# CONSTRAINT SATISFACTION PROBLEMS – PART III

Chapter 6

# Outline

- Constraint Satisfaction Problems (CSP)
- Backtracking search for CSPs
- Local search for CSPs

# Standard search formulation

Let's start with the straightforward approach, then improve it

- **Initial state**: the **empty assignment** { }
- **Successor function**: assign a value to an unassigned variable that does not conflict with **current assignment**
  - → **fail** if no legal assignments
- **State**: partial assignment
- **Goal test**: the current assignment is complete

- This is the same for all CSPs
- Every solution appears at depth n with n variables

  → use **depth-first search**

# Standard search formulation

**Branching factor:** maximum number of successors of a node

We could use **depth-first search**

- But for a CSP with $n$ variables of domain size $d$

- **The branching factor** $b$ is
  - at 1st level: $nd$ because any of $d$ values can be assigned to any of $n$ variables
  - at 2nd level: $(n-1)d$
  - …
  - at n-th level: $d$

- We generate **a tree** with **$n! \cdot d^n$ leaves,**
  **even if** only **$d^n$** possible **complete assignments**

# Backtracking search

- In CSPs, **variable assignments** are **commutative**, i.e.,

  {WA = red then NT = green } same as

  { NT = green then WA = red }

- Thus we need only consider **a *single* variable at each level** in the search tree → b=d → the number of **leaves** is **d^n**

- **Backtracking search = Depth-first search** for CSPs with single-variable assignments

A depth-first search that
- chooses values for SEARCH one variable at a time and
- backtracks when a variable has no legal values left to assign.

# Backtracking search

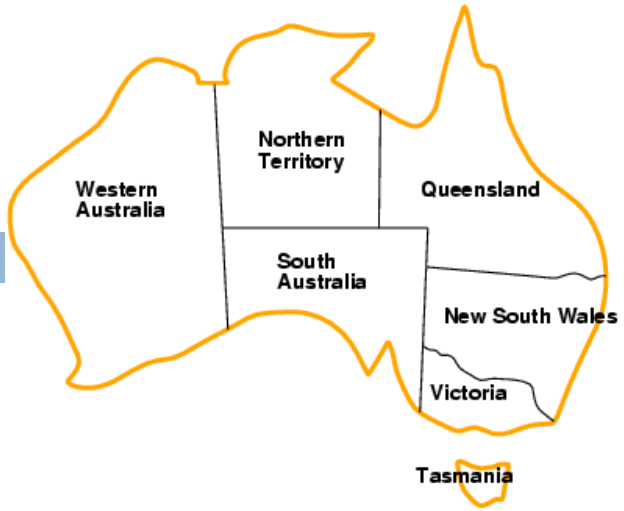
```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return BACKTRACK( { }, csp )

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment then
          add  {var = value} to assignment
          result ← BACKTRACK(assignment, csp)
          if result ≠ failure then return result
          remove  {var = value} from assignment
  return failure
```

# Review: Map-Coloring

## CSP formulation

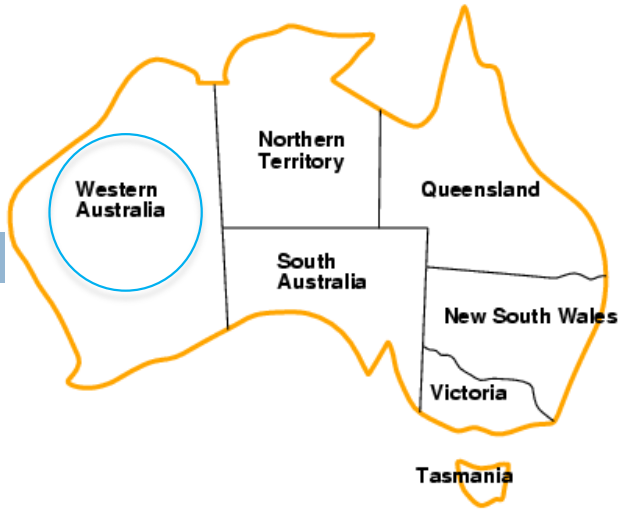

- Variables *WA, NT, Q, NSW, V, SA, T*
- Domains $D_i$ = {red, green, blue}
- Constraints: **adjacent** regions must have **different colors**

