Artificial Intelligence
Constraint Satisfaction Problems

Michael McConnell

University of Vermont

March 15, 2022

Logistics

- Uninformed Search
- Informed Search
- Adversarial Search
- Today: **CSP in a search context**
- Friday: Active learning session (no exam)

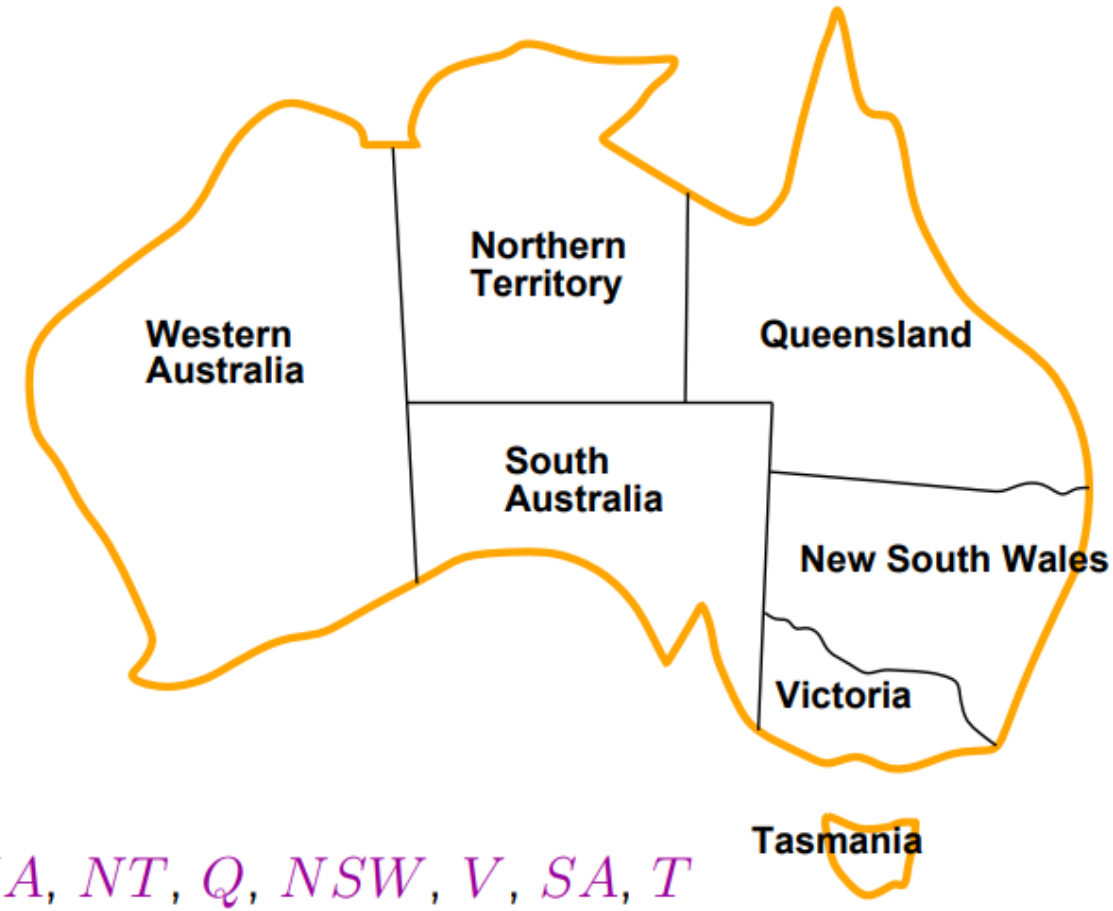
Constraint Satisfaction Problems

- **Search Problems:** We treat the state as a type of black box essentially.
- **Constraint Satisfaction Problems:** The state is defined by variables X with values from some domain D.
- The goal test is a set of constraints specifying allowable combinations of values for subsets of variables.
- We can use this to create general-purpose algorithms with more power than standard search algorithms

Map Coloring

- **Variables:** In our map coloring example, the variables are each region.
- **Domain**: The colors we are setting the map to.
- **Constraints:** All adjacent regions must be different colors. No two regions which share a border can be of the same color.

Map Coloring

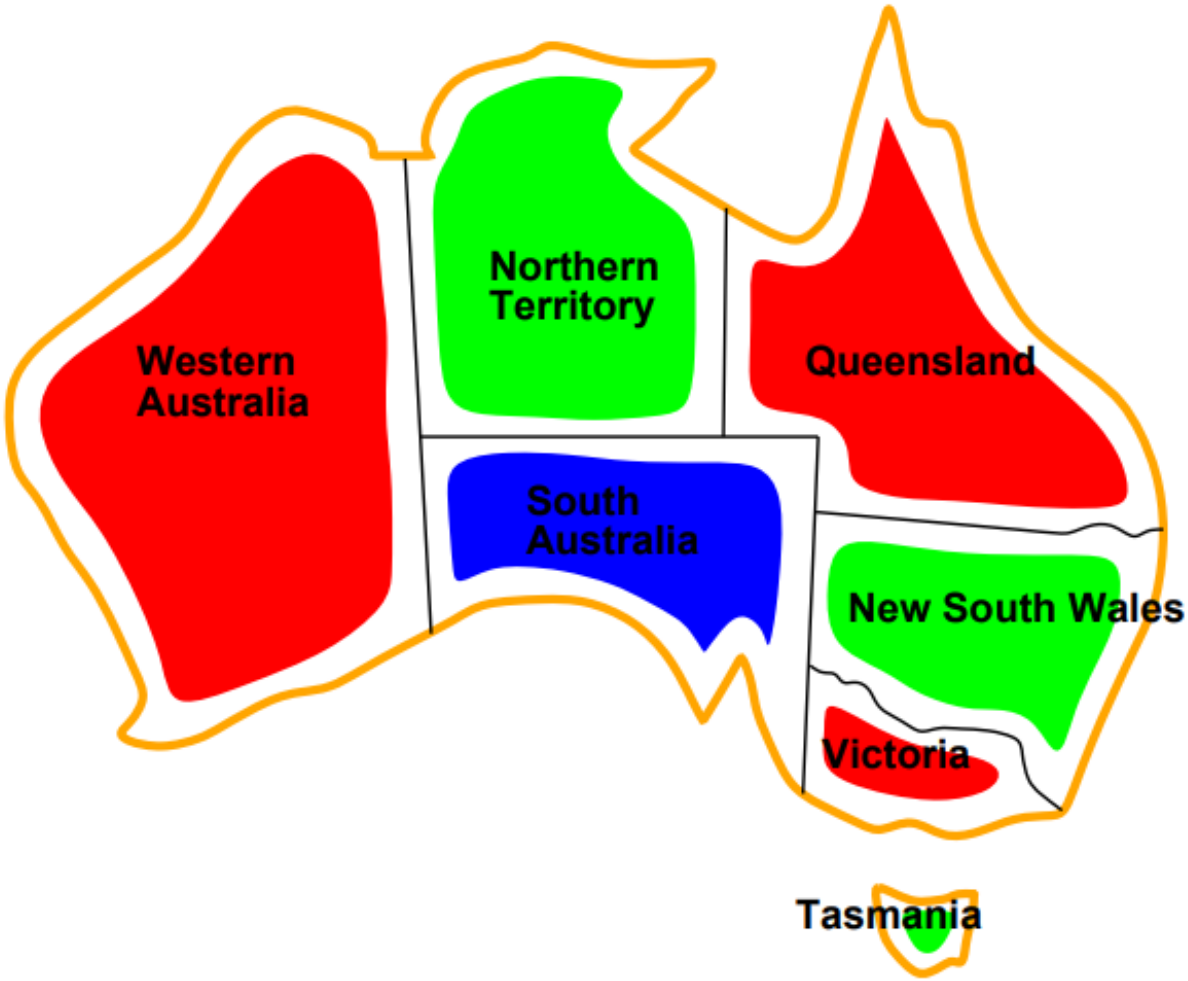


Variables WA, NT, Q, NSW, V, SA, T

1

[1] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Map Coloring



2

[2] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Example: 4-Queens

- State Variables/Domain: 4 queens in 4 separate columns (domain is where each queen can be located.)
- Operations: Move the queen within the column.
- Constraints: Current number of attacks.
- Goal Test: No attacks

