Constraints Satisfaction Problems

Dra. Mª Dolores Rodríguez Moreno

Objectives

Specific Objectives

- Review what CSP is
- Main CSP techniques

Source

- Stuart Russell & Peter Norvig (2009). Chapter 6. Artificial Intelligence: A Modern Approach. (3rd Edition). Ed. Pearsons

Outline

- **Introduction**
- Example
- Techniques
  - Bactracking search in CSPs
    - Heuristics
    - Foward checking
    - Constraint Propagation
  - Local search in CSPs
- Conclusions

Introduction

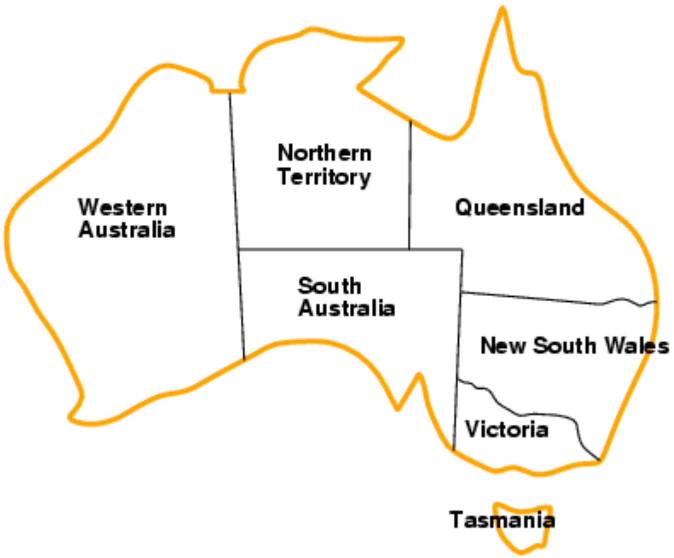
- A CSP is defined by:
  1. A set of variables {V1,...,Vn}
  2. A domain of values for each variable Dom [Vi]
  3. A set of constraints {C1,...,Cm}
- A goal test is a set of constraints specifying allowable combinations of values for subsets of variables
- An assignment that does not violate any restriction is called a consistent assignment
- Some problems require a solution that maximizes an objective function

Outline

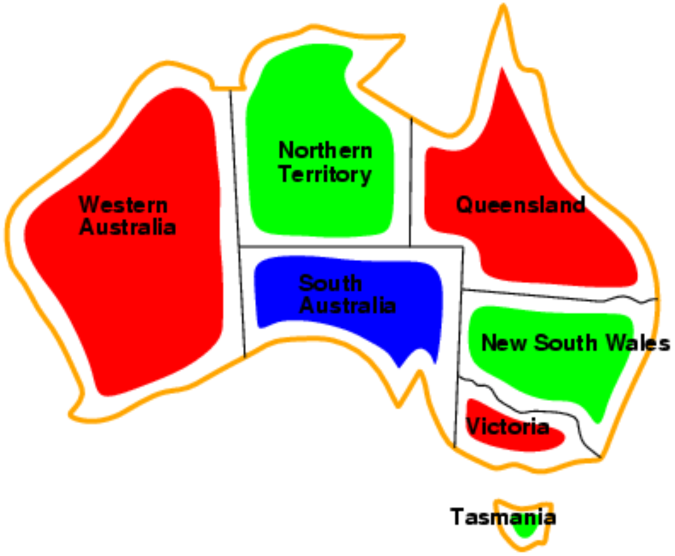
- Introduction
- **Example**
- Techniques
  - Bactracking search in CSPs
    - Heuristics
    - Foward checking
    - Constraint propagation
  - Local search in CSPs
- Conclusions

Example (I)

- Variables WA, NT, Q, NSW, V, SA, T
- Domains Di = {red,green,blue}
- Constraints: adjacent regions must have different colors
- e.g., WA $\neq$ NT, or (WA,NT) in {(red,green),(red,blue),(green,red), (green,blue),(blue,red), (blue,green)}