# Backtracking example



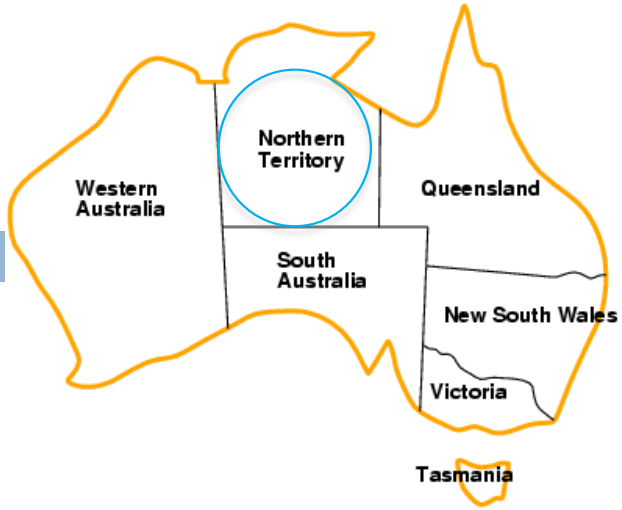

Part of the search tree

# Backtracking example

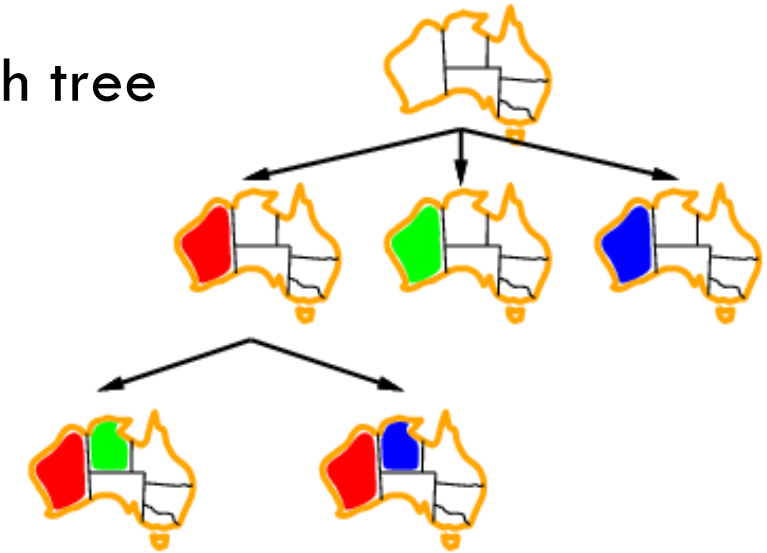



Part of the search tree

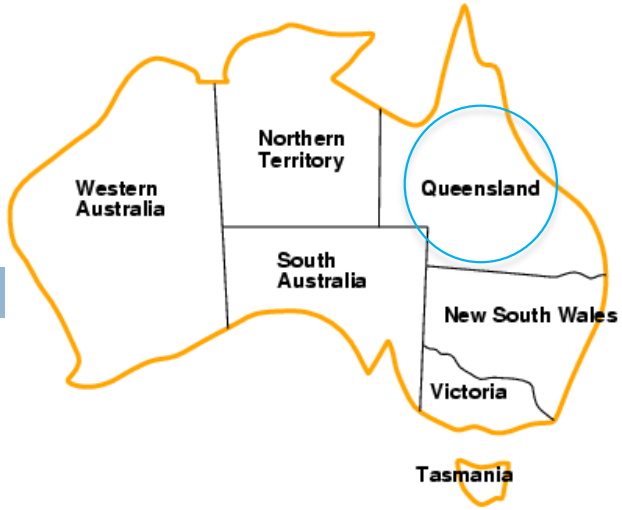
