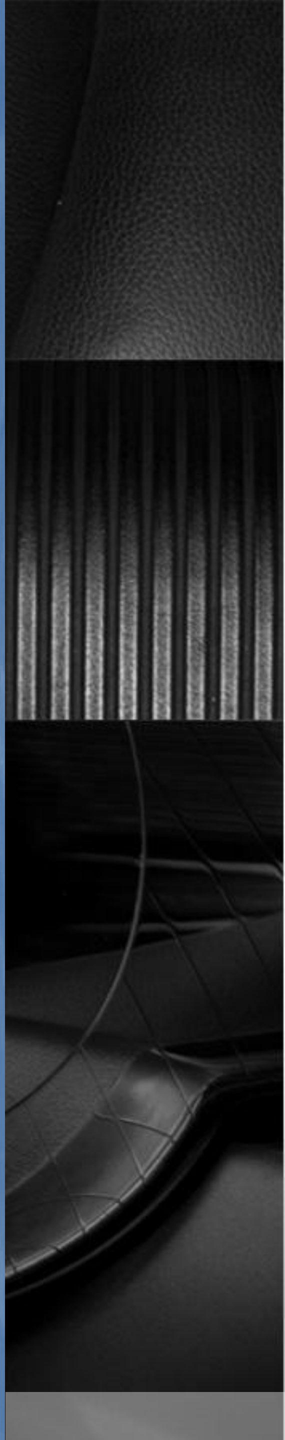
COMP4431 Artificial Intelligence

Constrains

Raymond Pang
Department of Computing
The Hong Kong Polytechnic University

Partly based on Prof. Liu Slides



Constrains

- Lots of constrains in daily life, due to:
  - Limited time
  - Limited resources
  - Certain policy
- Example
  - Resources allocation

    We have finite number of resources, e.g. food, toys, place etc.
  - Timetabling

    Time crashing must be avoided
  - Puzzles

Constraint Satisfaction Problem (CSP)

- The problem is defined through
  - a set of variables
  - a set of domains (possible values of variables specified by the problem)
  - a set of constraints

    describe allowable combinations of values for a subset of the variables

- *State* in a CSP
  - defined by an assignment of values to some or all variables

- *Solution* to a CSP
  - must assign values to all variables
  - must satisfy all constraints

Constraint Satisfaction Problems

- The formulation
  - Finite set of variables $X_1, X_2, \ldots, X_n$
  - Nonempty domain of possible values for each variable $D_1, D_2, \ldots, D_n$
  - Finite set of constraints $C_1, C_2, \ldots, C_m$
    - Each constraint C_i limits the values that variables can take,
    - e.g., $X_1 \neq X_2$
  - Each constraint C_i is a pair <scope, relation>
    - Scope = Tuple of variables that participate in the constraint.
    - Relation = List of allowed combinations of variable values.
      - May be an explicit list of allowed combinations.
      - May be an abstract relation allowing membership testing and listing.

CSPs --- what is a solution?

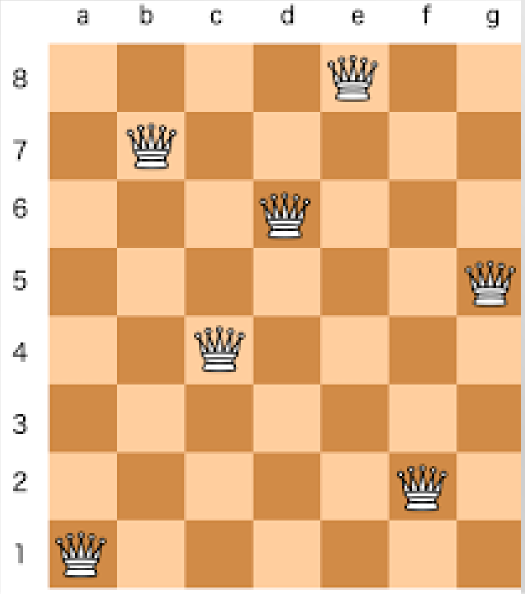
- A *state* is an *assignment* of values to some or all variables.
  - ❑ An assignment is *complete* when every variable has a value.
  - ❑ An assignment is *partial* when some variables have no values.
- Consistent assignment
  - ❑ assignment does not violate the constraints
- A solution to a CSP is a complete and consistent assignment.
- Some CSPs require a solution that maximizes an *objective function*.

8-Queens as a Constraint Satisfaction Problem (CSP)

- Variables: Queens, one per column
  - ❑Q1, Q2, …, Q8
- Domains: row placement, {1,2,…,8}
- Constraints:

Qi != Qj (j != i)