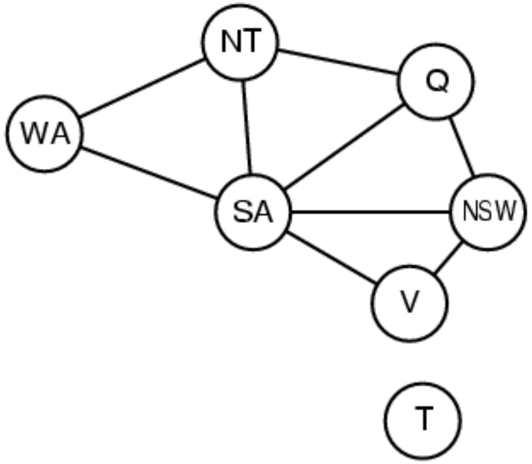
Example (II)



- Solutions are complete and consistent assignments, e.g., WA = red, NT = green,Q = red,NSW = green,V = red, SA = blue,T = green

Example (III)

- Binary CSP: each constraint relates two variables
- Constraint graph: nodes are variables, arcs are constraints



Varieties of CSPs

- There is a great variety of CSPs
- CSP with discrete variables
  - Finite domains
    - n variables, domain size $d \rightarrow O(d^n)$ complete assignments
    - eg. Boolean CSPs (var. can be T/F) although it can include satisfiability problemas where NP-complete (exponential time)
  - Infinite domains
    - Integers, strings, etc.
    - e.g., job scheduling, variables are start/end days for each job
    - Need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
- CSP with continous variables
  - e.g., start/end times for Hubble Space Telescope observations
  - linear constraints solvable in polynomial time by linear programming

Type of constraints

- Unary constraints involve a single variable,
  - e.g., SA $\neq$ green

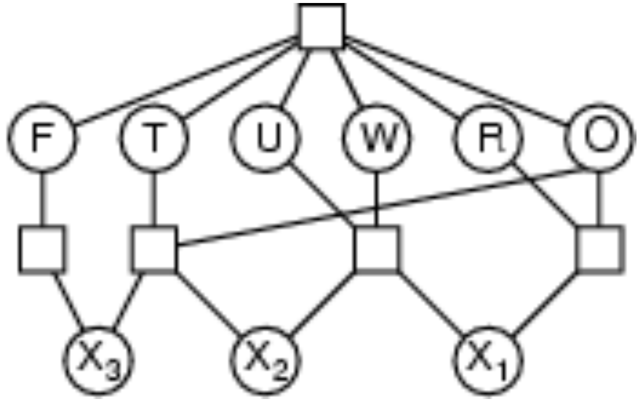
- Binary constraints involve pairs of variables,
  - e.g., SA $\neq$ WA

- Higher-order constraints involve 3 or more variables,
  - e.g., cryptarithmetic column constraints

Type of constraints: Higher-order

- Each letter represents a different digit

```
   T W O
 + T W O
 -------
 F O U R
```



- Variables: *F T U W R O*
  - Auxiliar : $X_1\ X_2\ X_3$
    - *Represent 0 or 1 transfered to the next column*
- Domain: {*0,1,2,3,4,5,6,7,8,9*}
- Constraints: *Alldiff(F,T,U,W,R,O) or also* F ≠ T, F ≠ U, …
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F,\ T \neq 0,\ F \neq 0$

Real-world CSPs

- Assignment problems
  - e.g., who teaches what class
- Timetabling problems
  - e.g., which class is offered when and where?
- Transportation scheduling
- Factory scheduling
- Notice that many real-world problems involve real-valued variables

Outline

- Introduction
- Example
- Techniques
  - **Bactracking search in CSPs**
    - Heuristics
    - Foward checking
  - Local search in CSPs
- Conclusions

Introduction (I)