# Backtracking example



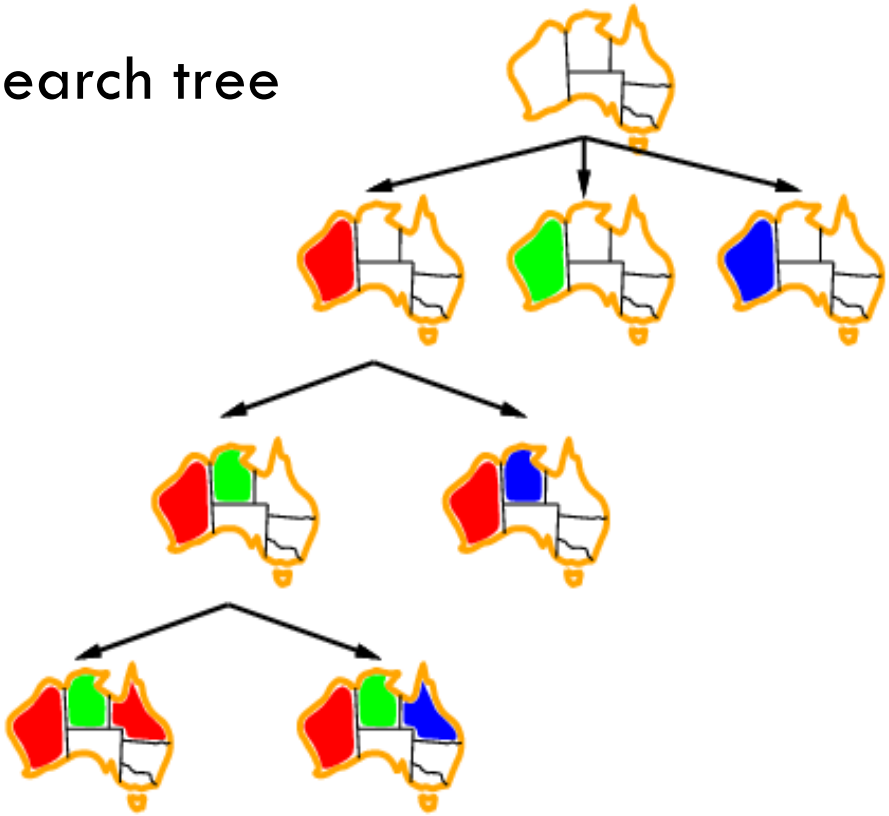

Part of the search tree



# Backtracking example

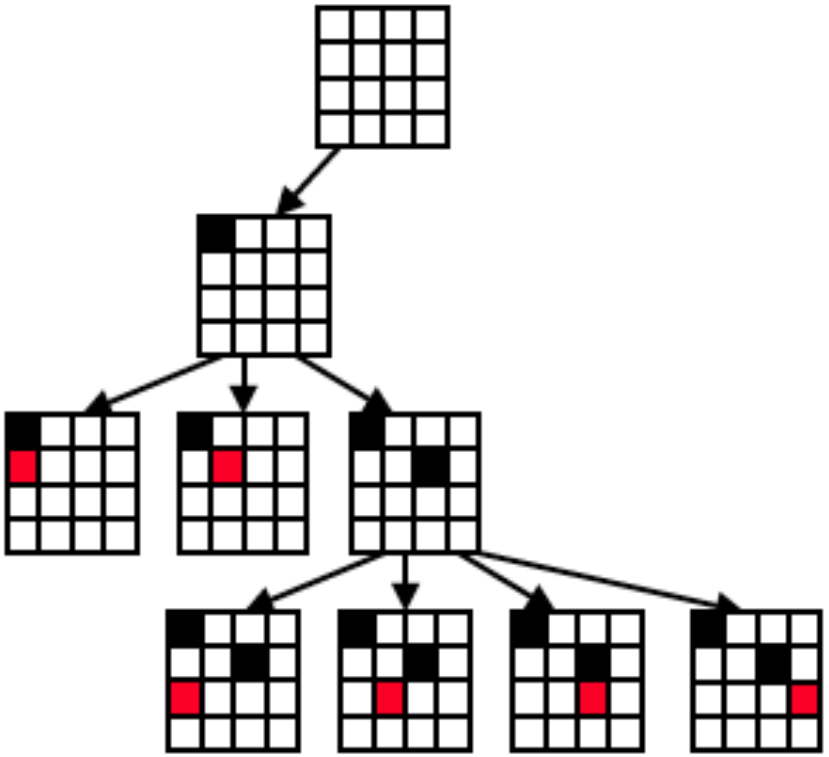



Part of the search tree



