CSP Review

YAO ZHAO

CSPs definition

- A constraint satisfaction problem consists of **three elements**:
  - A set of **variables**, X = {X1, X2, · · ·Xn}
  - A set of **domains** for each variable: D = {D1, D2, · · ·Dn}
  - A set of **constraints** C that specify allowable combinations of values.

Basic Algorithm for CSP

- Backtracking Search(BTS)

```
250 def backtracking_search(csp,
251                        select_unassigned_variable=first_unassigned_variable,
252                        order_domain_values=unordered_domain_values,
253                        inference=no_inference):
254     """[Figure 6.5]"""
255
256     def backtrack(assignment):
257         if len(assignment) == len(csp.variables):
258             return assignment
259         var = select_unassigned_variable(assignment, csp)
260         for value in order_domain_values(var, assignment, csp):
261             if 0 == csp.nconflicts(var, value, assignment):
262                 csp.assign(var, value, assignment)
263                 removals = csp.suppose(var, value)
264                 if inference(csp, var, value, assignment, removals):
265                     result = backtrack(assignment)
266                     if result is not None:
267                         return result
268                 csp.restore(removals)
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
```

Backtracking Search(BTS)

Backtracking search is simple and systematic:

- Start with no assignment Pick a variable and assign it.
- Repeat.
- If we hit a dead end, backtrack up the search tree until we find a variable that can have its value set to something different
- Backing all the way up the tree to the root and finding no more values means No solution.

DFS that chooses one variable at a time

How to improving BTS

- **Which variable should be assigned next?**
- **In what order should its values be tried?**
- **Can we detect inevitable failure early?**

Reference code

```
250  def backtracking_search(csp,
251                          select_unassigned_variable=first_unassigned_variable,
252                          order_domain_values=unordered_domain_values,
253                          inference=no_inference):
254      """[Figure 6.5]"""
255
256      def backtrack(assignment):
257          if len(assignment) == len(csp.variables):
258              return assignment
259          var = select_unassigned_variable(assignment, csp)
260          for value in order_domain_values(var, assignment, csp):
261              if 0 == csp.nconflicts(var, value, assignment):
262                  csp.assign(var, value, assignment)
263                  removals = csp.suppose(var, value)
264                  if inference(csp, var, value, assignment, removals):
265                      result = backtrack(assignment)
266                      if result is not None:
267                          return result
268                  csp.restore(removals)
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
```

Improving Backtracking Efficiency

- **Which variable should be assigned next?**

Minimum Remaining Values heuristic : Choose the variable with the fewest legal values in its domain.)

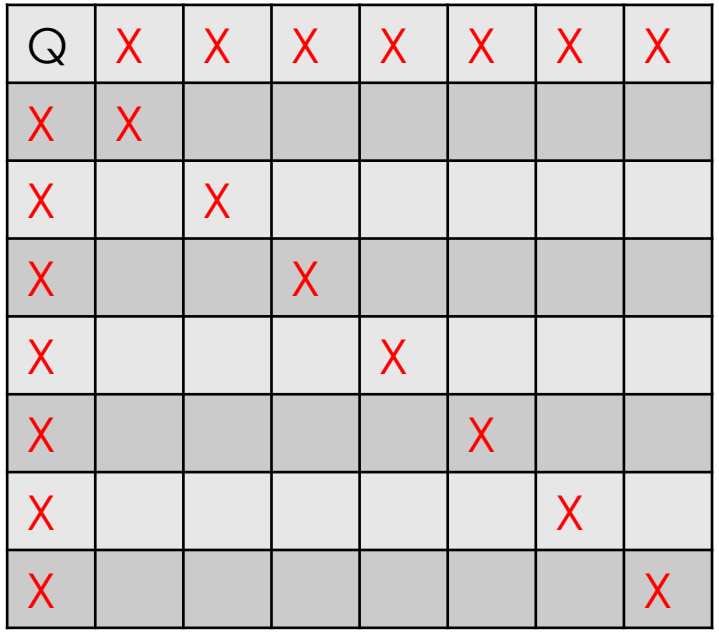
- **In what order should its values be tried?**

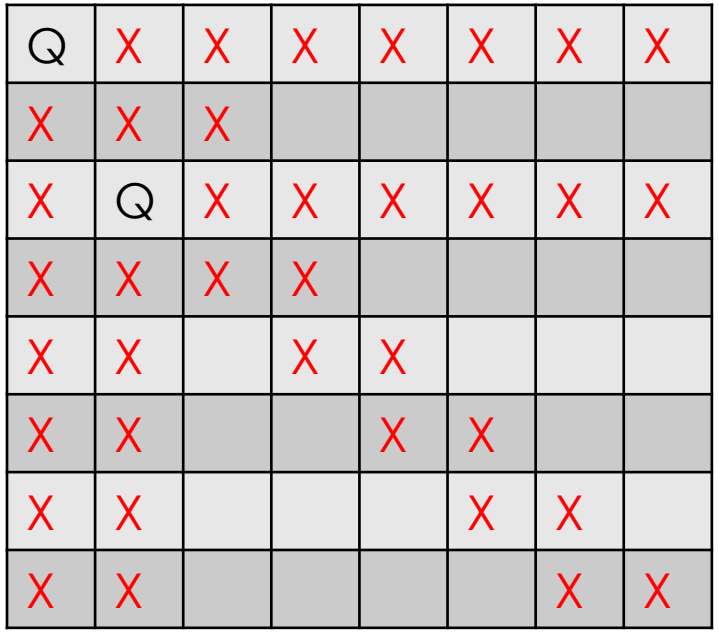
Least constraining value heuristic: Given a variable, choose the least constraining value: the one that rules out the fewest values in the remaining variables)

- **Can we detect inevitable failure early?**

Inference: FC and using constraint propagation, e.g., arc consistency test.