- States are defined by the values assigned so far
- Every solution appears at depth n with n variables
  → use depth-first search
- The states are defined by the assigned values (**incremental formulation**):
  - Initial state: the empty assignment { }
  - Successor function: assign a value to an unassigned variable that does not conflict with current assignment → fail if there is no legal assignments
  - Goal test: the current assignment is complete
  - Cost path: a constant cost (eg 1) for each step
- Path is irrelevant, so we can also use **complete-state formulation** (each state is a complete assignment that may or may not meet the restrictions)

Introduction (II)

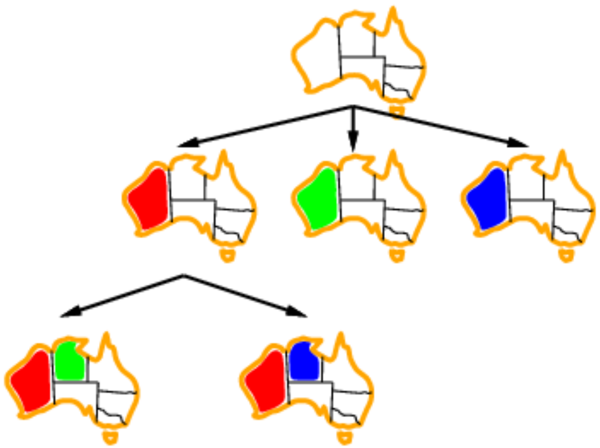
- Variable assignments are commutative, i.e.,

  [WA = red then NT = green] same as [NT = green then WA = red]
- Only need to consider assignments to a single variable at each node
- Depth-first search for CSPs with single-variable assignments is called backtracking search
- Backtracking search is the basic uninformed algorithm for CSPs
- Can solve n-queens for $n \approx 25$

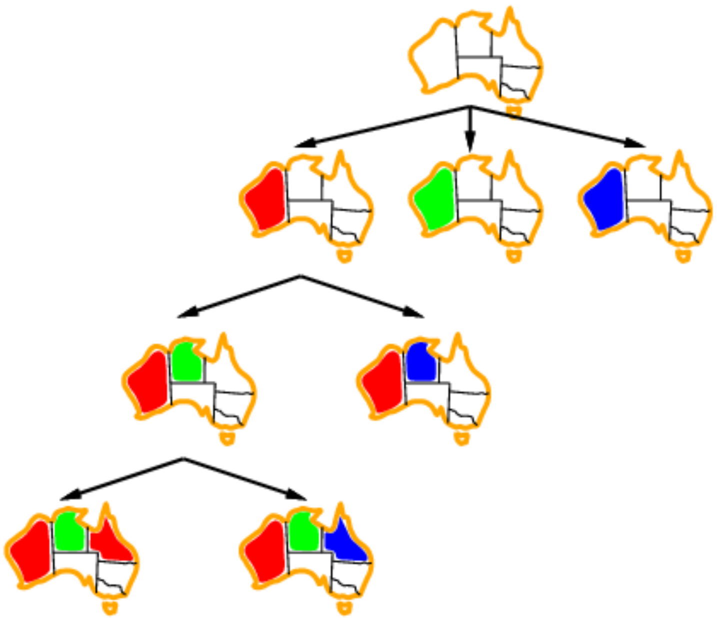
Backtracking search

Backtracking search

Backtracking search



Backtracking search



Improve Backtracking search

- General-purpose methods (heuristics) can give huge gains in speed:
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?

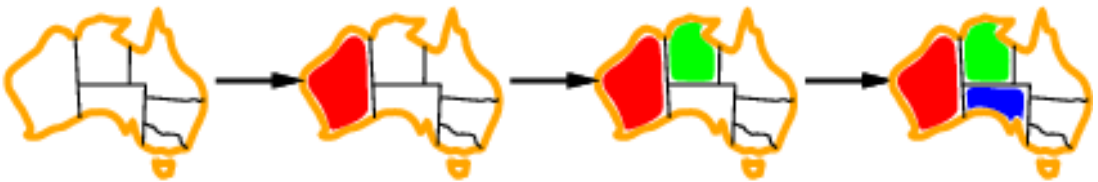
- Let's see 3 heuristics

Outline

- Introduction
- Example
- Techniques
  - Bactracking search in CSPs
    - **Heuristics**
    - Foward checking
    - Constraint propagation
  - Local search in CSPs
- Conclusions