|Qi – Qj| != |i – j|

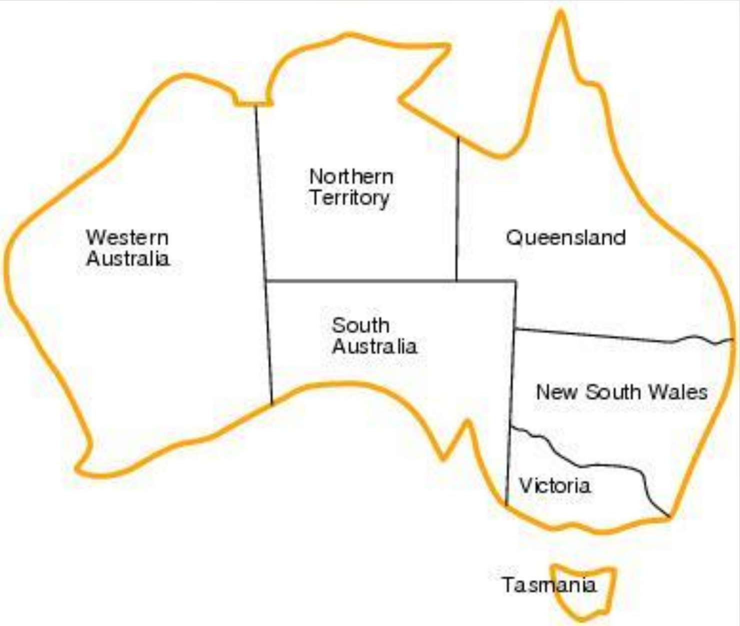


Sudoku as a Constraint Satisfaction Problem (CSP)

- Variables: 81 variables
  - ❑ A1, A2, A3, …, I7, I8, I9
  - ❑ Letters index rows, top to bottom
  - ❑ Digits index columns, left to right
- Domains: The nine positive digits
  - ❑ A1 $\in$ {1, 2, 3, 4, 5, 6, 7, 8, 9}
  - ❑ Etc.
- Constraints: 27 *Alldiff* constraints
  - ❑ *Alldiff*(A1, A2, A3, A4, A5, A6, A7, A8, A9)
  - ❑ Etc.

	1	2	3	4	5	6	7	8	9
A		6		1		4		5	
B			8	3		5	6		
C	2								1
D	8			4		7			6
E			6				3		
F	7			9		1			4
G	5								2
H			7	2		6	9		
I		4		5		8		7	