Example: Sudoku

	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

1	2	6	4	3	7	9	5	8
8	9	5	6	2	1	4	7	3
3	7	4	9	8	5	1	2	6
4	5	7	1	9	3	8	6	2
9	8	3	2	4	6	5	1	7
6	1	2	5	7	8	3	9	4
2	6	9	3	1	4	7	8	5
5	4	8	7	6	9	2	3	1
7	3	1	8	5	2	6	4	9

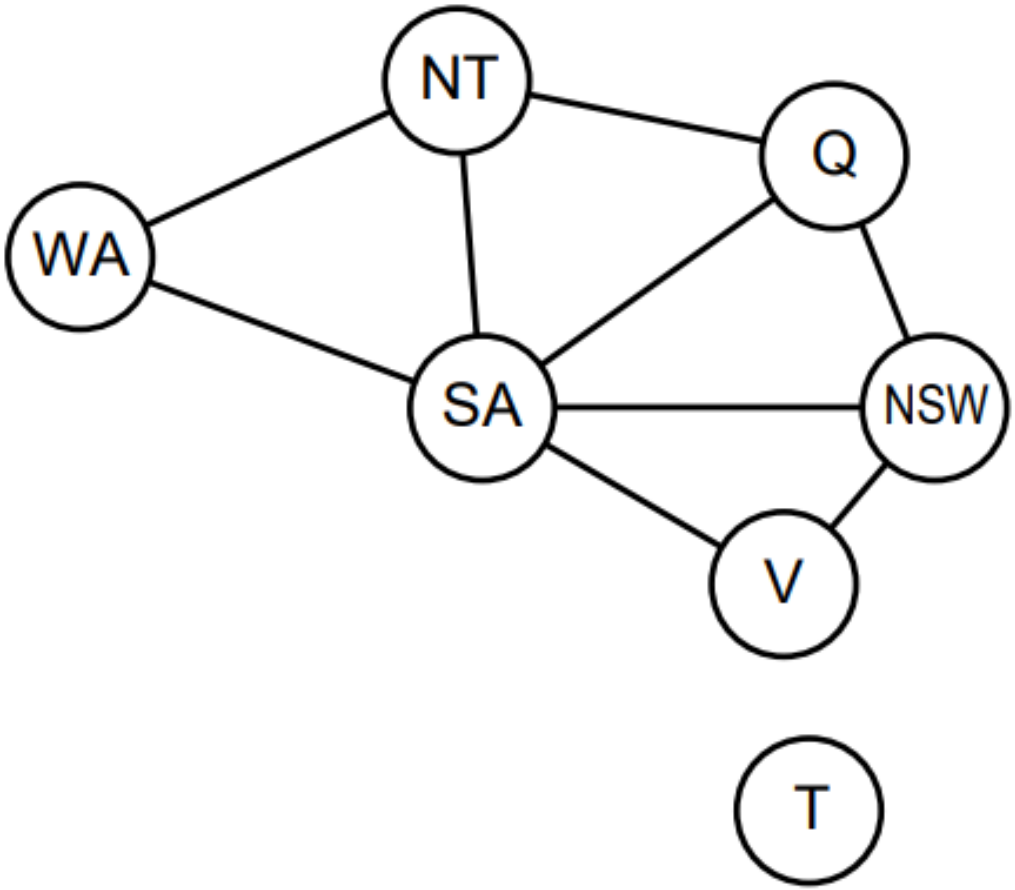
Example: Sudoku

- Variables/ Domain: We have each cell representing a variable, and the domain of each cell is 1 through 9 for any cell that is empty.
- State: Any complete board where each value is filled.
- Goal State: Column, Block, and Row constraints on each of the values are met.

Example: 8-Puzzle

- Variables: The 9 cells of the puzzle, the domain is the value in each cell, Blank or 1 through 8.
- State: If we have a value for each cell we have a state.
- Constraints/Goal: To have each piece in the spot it is required to be in.

Constraint Graph



3

[3] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Constraint Graphs

- In a constraint graph nodes are variables and edges (we will refer to sometimes in this lecture as 'arc's) show the various constraints.
- Most of the time with CSP problems we will convert into this kind of a graph structure in order to perform our searches.
- Note: Tasmania is an independent subproblem, it does not matter what color we choose it to be.

Constraint Variables

- **Discrete variables**: With these we can consider two types of contexts finite domains and infinite domains. An example of a finite domain would be Boolean Satisfiability problems (probably seen in CS224 or CS125). An infinite discrete domain might consist of integers or strings, etc. This requires some formalization for talking about these constraints. Linear constraints placed on these are solvable.
- **Continuous Variables**: Linear constraints are solvable in polynomial time using linear optimization methods (linear programming).

Constraint Variables

- **Unary Constraints:** Constraints that involve a single variable. South Australian cannot be green.
- **Binary Constraints:** Constraints involving a variable pairs, SA!= WA.
- **Higher Order Constraints:** Constraints that involve 3 or more variables at the same time. (think of this as solving some linear algebra problem with column constraints)
- **Soft Constraints:** Also called 'preferences' by some texts. In our coloring example we might say that red is better than green. We might represent this by a separate cost for each variable assignment which adds another dimension for solving. We call this Constrained Optimization.

Real World CSP Examples

- Timetable Problems (what class is offered in what classroom)
- Transportation Scheduling
- Spreadsheets
- Factory Scheduling
- Floorplans

Standard Search Formulation

- **Initial State:** The empty assignment $\{\}$
- **Successor Function:** We want to assign a value to an unassigned variable that does not conflict with the current assignment. This process needs to fail if there is no possible assignment that matches our constraints.
- **Goal Test:** Test that the current assignment state is complete and meets all the constraints.
- This setup is going to be the same for all CSPs.