Heuristics: MRV

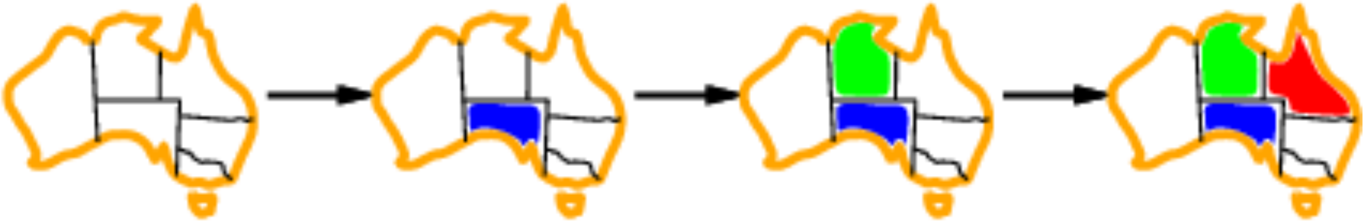
- Most constrained variable:
  - Choose the variable with the fewest legal values
  - If X has 0 remaining legal values, selecting X will prevent assignments to other variables that fail when X is chosen



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

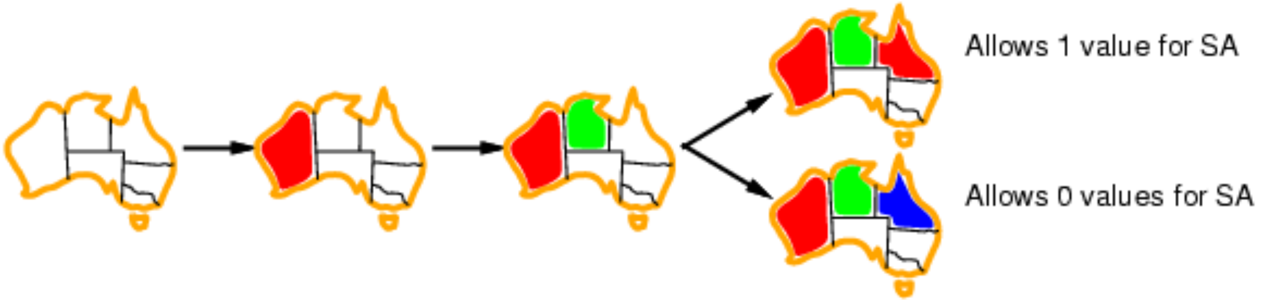
Heuristics (number of constraints)

- Tie-breaker among most constrained variables
- Most constraining variable:
  - choose the variable with the most constraints on remaining variables



Heuristics (LCV)

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



- Combining these heuristics makes 1000 queens feasible

Outline

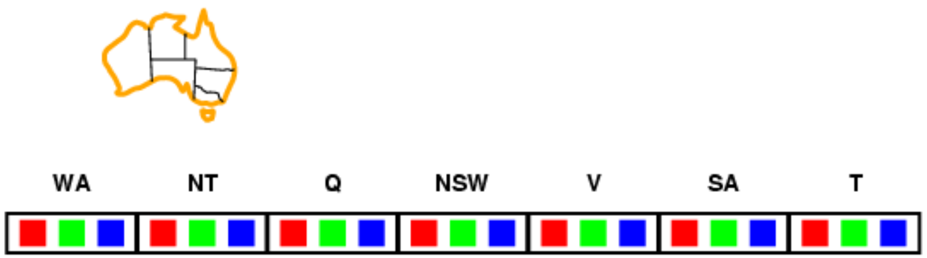
- Introduction
- Example
- Techniques
  - Bactracking search in CSPs
    - Heuristics
    - **Foward checking**
    - Constraint propagation
  - Local search in CSPs
- Conclusions