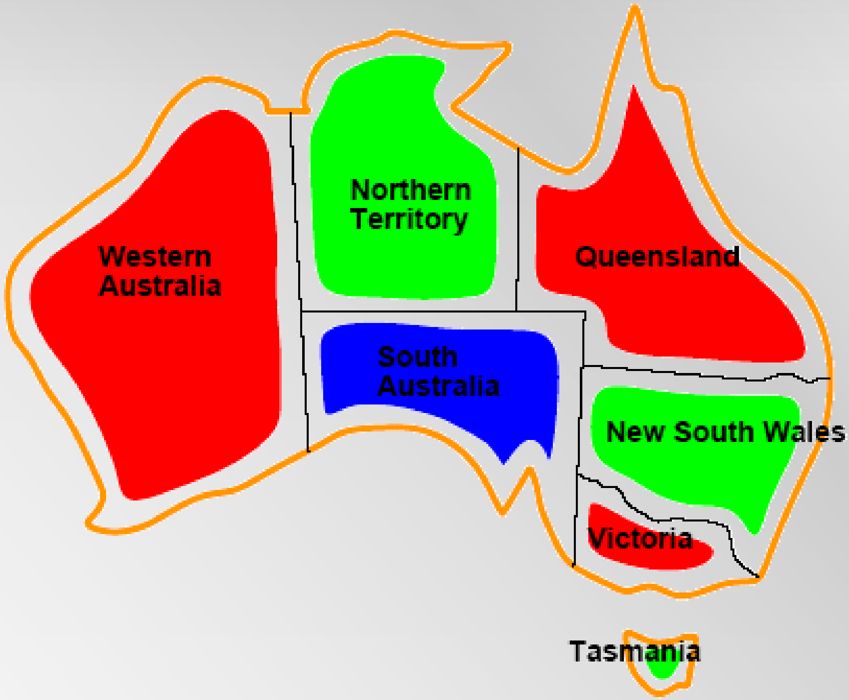
Classical CSP Example: Map Coloring



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

E.g. $WA \neq NT$

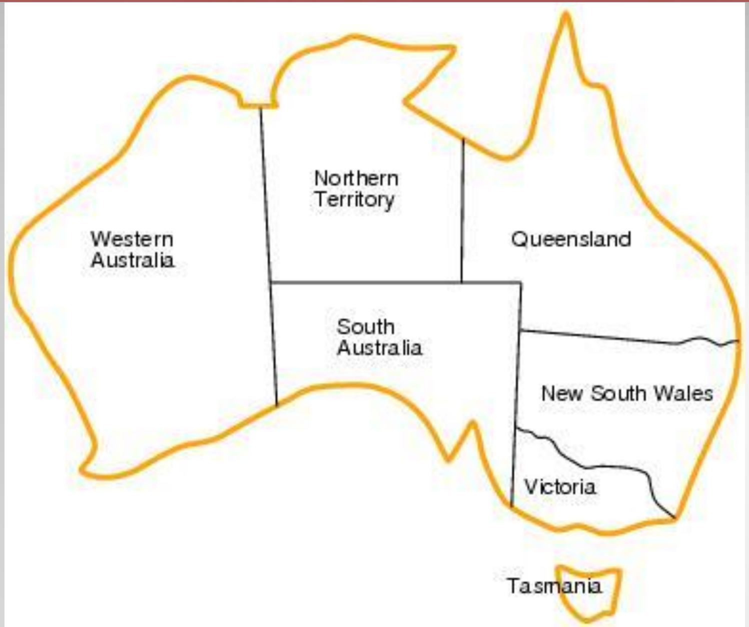
Classical CSP Example: Map Coloring

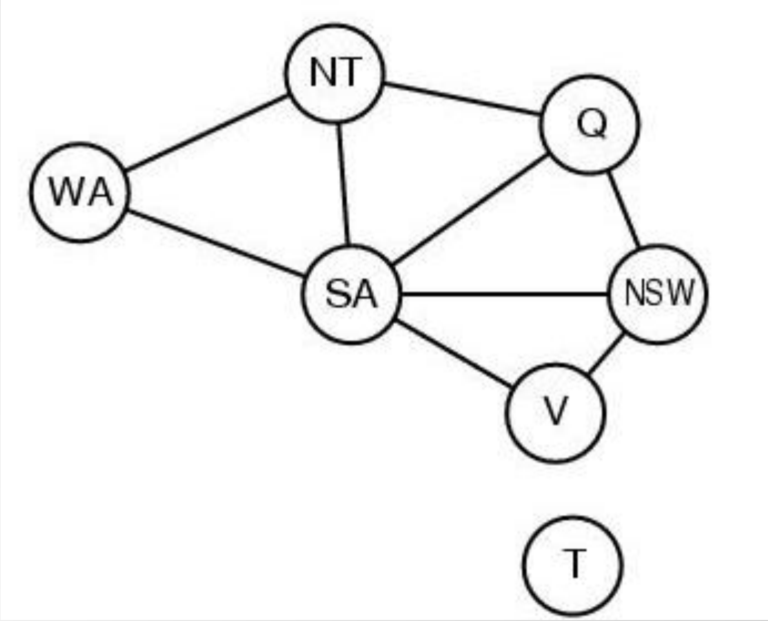


- Solutions are assignments satisfying all constraints, e.g. *{WA=red,NT=green,Q=red,NSW=green,V=red,SA=blue,T=green}*

Constraint Graph

- Constraint Graph:
  - nodes are variables
  - arcs are binary constraints
- Graph can be used to simplify search





CSP as a Standard Search Problem

- A CSP can easily be expressed as a standard search problem.
- Incremental formulation
  - ❑ *Initial State*: the empty assignment {}
  - ❑ *Successor function*: Assign a value to an unassigned variable provided that it does not violate a constraint
  - ❑ *Goal test*: the current assignment is complete
  - ❑ *Path cost*: constant cost for every step (so not really relevant)
- Can also use complete-state formulation
  - ❑ Local search techniques

Backtracking Search for CSPs

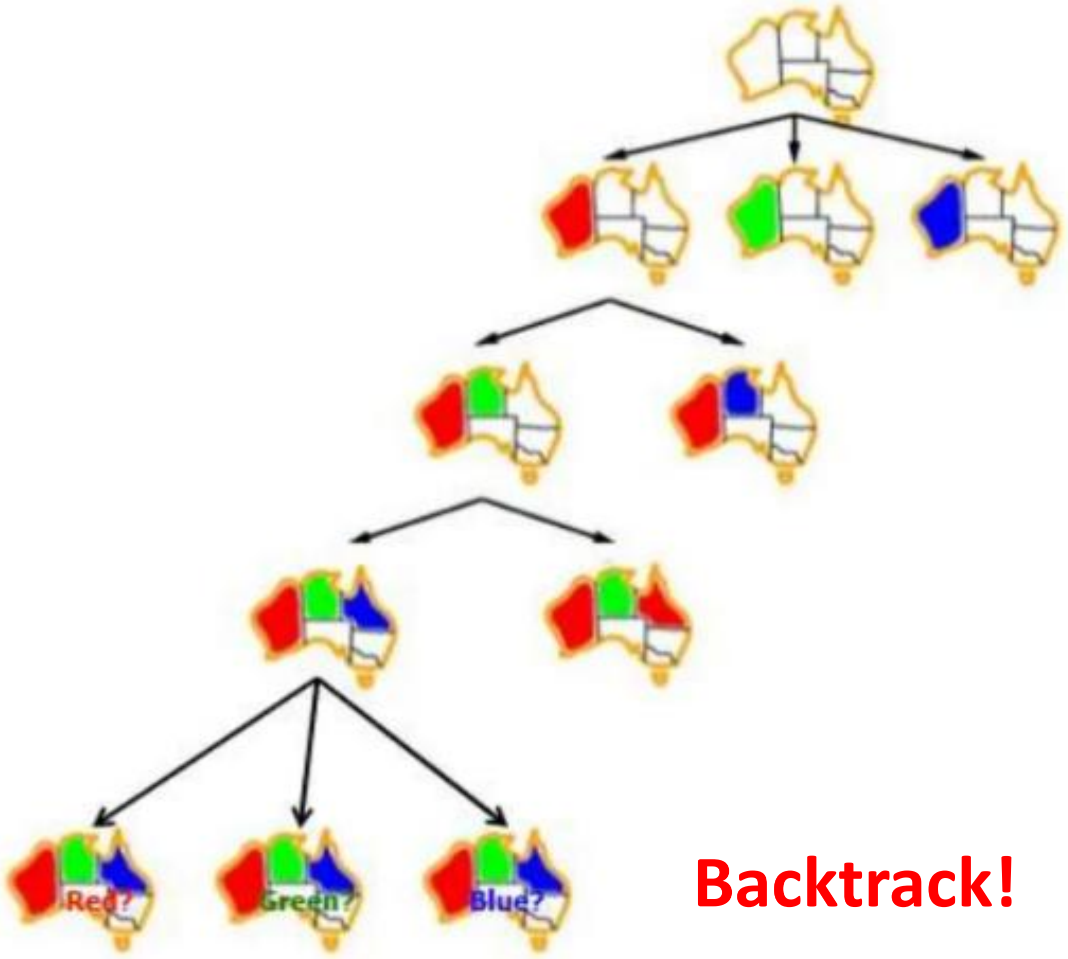
- A variation of depth-first search that is often used for CSPs
  - □ values are chosen for one variable at a time
  - □ if no legal values are left, the algorithm backs up and changes a previous assignment
  - □ very easy to implement

    initial state, successor function, goal test are standardized
  - □ not very efficient
  - □ can be improved by trying to select more suitable unassigned variables first