Backtracking Search

- Backtracking is the basic 'uninformed' search model for CSPs. It is just doing search with a fixed method.
- If we have a CSP with single variable assignment, backtracking is equivalent to Depth-First Search.
- We only need to consider assignments of a single variable at each node so if we are at a depth d, then there are d^n leaves. (where n is the number of variables we are considering).
- Variable assignment in this setting is commutative, so in our map coloring example an assignment set like [WA = red and Tasmania = Blue] is equivalent to [Tasmania = Blue and WA = red]
- Backtracking can solve n-queens up to about n=25

Backtracking Algorithm Pseudocode

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

4

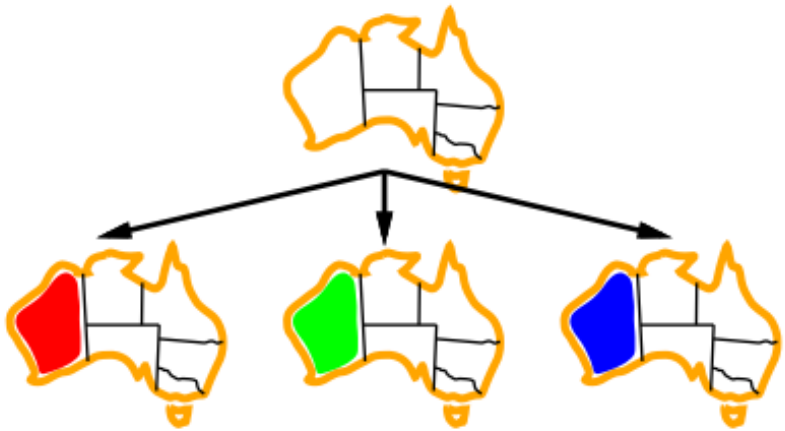
[4] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Map Coloring with Backtracking

5

[5] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

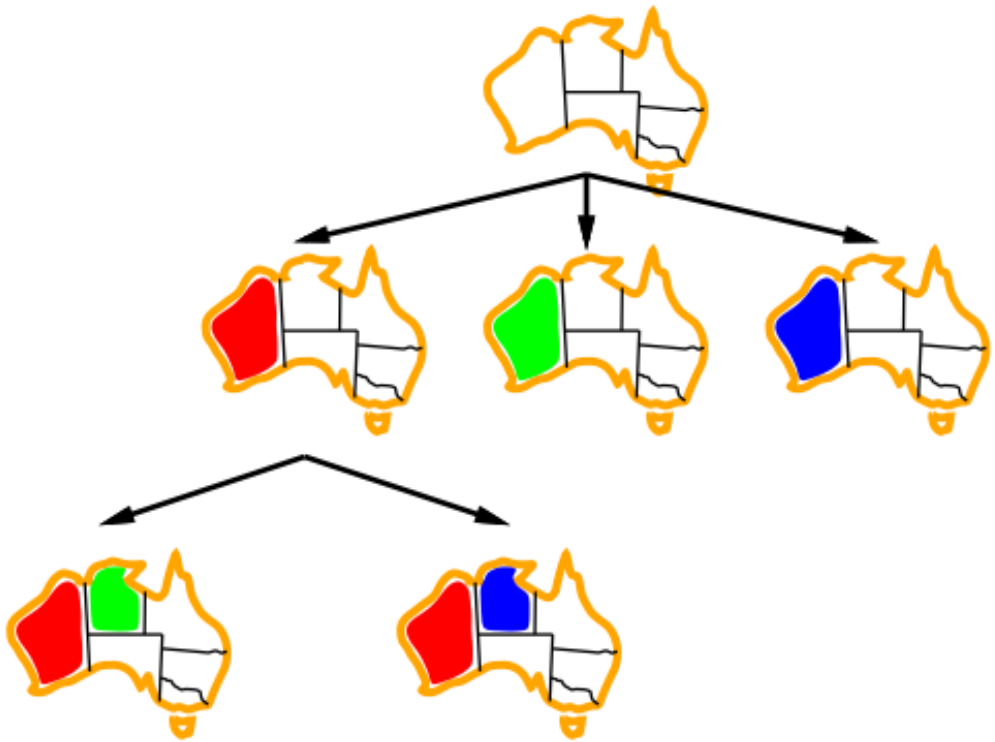
Map Coloring with Backtracking



6

[6] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

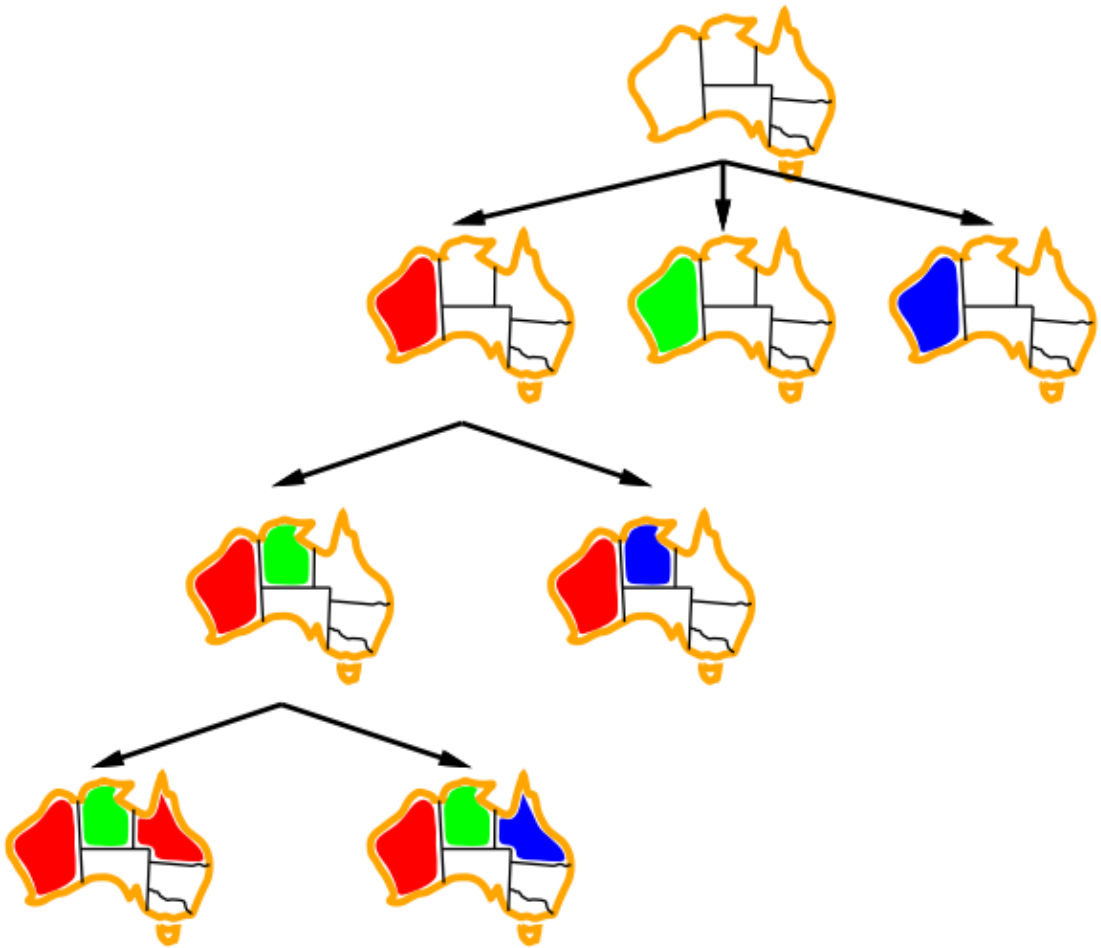
Map Coloring with Backtracking



7

[7] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Map Coloring with Backtracking



8

[8] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

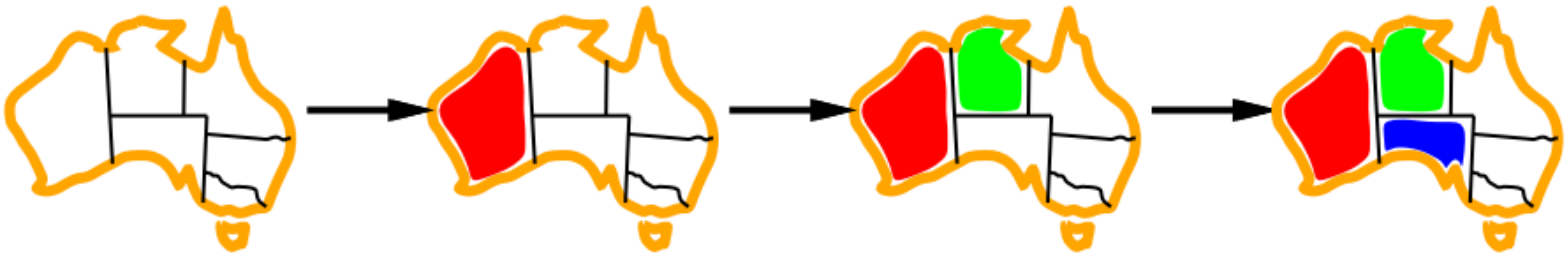
Improving Backtracking

- What variable should we assign next?
- In what order should values be tested?
- Can we have some earlier failure detection on a given branch?
- Can we use something in the problem structure to give us an advantages?