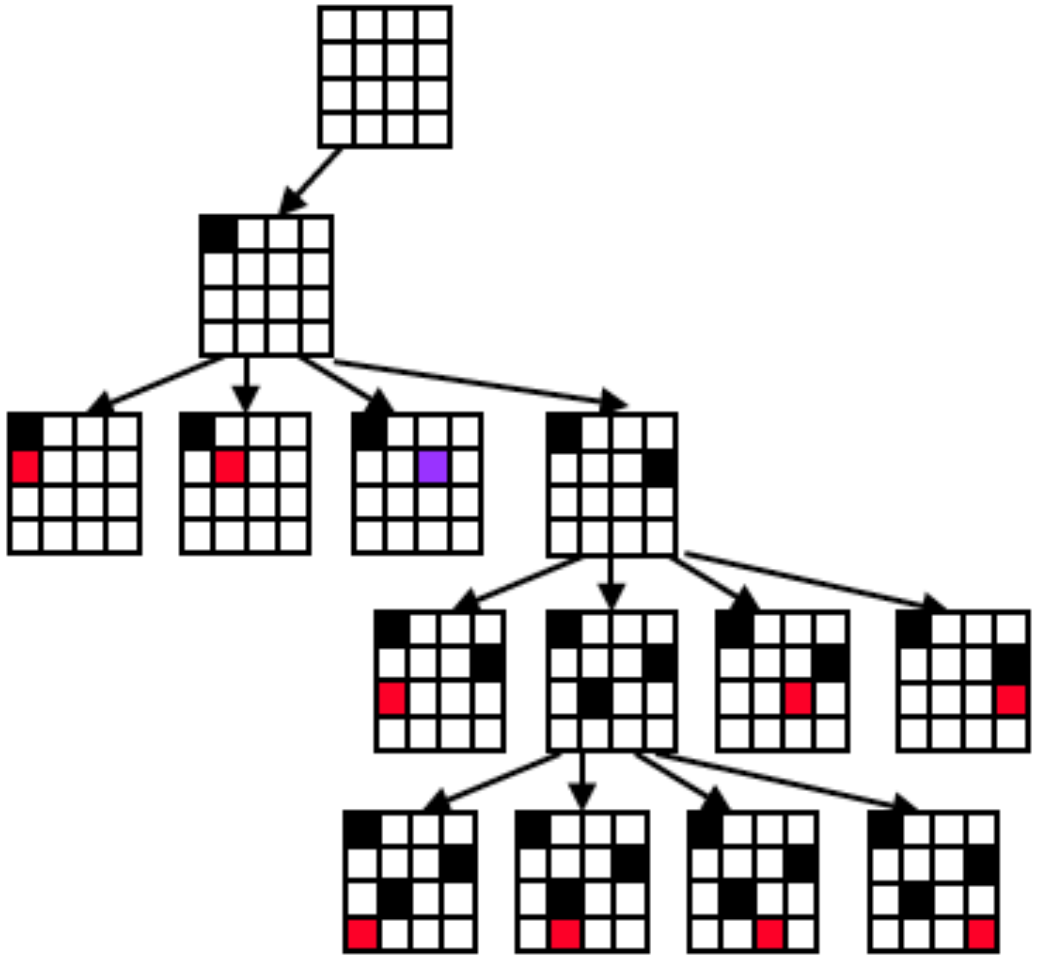
# Backtracking example

- 4X4 Queens



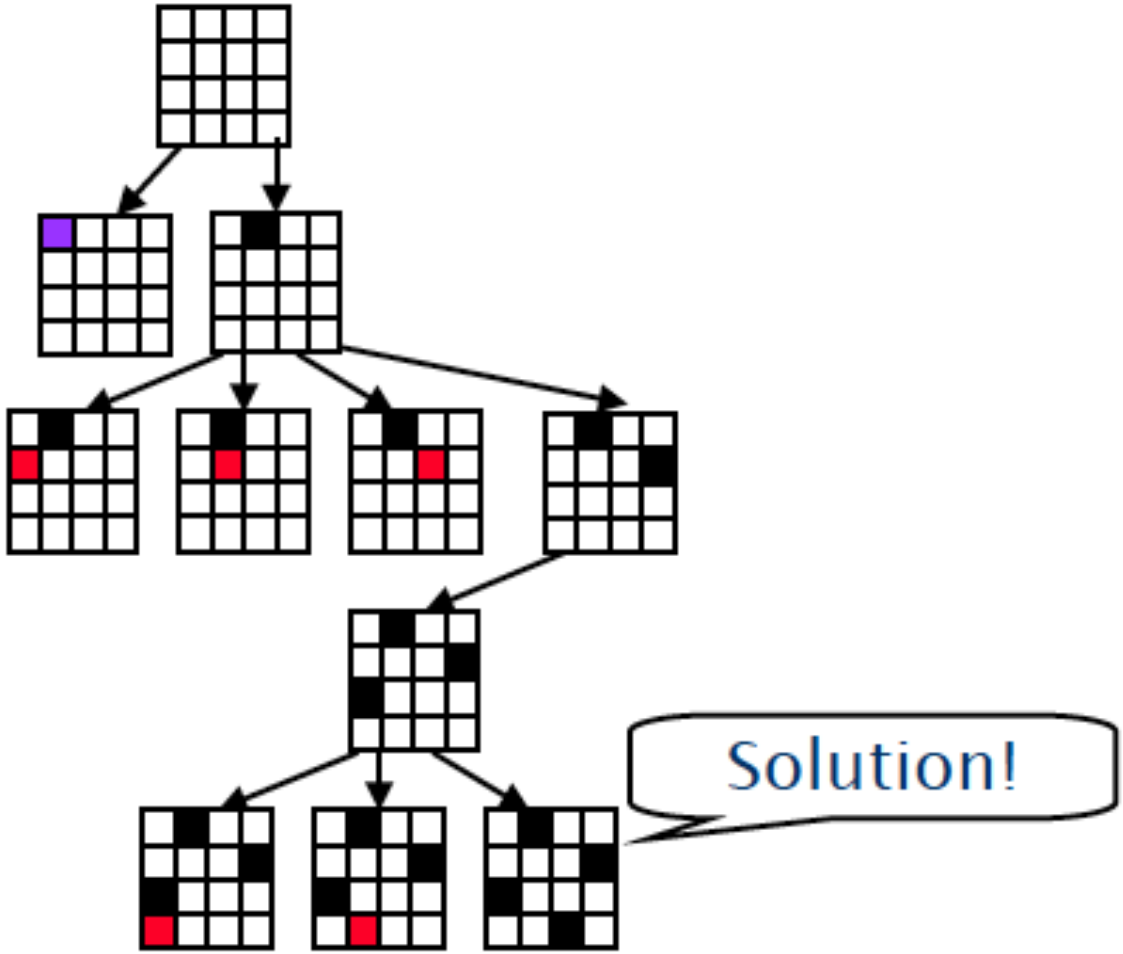
# Backtracking example

- 4X4 Queens



# Backtracking example

- 4X4 Queens



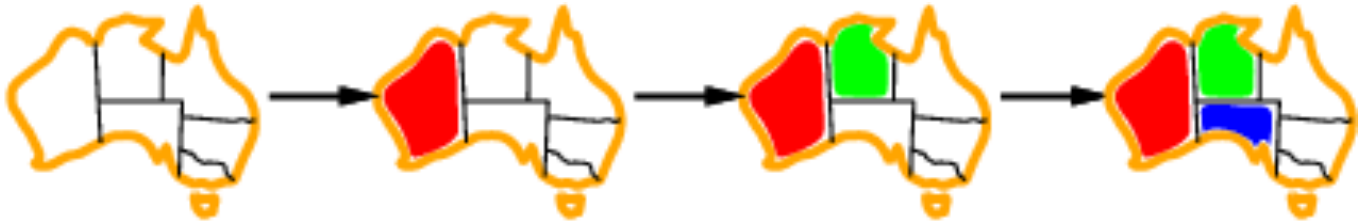
# Improving backtracking efficiency

- General-purpose methods can give huge gains in speed:
  - Which variable should be assigned next?
  - In what order should its values be tried?