Steps in Backtracking

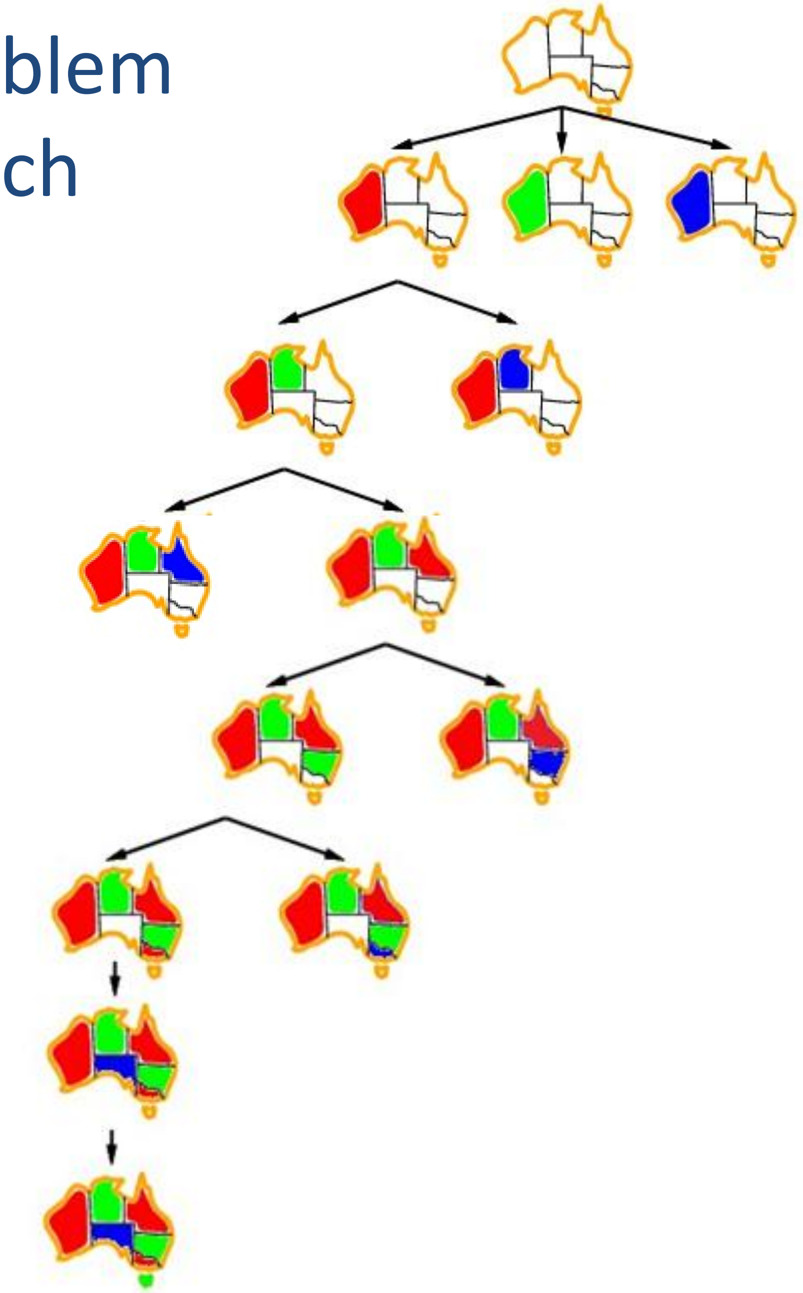
1. **Initialization:** Start with an empty assignment.
2. **Selection:** Choose an unassigned variable.
3. **Assignment:** Assign a value to the chosen variable.
4. **Consistency Check:** Check if the current assignment is consistent with the constraints.
5. **Recursion:** If the assignment is consistent, recursively try to assign values to the remaining variables.
6. **Backtrack:** If the assignment is not consistent, or if further assignments do not lead to a solution, undo the last assignment (backtrack) and try the next possible value.

Solve Map-Coloring Problem using Backtracking Search



Backtrack!