Minimum Remaining Values

- For this improvement we say that we want to if possible always choose the variable with the fewest remaining legal values.

Minimum Remaining Values



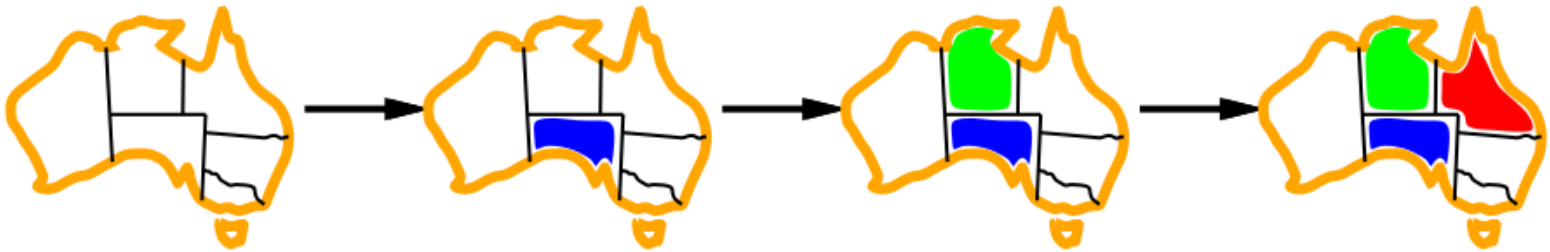
9

[9] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Degree Heuristic

- What happens if there is a tie between two variables after applying the Minimum Remaining Values technique?
- We can use a degree heuristic to choose the variable with the most constraints on the remaining variables.

Degree Heuristic



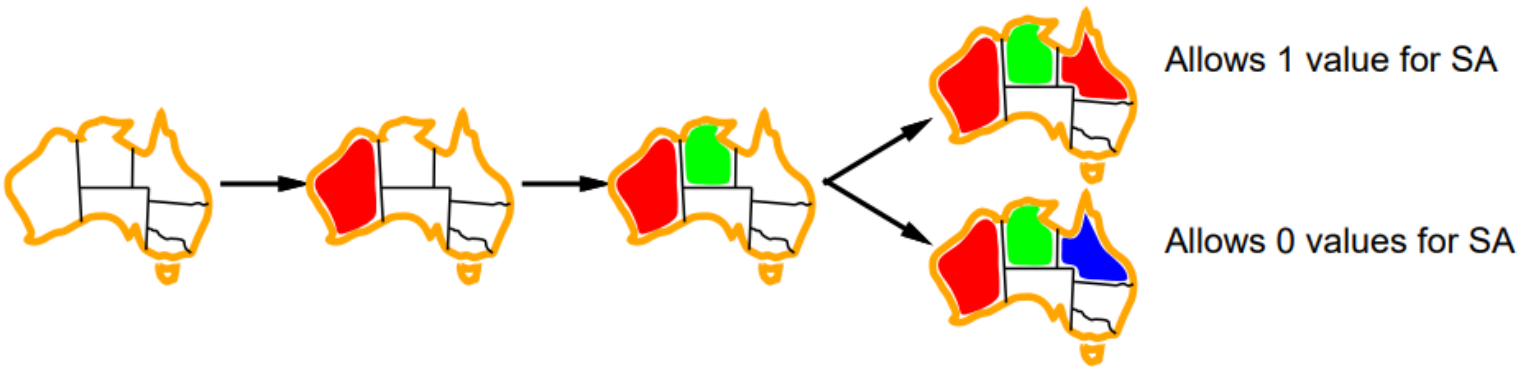
10

[10] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Least Constraining Value

- With a given variable we wnt to pick the least constraining value.
- By least constraining we mean that if there is some value that rules out the fewest options on the other remaining values going forward we want to choose that.
- With these improvements we can get up to 1000 queens.

Least Constraining Value



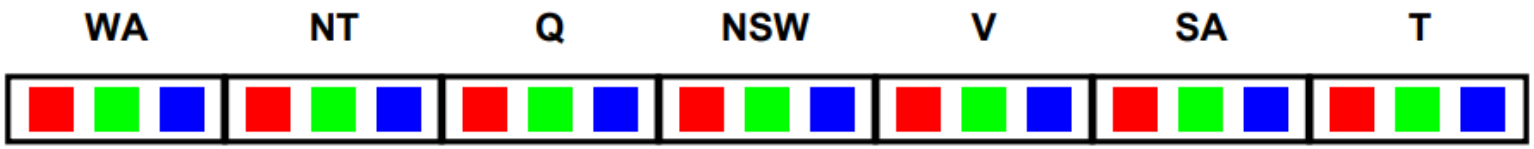
11

[11] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Forward Checking

- **Concept:** Keep track of all remaining legal values for each unassigned variable.
- Terminate the search loop when any of the variables no longer has a legal value.
-

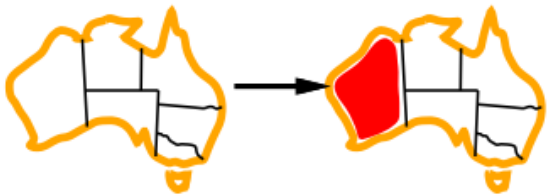
Forward Checking

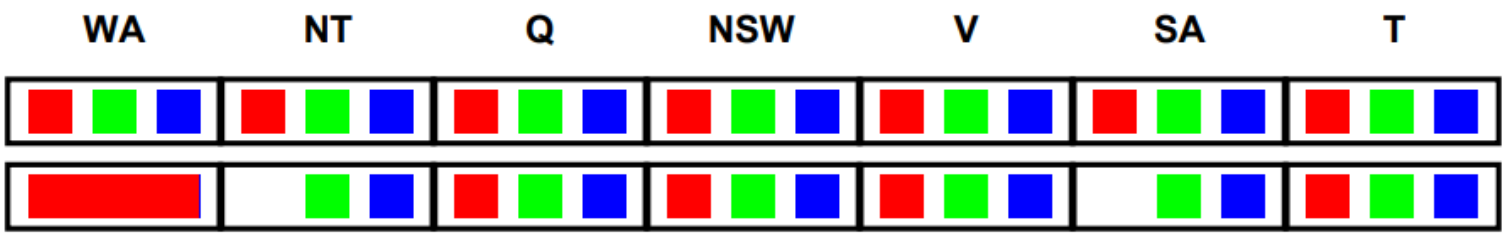


[12]

