CSP ^{赵耀}

CSP的基本算法

► CSP的基本算法框架为 回溯法

```
def backtracking_search(csp,
                               select_unassigned_variable=first_unassigned_variable,
251
                               order_domain_values=unordered_domain_values,
252
                               inference=no_inference):
253
          """[Figure 6.5]"""
254
255
          def backtrack(assignment):
256 ▼
              if len(assignment) == len(csp.variables):
257 ▼
258
                   return assignment
              var = select_unassigned_variable(assignment, csp)
259
              for value in order_domain_values(var, assignment, csp):
260 ₩
261 ▼
                   if 0 == csp.nconflicts(var, value, assignment):
                       csp.assign(var, value, assignment)
262
                       removals = csp.suppose(var, value)
263
                       if inference(csp, var, value, assignment, removals):
264 ₩
                           result = backtrack(assignment)
265
                           if result is not None:
266 ▼
267
                               return result
                       csp.restore(removals)
268
269
              csp.unassign(var, assignment)
270
              return None
271
272
          result = backtrack({})
273
          assert result is None or csp.goal_test(result)
          return result
274
```

CSP问题的表达

- ▶ 由一个变量集合和一个约束集合组成。
- ▶ 问题的状态:对一些或全部变量的赋值
- ▶ 状态切换:对一个变量进行赋值或者取消赋值
- ▶ Goal: 满足所有约束的完全赋值(或者使目标函数最大化)

CSP回溯流程

- ▶ 如果达到完全赋值,则返回该完全赋值
- ▶ 否则,选择一个未赋值的变量X_i
- ▶ 依次从该变量的值域中选值v赋值给Xi
- > 判断该赋值加入后是否有效,如果有效,继续递归,处理下一个变量
- ▶ 如果无效,取消赋值,尝试另一个值
- ▶ 找完所有可能的值赋值给Xi均失败,返回失败

CSP优化

- ▶ 如何选择下一个变量(最多约束变量启发式,也叫最少剩余值启发式)
- ▶ 如何缩小值域以及如何在一个给定的值域中选值(最少约束值启发式)
- ▶ 如何提前预判,在调用下一层之前就知道下层调用有没有必要做(根据约束进行推 论,inference函数)

对应到code

```
def backtracking_search(csp,
250
                               select_unassigned_variable=first_unassigned_variable,
251
                               order_domain_values=unordered_domain_values,
252
                               inference=no_inference):
253
          """[Figure 6.5]"""
254
255
          def backtrack(assignment):
256 ▼
              if len(assignment) == len(csp.variables):
257 ▼
258
                  return assignment
              var = select_unassigned_variable(assignment, csp)
259
              for value in order_domain_values(var, assignment, csp):
260 ₩
                  if 0 == csp.nconflicts(var, vacue, assignment):
261 ▼
                       csp.assign(var, value, assignment)
262
                       removals = csp.suppose(var, value)
263
                      if interence(csp, var, value, assignment, removals):
264 ₩
                           result = backtrack(assignment)
                           if result is not None:
266 ▼
267
                               return result
                       csp.restore(removals)
268
269
              csp.unassign(var, assignment)
270
              return None
271
272
          result = backtrack({})
273
          assert result is None or csp.goal_test(result)
274
          return result
```

优化函数

- ▶ select_unassigned_variable,根据具体问题设计Minimum-remaining-values heuristic,优先选择最小剩余值的变量
- ▶ order_domain_values, 根据具体问题设计Least-constraining-values heuristic, 约束最少的值排在前面
- ▶ inference,根据具体问题设计Maintain arc consistency, node consistency或者forward checking and so on
- ▶ 根本目的,在不影响结果的前提下,最大可能的剪枝