# Most constrained variable

- **Most constrained variable**: **FIRST FAIL**

  chooses the variable with the fewest legal values

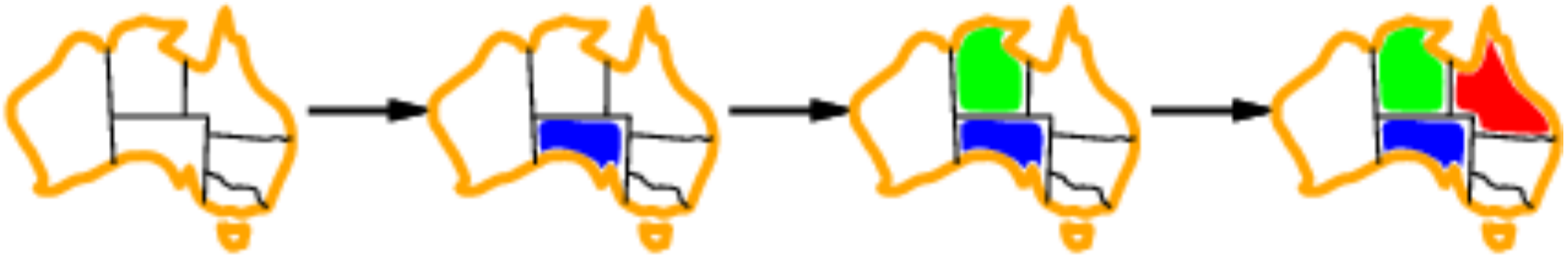


- a.k.a. minimum remaining values (MRV) heuristic

# Most constraining variable

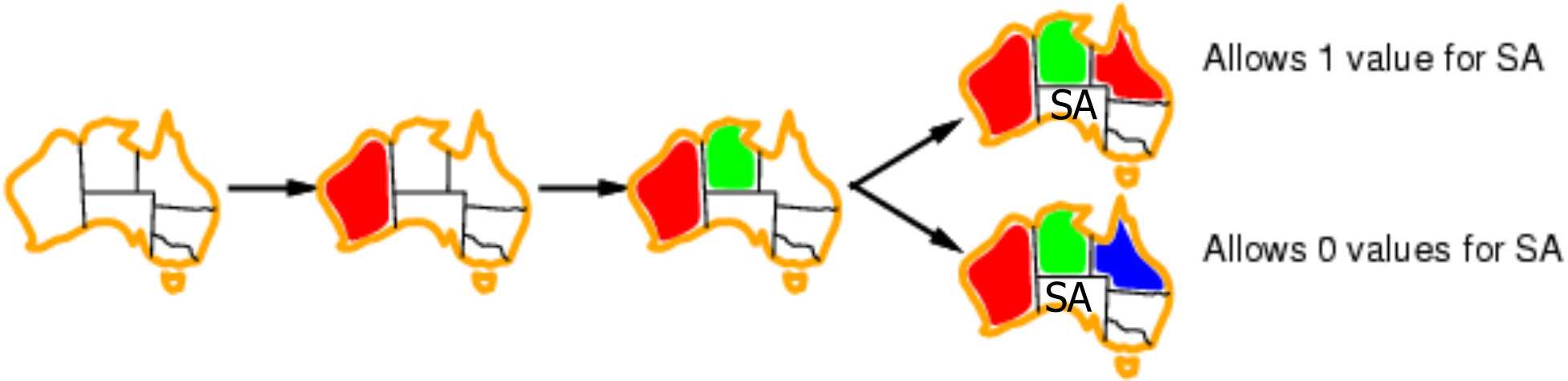

- **Tie-breaker** among **most constrained variables**
- Most constraining variable (aka, **degree heuristic**):

  chooses the variable involved in the most constraints with unassigned variables



# Least constraining **value**

- Given a variable, chooses the **least constraining value**:
  **SUCCEED FIRST**
  - the one that rules out the fewest values in the remaining variables



# Value ordering

- The ordering of values **does not matter** if
  - **all solutions needed**
  - **no solution**

because we have to consider every value