[12] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Forward Checking





13

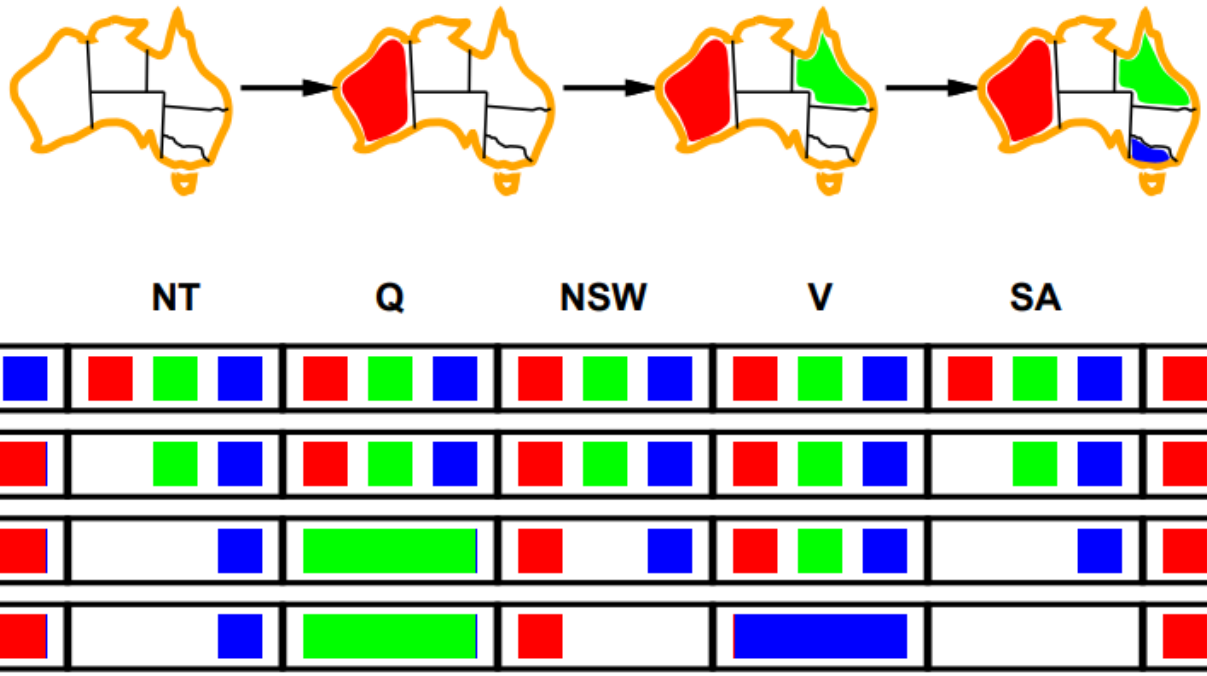
[13] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Forward Checking

14

[14] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Forward Checking



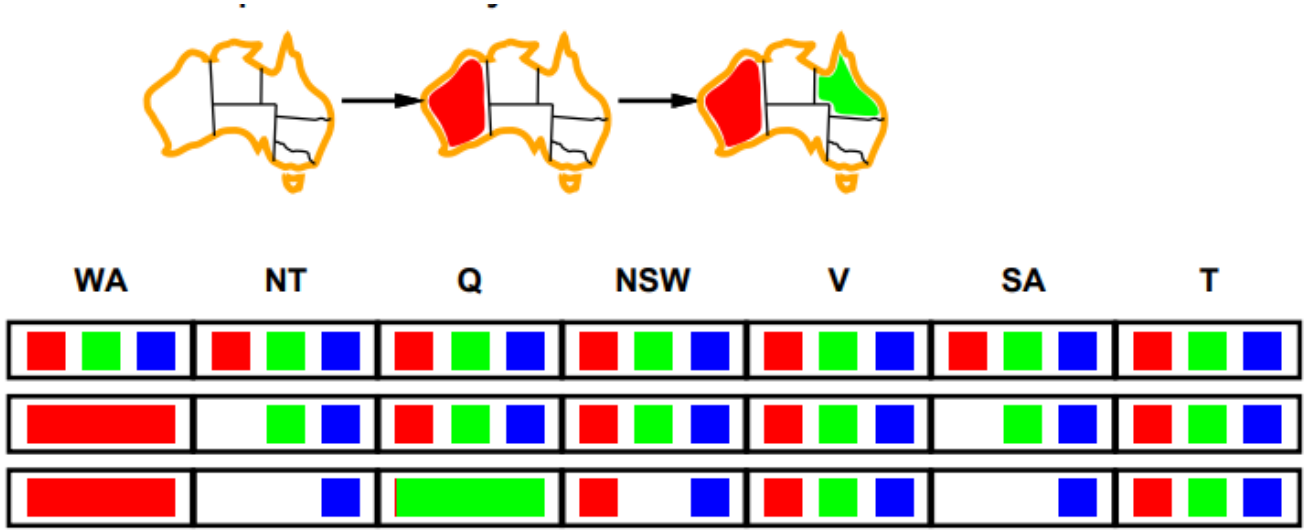
15

[15] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Constraint Propagation

- Forward checking is just propagating information from assigned to unassigned variables. It doesn't provide any early detection for failures.
- Constraint propagation methods on the other hand repeatedly enforce all these constraints locally at each step.

Constraint Propagation



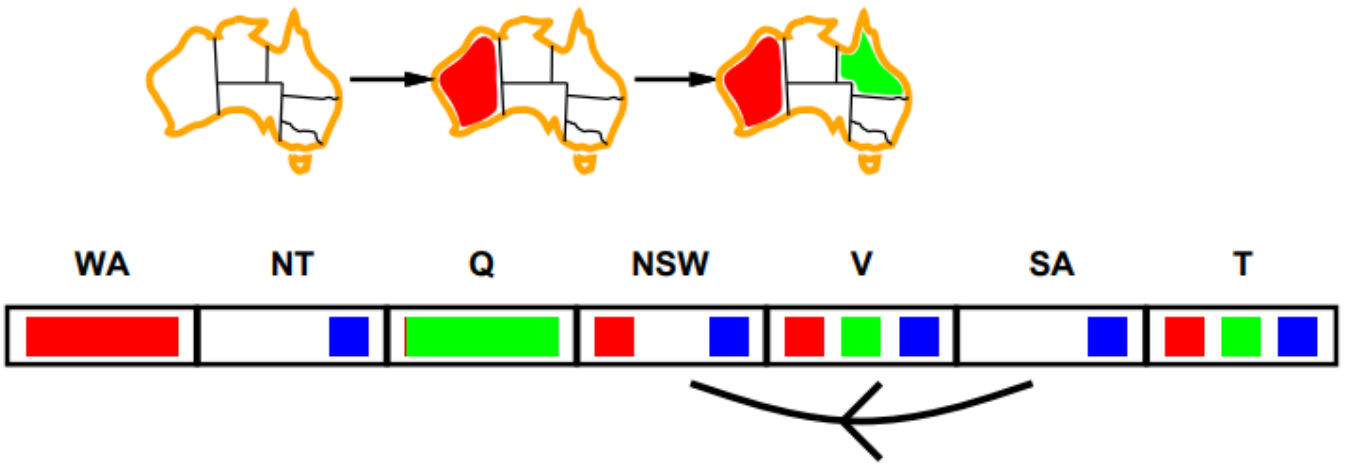
16

[16] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Arc Consistency

- The easiest form of constraint propagation is called arc consistency.
- We treat each edge of the graph as an arc and for every set of possibilities at each pair of nodes, we say is consistent if for every value (x) at one node there is some allowed (y) at the second.
- If The set of variables at the node with x in it loses a value based on some new assignment, we then need to recheck all the neighbors of that node in the graph to see if they remained consistent.
- It doesn't matter when this is ran, it can be either immediately run after each new assignment or at the start of each new round. Both are equivalent.

