScanResult', 'MRIScanResult', 'Disability', 'Mortality', 'StrokeType']
running time =  1.2300028800964355  s
Elimination width =  3
random order of variable elimination:  ['MRIScanResult', 'Mortality', 'StrokeType', 'Disability', 'CTScanResult']
running time =  2.509963274002075  s
Elimination width =  4
random order of variable elimination:  ['CTScanResult', 'MRIScanResult', 'StrokeType', 'Disability', 'Mortality']
running time =  2.8402833938598633  s
Elimination width =  4
random order of variable elimination:  ['StrokeType', 'Mortality', 'MRIScanResult', 'CTScanResult', 'Disability']
running time =  7.700991153717041  s
Elimination width =  6
random order of variable elimination:  ['CTScanResult', 'StrokeType', 'MRIScanResult', 'Disability', 'Mortality']
running time =  4.739139795303345  s
Elimination width =  5
random order of variable elimination:  ['StrokeType', 'Mortality', 'Disability', 'CTScanResult', 'MRIScanResult']
running time =  7.532074213027954  s
Elimination width =  6
```

Figure 4: p4 result

```
count p5 = P(Disability='Negligible')

our order of variable elimination:  ['Anticoagulants', 'CTScanResult', 'MRIScanResult', 'PatientAge', 'Mortality', 'StrokeType']
running time =  2.107309579849243  s
Elimination width =  3
random order of variable elimination:  ['MRIScanResult', 'PatientAge', 'StrokeType', 'Anticoagulants', 'Mortality', 'CTScanResult']
running time =  4.941882610321045  s
Elimination width =  4
random order of variable elimination:  ['CTScanResult', 'StrokeType', 'PatientAge', 'Anticoagulants', 'Mortality', 'MRIScanResult']
running time =  12.89268684387207  s
Elimination width =  5
random order of variable elimination:  ['StrokeType', 'Mortality', 'MRIScanResult', 'Anticoagulants', 'PatientAge', 'CTScanResult']
running time =  20.77104425430298  s
Elimination width =  6
random order of variable elimination:  ['Anticoagulants', 'PatientAge', 'StrokeType', 'MRIScanResult', 'Mortality', 'CTScanResult']
running time =  4.402190208435059  s
Elimination width =  4
random order of variable elimination:  ['StrokeType', 'Anticoagulants', 'PatientAge', 'Mortality', 'CTScanResult', 'MRIScanResult']
running time =  22.612092971801758  s
Elimination width =  6
```

Figure 5: p5 result

结果截图展示

结果分析 可以看出不同的变量消除顺序对于结果的影响是巨大的，而我们的算法找出的消除顺序无论是在消除宽度上还是运行时间上都有很好的效果。

3 Due: 11:59pm, Saturday, Nov. 28, 2020

Please hand in a file named P03_YourNumber.pdf, and send it to ai_2020@foxmail.com