Solve Map-Coloring Problem using Backtracking Search

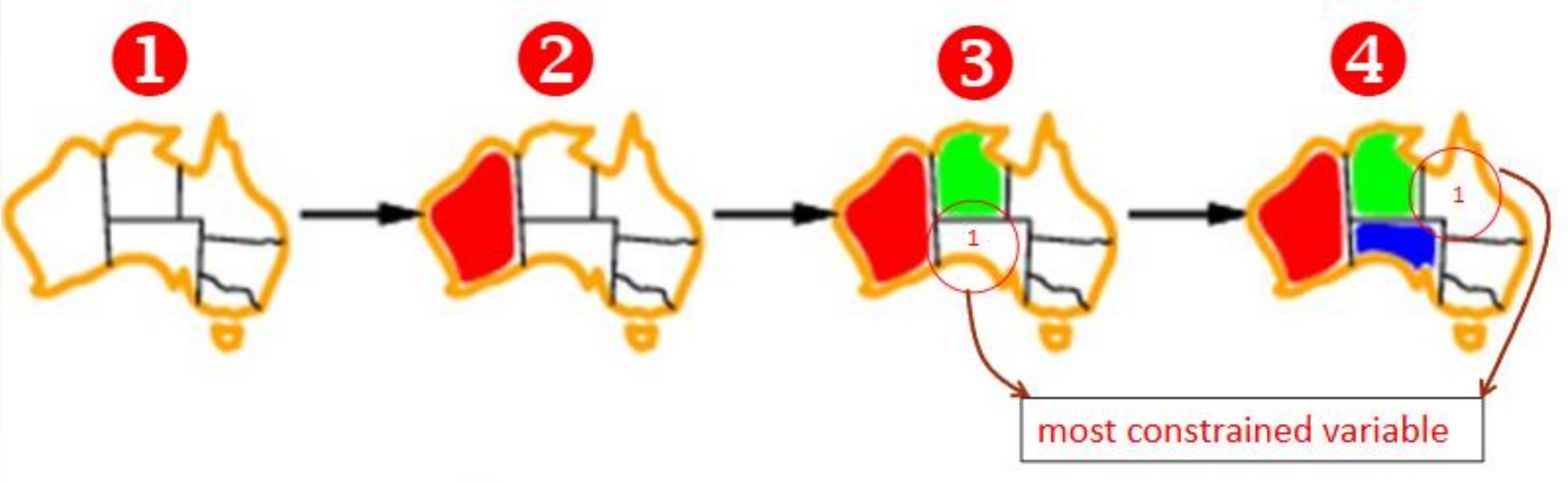


Improving CSP efficiency

- Previous improvements on uninformed search
  - introduce heuristics
- For CSPS, general-purpose methods can give large gains in speed, e.g.,
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?

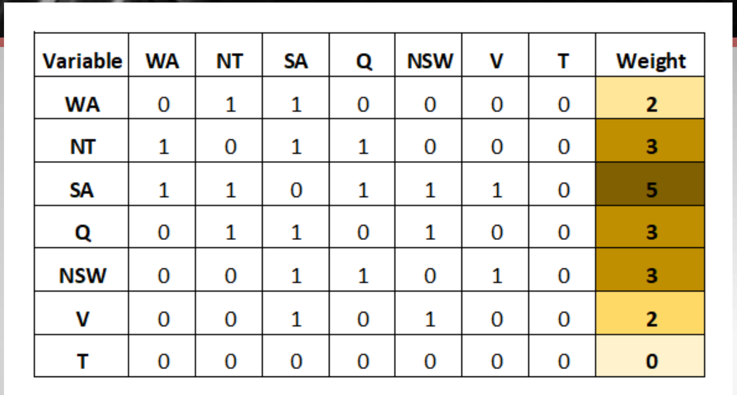
Variable Choice Heuristics

- Most-constrained variable :
  - Also called Minimum remaining values (MRV)
  - choose the variable with the fewest legal values