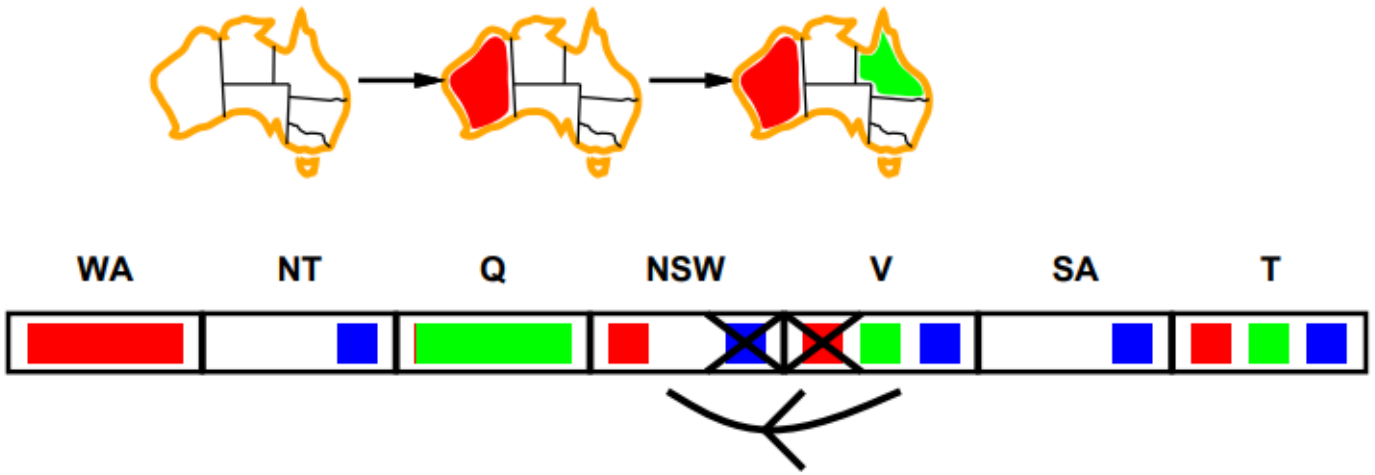
Graph Example



17

[17] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

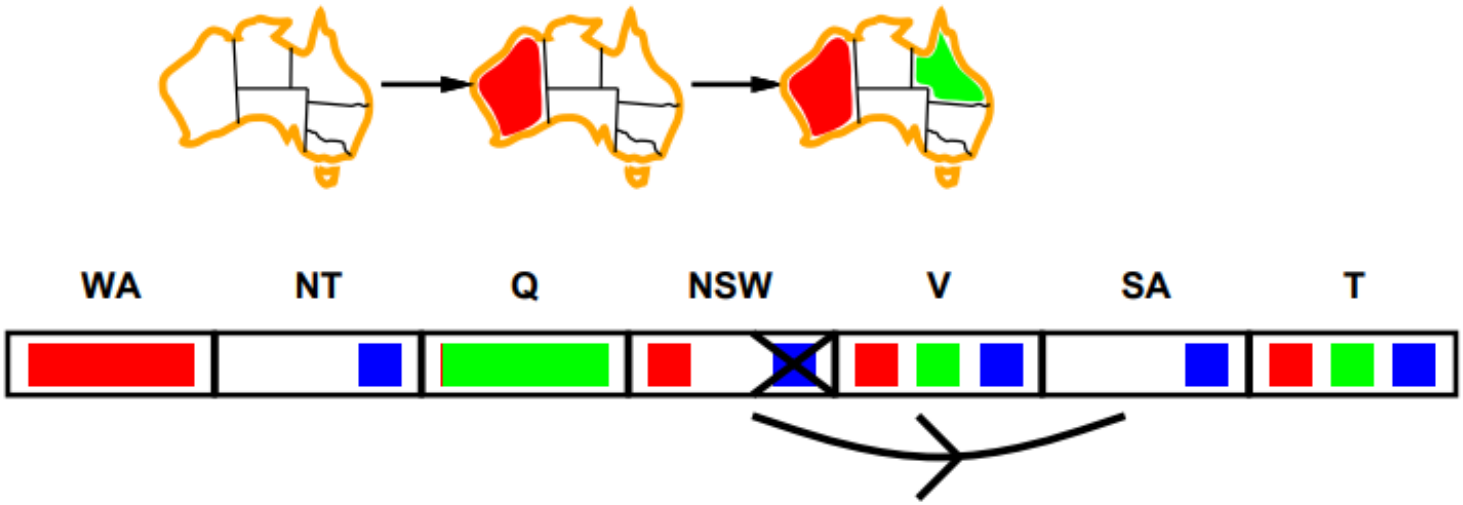
Arch Consistency Example



18

[18] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

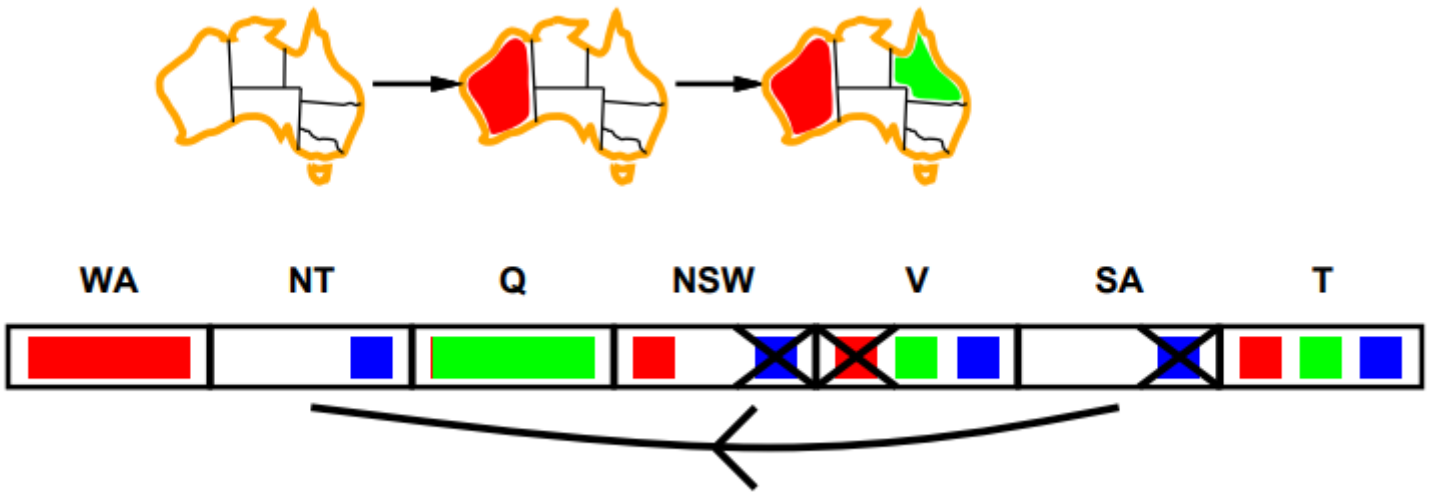
Arc Consistency Example



19

[19] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Arc Consistency Example



20

[20] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Arch Consistency Algorithm

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains
  inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \ldots, X_n\}$
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**
      for each X_k **in** NEIGHBORS[X_i] **do**
        add (X_k, X_i) to *queue*