Eight-Queen Problem(1)

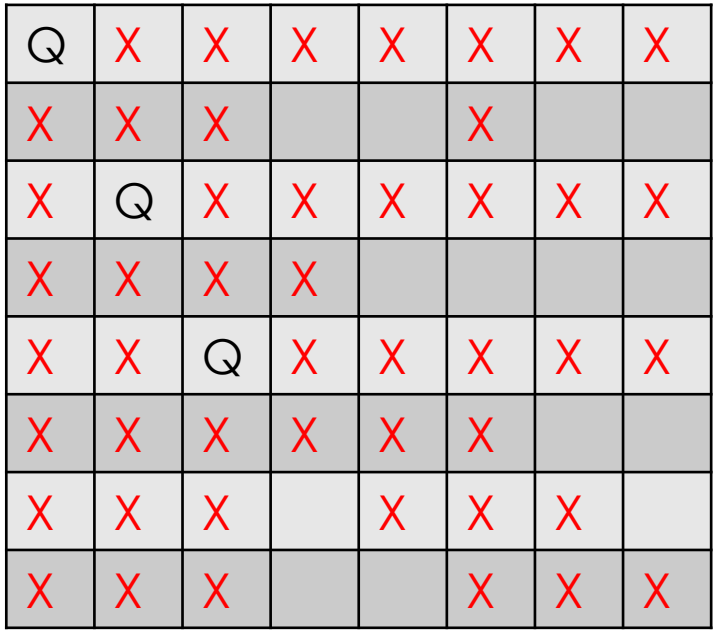


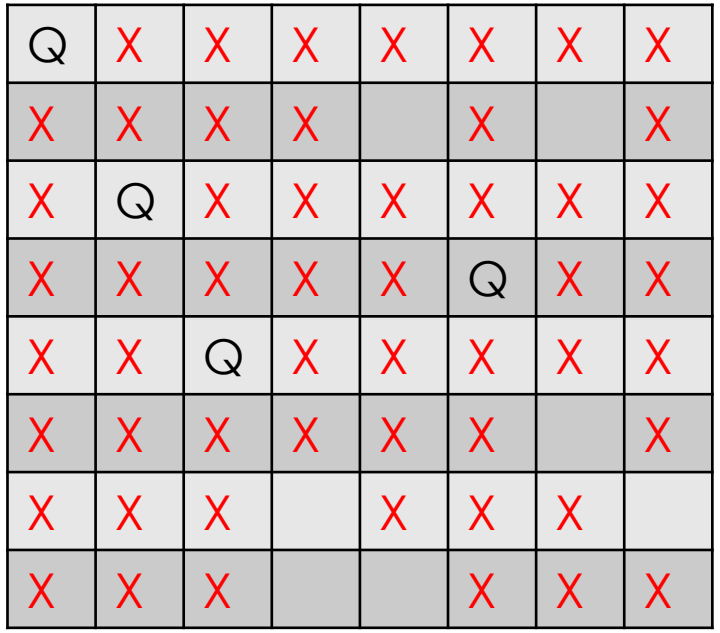


4+4+4+4+4+5

- Assign a queen to a position and remove all assignments inconsistent with it
- We have 8 possible choices for the 1st queen
- We have 6 possible choices for the 2nd queen

Eight-Queen Problem(2)





- Assign a queen to a position and remove all assignments inconsistent with it
- We choose the variable with minimal possible values: principle of Minimum remaining values
- We have 4 possible choices for the 3rd queen
- We have 1 possible choice for the 4th queen

Eight-Queen Problem(3)

Q	X	X	X	X	X	X	X
X	X	X	X		X		X
X	Q	X	X	X	X	X	X
X	X	X	X	X	Q	X	X
X	X	Q	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X		X	X	X	Q
X	X	X			X	X	X

Q	X	X	X	X	X	X	X
X	X	X	X	X	X	Q	X
X	Q	X	X	X	X	X	X
X	X	X	X	X	Q	X	X
X	X	Q	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	Q
X	X	X	X	Q	X	X	X

- Assign a queen to a position and remove all assignments inconsistent with it
- We choose the variable with minimal possible values: Minimum remaining values
- We have 1 possible choice for the 5th queen
- We have 1 possible choice for the 6th queen
- We assign the position for the 7th queen and find no inconsistent for the 8th queen
- backtrack to the 3rd queen