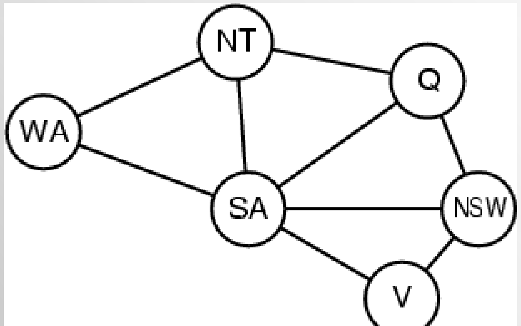


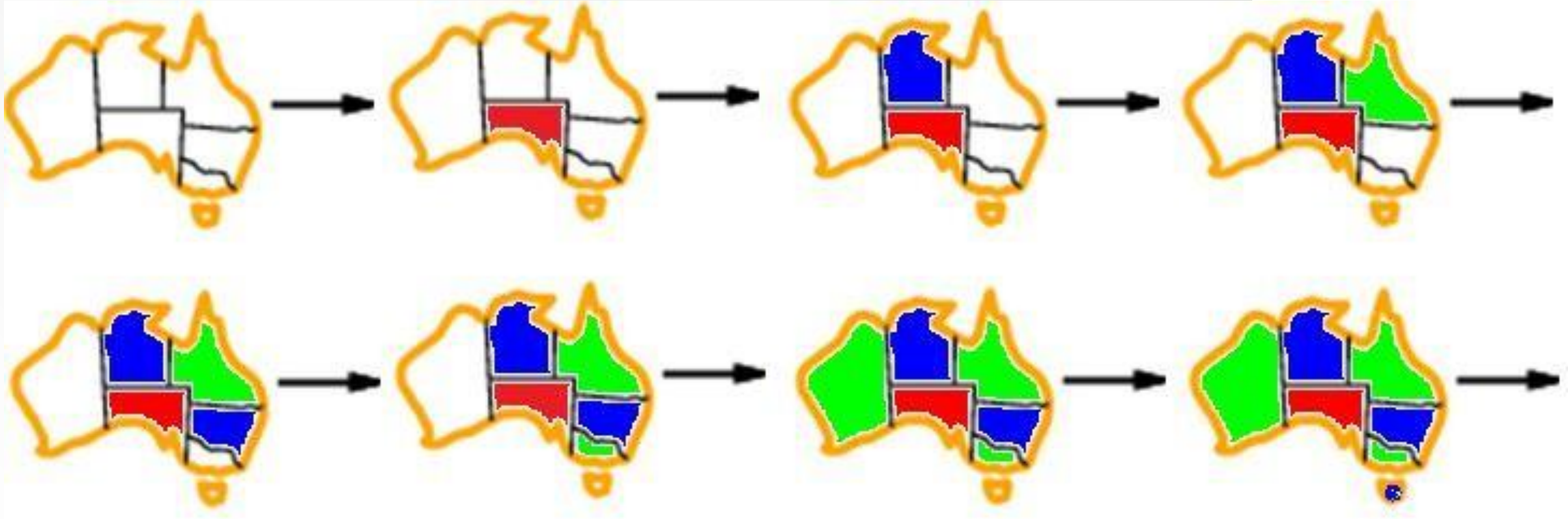
Variable Choice Heuristics

- Degree heuristic:
  - ❑ choose the variable with the most constraints on remaining variables
  - ❑ Or the most connected node in constraint graph (e.g. first is SA)



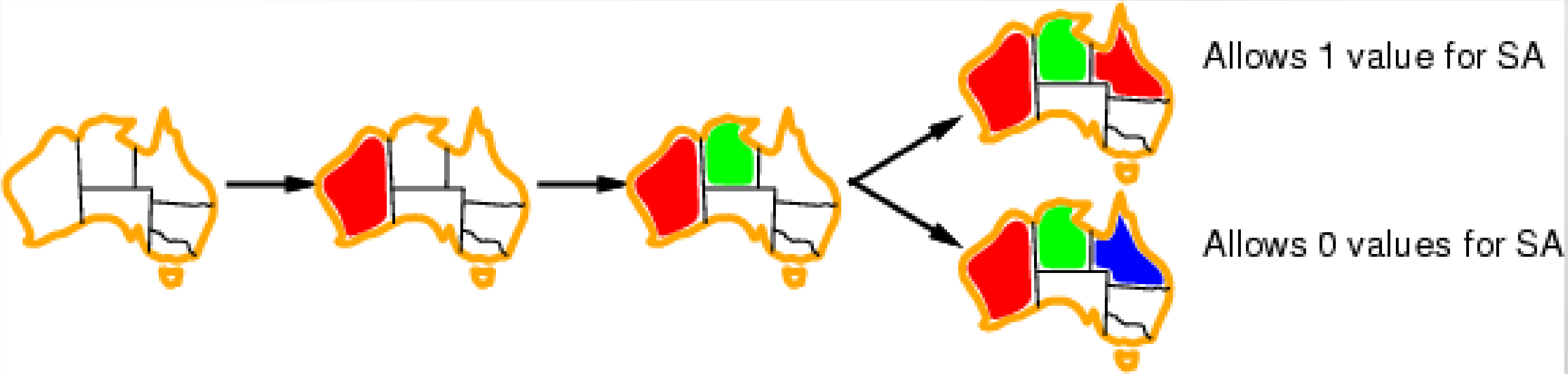
Variable	WA	NT	SA	Q	NSW	V	T	Weight
WA	0	1	1	0	0	0	0	2
NT	1	0	1	1	0	0	0	3
SA	1	1	0	1	1	1	0	5
Q	0	1	1	0	1	0	0	3
NSW	0	0	1	1	0	1	0	3
V	0	0	1	0	1	0	0	2
T	0	0	0	0	0	0	0	0





Value Choice Heuristics

- Least constraining value:
  - ❑ for a given a variable, choose the least constraining value: the one that rules out the fewest values in the remaining variables



Backtracking Search

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

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

Backtracking search

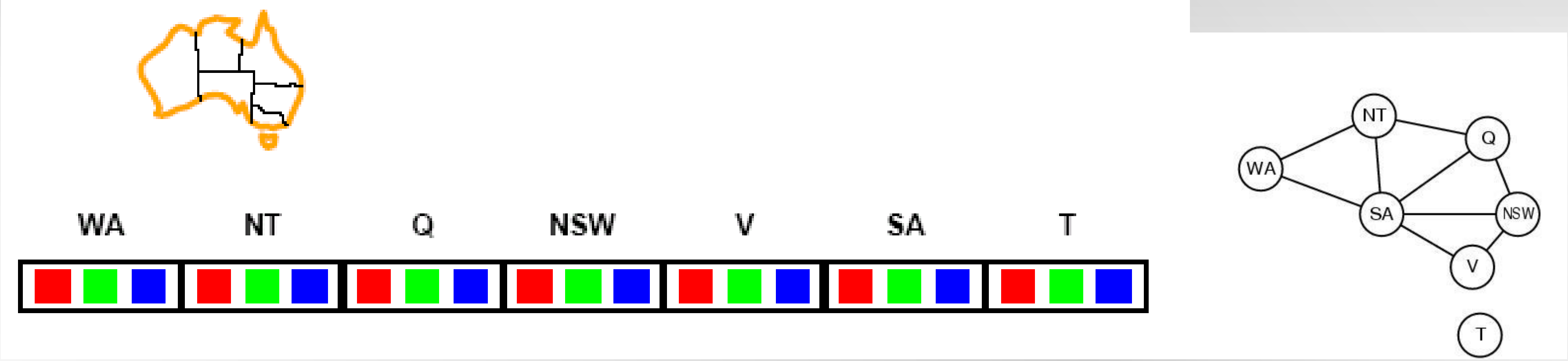
SELECT-UNASSIGNED-VARIABLE

- Most-constrained variable
- If a tie (such as choosing the start state), choose the variable involved in the most constraints (degree heuristic) E.g., in the map example, SA adjacent to the most states.