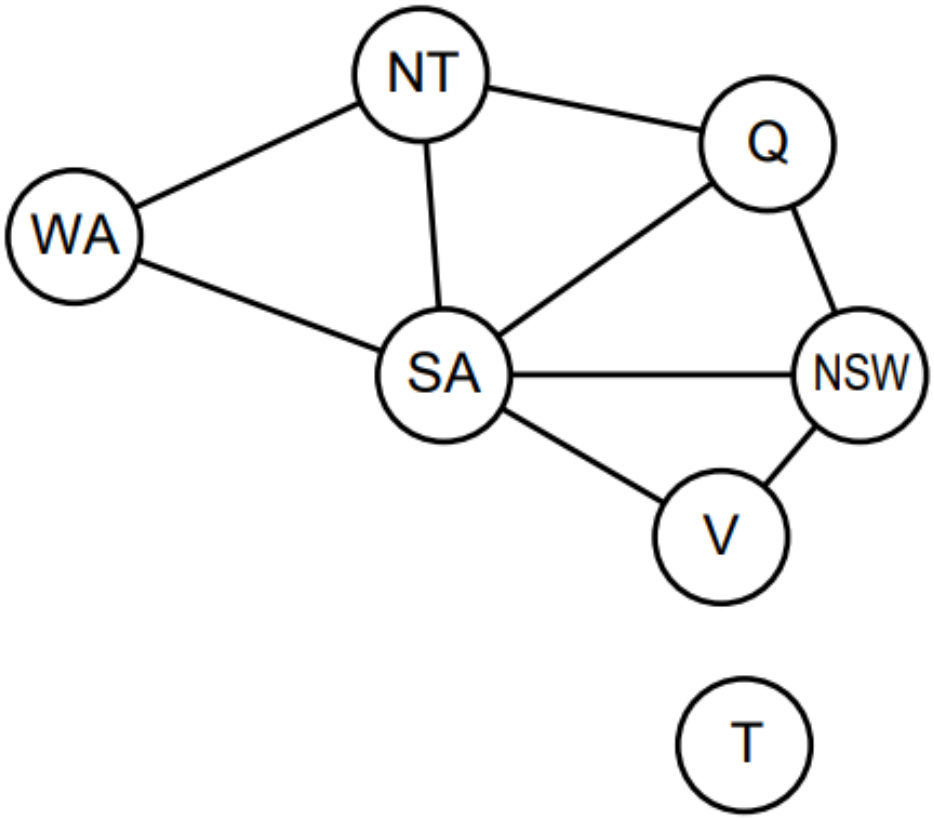
---

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff succeeds
  removed $\leftarrow$ *false*
  for each x **in** DOMAIN[X_i] **do**
    if no value y in DOMAIN[X_j] allows (x,y) to satisfy the constraint $X_i \leftrightarrow X_j$
      then delete x from DOMAIN[X_i]; *removed* $\leftarrow$ *true*
  return *removed*

21

[21] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

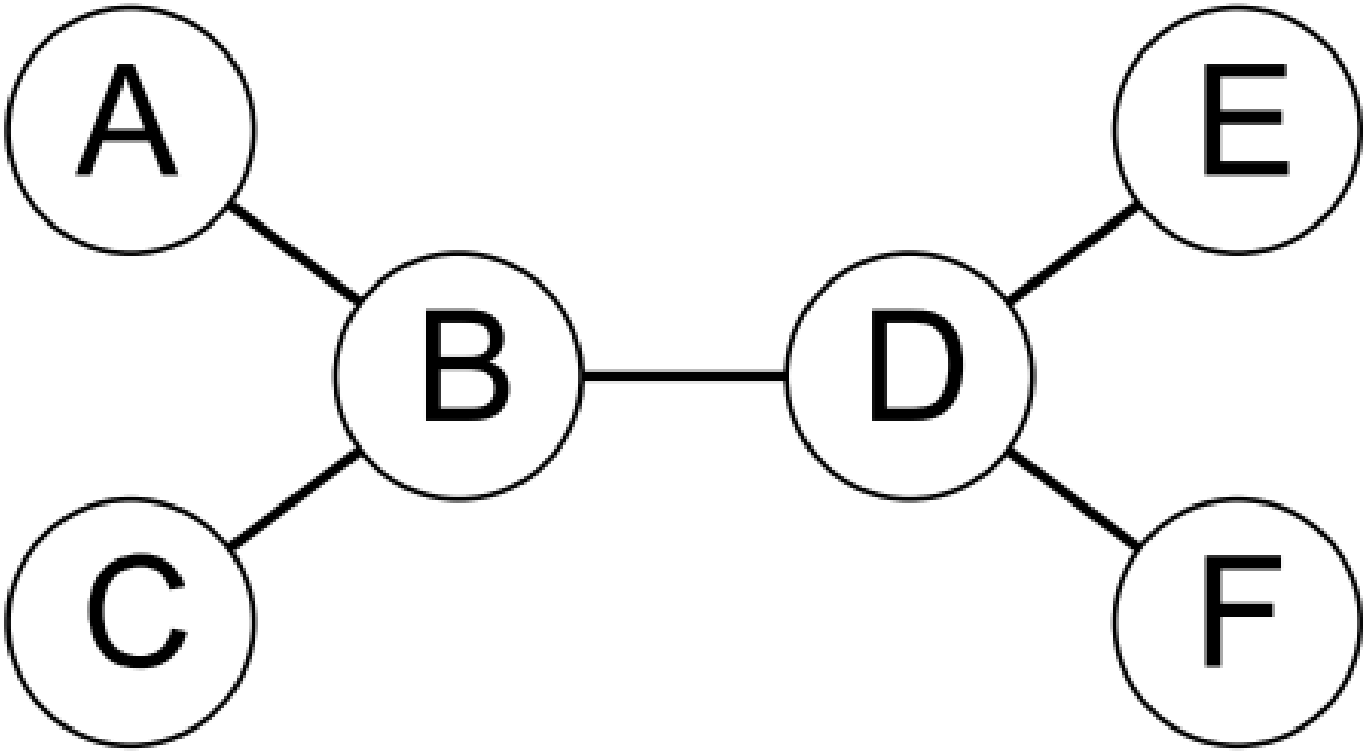
Map Graph



22

[22] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Loopless Graph Example



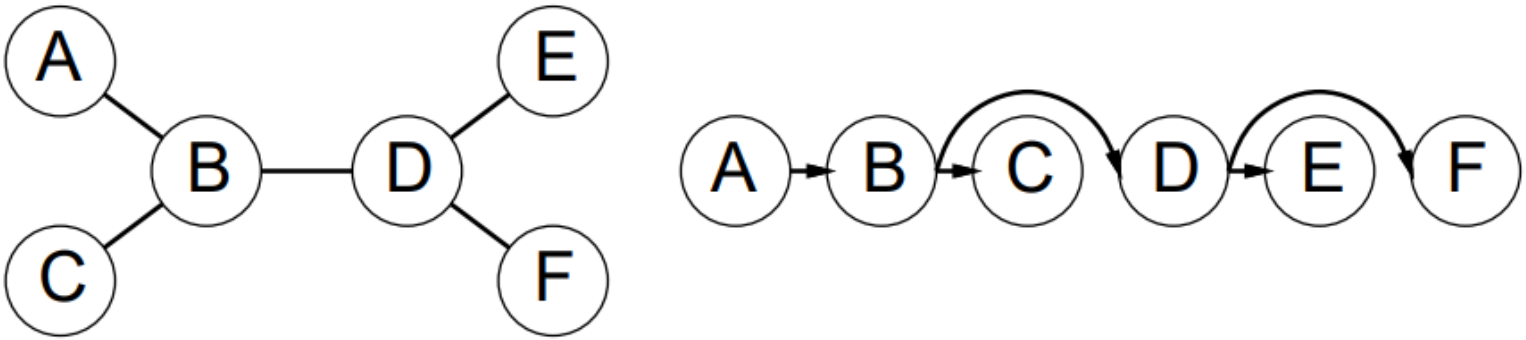
23

[23] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Tree Structured Graphs

- In a graph like the example on the last slide there are no loops in the constraint edges. In these kinds of cases the complexity can be reduced.
- In the case of a general CSP the worst-case time is $O(d^n)$
- This property also applies to the logical reasoning we have already seen earlier in the class.