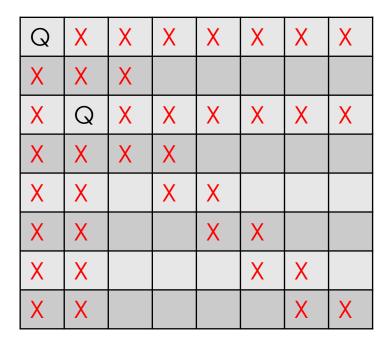
什么叫约束传播

▶ 使用约束去降低一个变量的合法的取值范围,因此又引起另一个变量的合法取值范围变小,像多米诺骨牌效应一般,使得变量的值域越来越小。

八皇后问题(一)

Q	Χ	Χ	Χ	Χ	Χ	Χ	Χ
Χ	Χ						
X X X X X		Χ					
Χ			Χ				
Χ				Χ			
Χ					Χ		
						Χ	
X							X

- 1、将一个皇后放入到一个方格里
- 2、移去所有可能攻击到的方格
- 3、刚开始有8个可以放置的位置



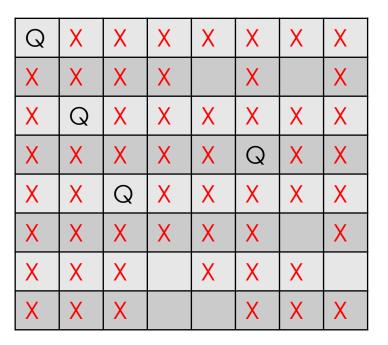
4+4+4+4+5

- 1、将另一个皇后放置在有着最小剩余值的行或列上
- 2、移去所有可能攻击到的方格
- 3、第二个皇后只有6个可以放置的位置

八皇后问题 (二)

Q	Χ	Χ	X	X	X	X	Χ
X	X	Χ			Χ		
Χ	Ø	Χ	Χ	Χ	Χ	Χ	X
X	X	X	Χ				
Χ	Χ	Q	Χ	Χ	Χ	Χ	Χ
Χ	Χ	Χ	Χ	Χ	Χ		
Χ	Χ	Χ		Χ	Χ	Χ	
Χ	Χ	Χ			Χ	Χ	X

- 1、将第三个皇后继续放置在有着最小剩余值的行或列上
- 2、移去所有可能攻击到的方格
- 3 第三个阜后只有4个可以放置的位置

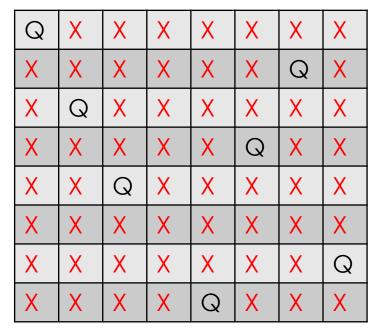


- 1、将第四个皇后继续放置在有着最小剩余值的行或列上
- 2、移去所有可能攻击到的方格
- 3 第四个阜后具有1个可以放置的位置

八皇后问题(三)

Q	Χ	Χ	Χ	Χ	Χ	Χ	X
Χ	Χ	Χ	Χ		Χ		Χ
Χ	Q	Χ	Χ	Χ	Χ	Χ	Χ
Χ	Χ	Χ	Χ	Χ	Q	Χ	X
X	Χ	Q	Χ	Χ	Χ	Χ	X
Χ	Χ	Χ	X	Χ	Χ	Χ	X
Χ	X	Χ		X	X	X	Q
Χ	X	Χ			Χ	Χ	X

- 1、将第五个皇后继续放置在有着最小剩余值的行或列上
- 2、移去所有可能攻击到的方格
- 3 第五个阜后只有1个可以放置的位置



- 1、将第六个皇后继续放 置在有着
- 最小剩余值的行或列上
- 2、移去所有可能攻击到 的方格
- 3、第六个皇后只有1个 可以放置的位置
- 4、第七个皇后再放上去, 移去所有可能攻击的方 格后发现,
- 第八位皇后无法再放上去,值域为空。此时应该再回溯。大家可以看到由于值域的约束,可以很容易跳回到第三位皇后处开始回溯

Thank You!