ORDER-DOMAIN-VALUES

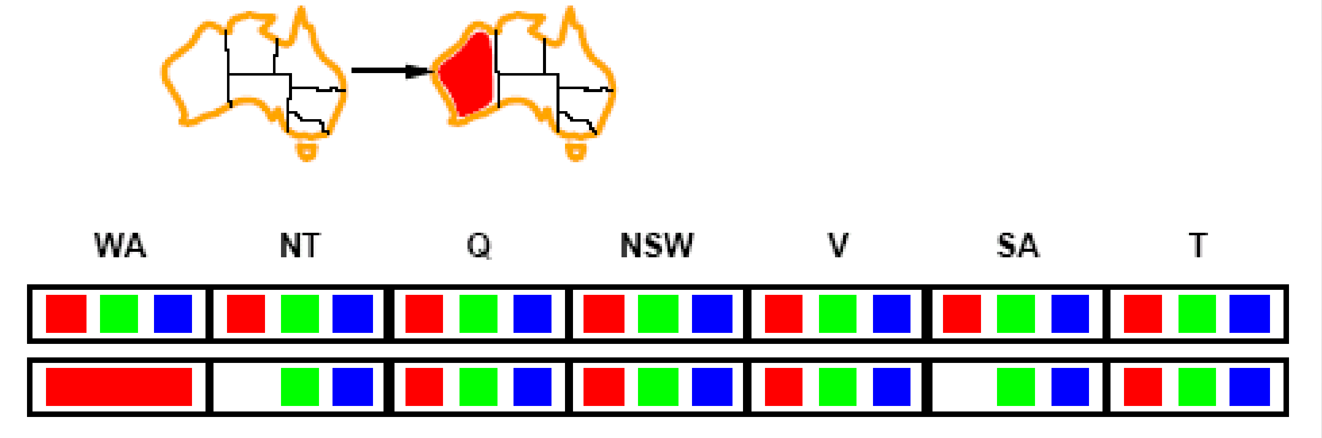
- Least Constraining Value
- Rules out the fewest choices for the variables it is in constraints with
- Leave the maximum flexibility

Forward checking (INFERENCE)

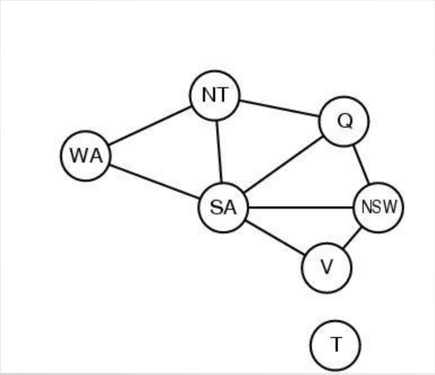


- Can we detect inevitable failure early?
  - *And avoid it later?*
- *Forward checking idea:* keep track of remaining legal values for unassigned variables.
- Terminate search when any variable has no legal values.