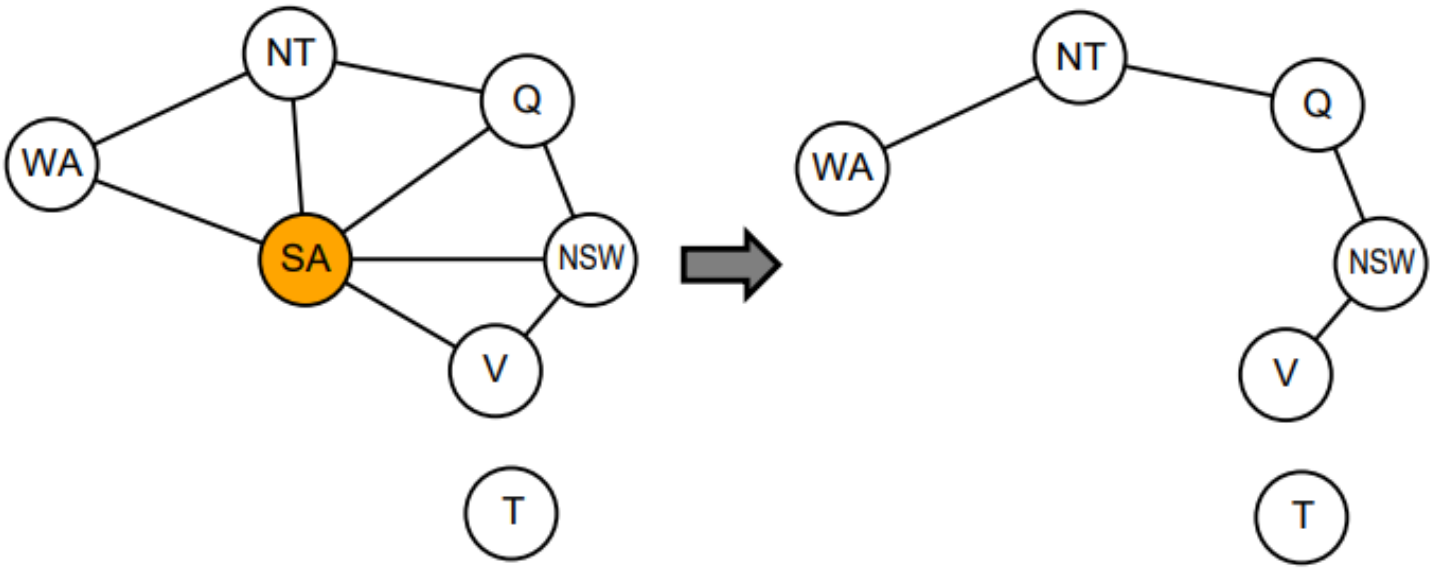
Tree Structured Graphs



24

[24] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Cutset Conditioning



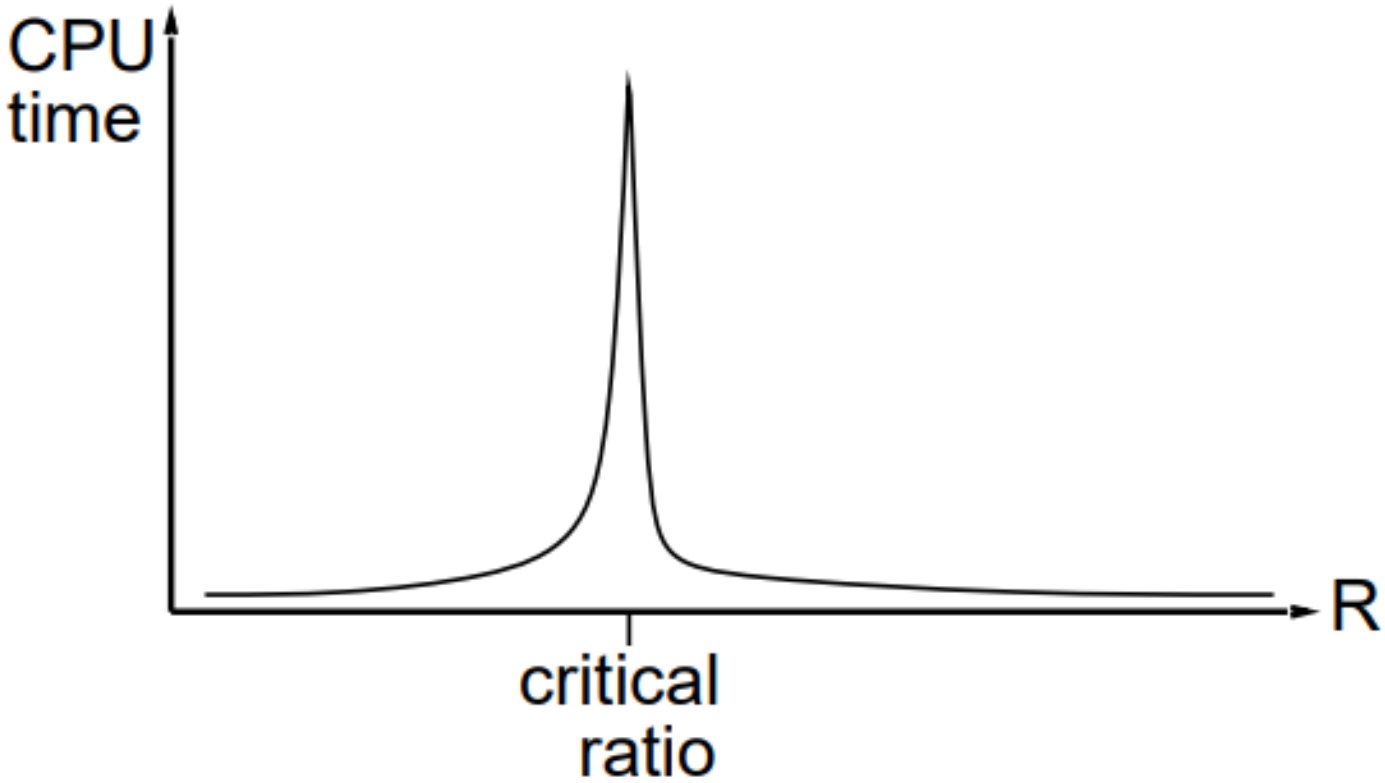
25

[25] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Iterative Algorithms for CSP Solving

- Hill Climbing and other similar algorithms generally are only working over sets of 'complete' states. ie all the variables are assigned to something.
- This doesn't work well for CSPs so to make it work we allow states to have unsatisfied constraints and allow the reassignment of variable values.
- When you select what 'next state' to attempt in the Hill Climb, we randomly select some variable that is conflicted in the current state.
- Then to select the value for that variable we simply pick a value that violates the fewest constraints. ie h(n) in your hill climb is just the total number of violated constraints.

Critical Ratio



26

[26] Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020

Sources

- Artificial Intelligence: A Modern Approach, Norvig and Russell, 2020
- Artificial Intelligence Course Slides, Fahiem Bacchus, 2014