Forward checking

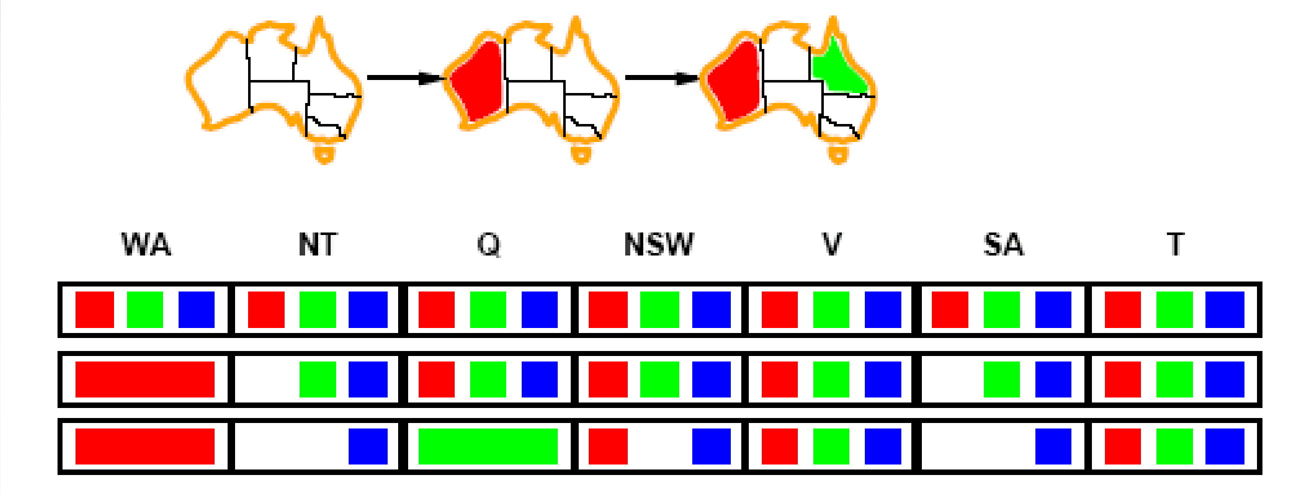


WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦

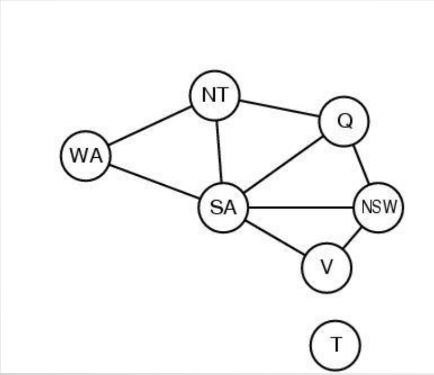


- Assign *{WA=red}*
- Effects on other variables connected by constraints to WA
  - *NT can no longer be red*
  - *SA can no longer be red*
- *Note: this example is not using MRV; if it were, we would choose NT or SA next. But, we will choose Q next. This example is from the text. It shows the example here, then talks through what would happen if we had used MRV.*

Forward checking

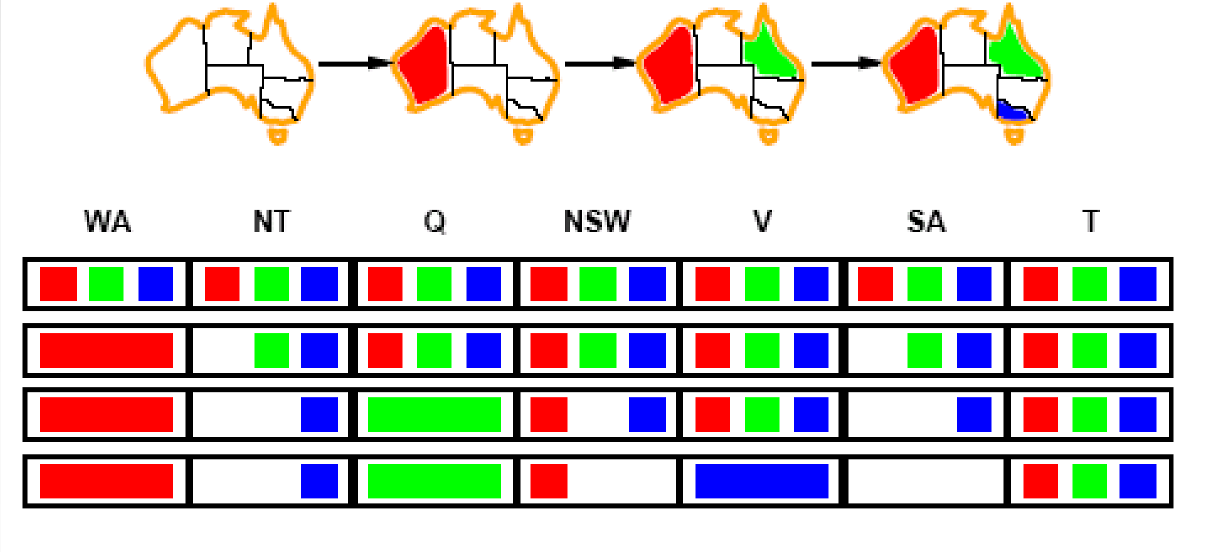


WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦

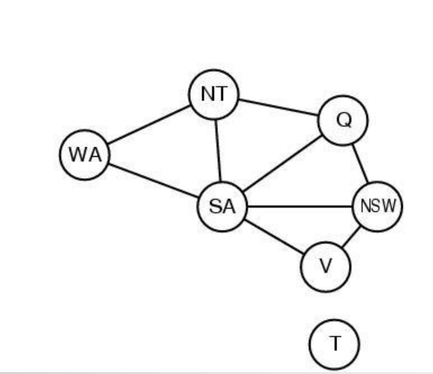


- Assign *{Q=green}*
- Effects on other variables connected by constraints with WA
  - *NT can no longer be green*
  - *NSW can no longer be green*
  - *SA can no longer be green*

Forward checking



WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦
🟥	🟦	🟩	🟥	🟦		🟥🟩🟦



- Assign *{V=blue}*
- Effects on other variables connected by constraints with WA
  - *NSW can no longer be blue*
  - *SA is empty*   *（Problem!!!)*
- Forward Checking has detected that partial assignment is *inconsistent* with the constraints and backtracking can occur.

Applications of CSP

- CSP enable solutions of complex problems
  - Exam and class scheduling
  - Job scheduling
  - Air traffic flow management
  - Bandwidth allocation in wireless communication
  - Register allocation in computer compiler
- Complex problems may involve more complex constraints

Varieties of constraints

- Unary constraints involve a single variable.
    - ❑ e.g. $SA \neq green$
- Binary constraints involve pairs of variables.
    - ❑ e.g. $SA \neq WA$
- Higher-order constraints involve 3 or more variables.
    - ❑ Professors A, B,and C cannot be on a committee together
    - ❑ Can always be represented by multiple binary constraints
- Preference (soft constraints)
    - ❑ e.g. *red* is better than *green* often can be represented by a cost for each variable assignment
    - ❑ combination of optimization with CSPs

Summary

- Constraint Satisfaction Problem
  - Problem definition
  - Backtracking search
  - Heuristics for CSP
    - Most-constrained variable
    - Degree heuristic
    - Least constraining value
  - Application of CSP