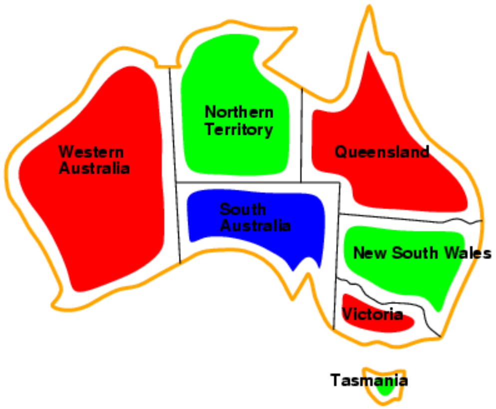
Introduction

- If we look at certain restrictions early or before the search starts, we can drastically reduce the search space
- There are 2 techniques:
  - Forward checking
  - Constraints Propagation

Foward checking

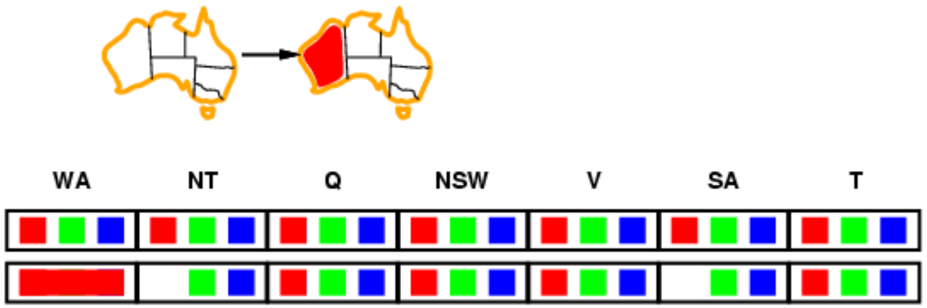
- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values

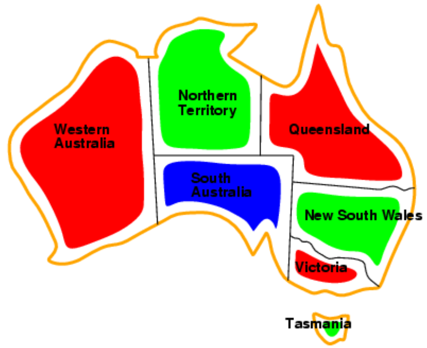




Foward checking

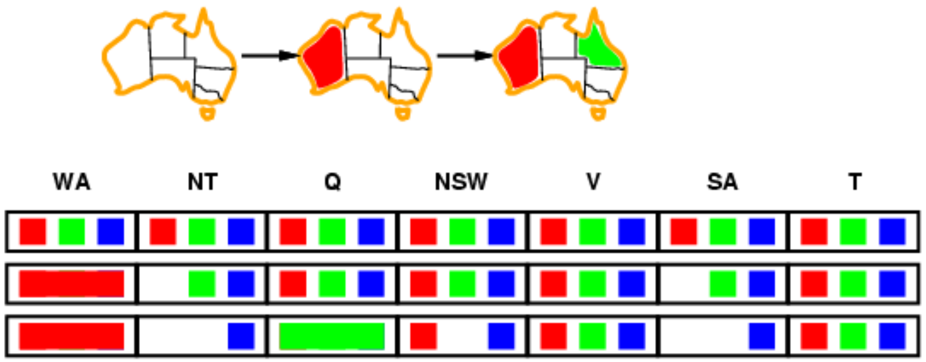
- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values

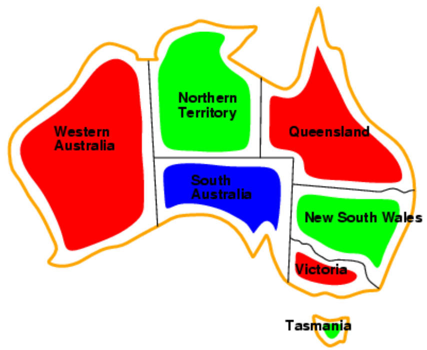




Foward checking

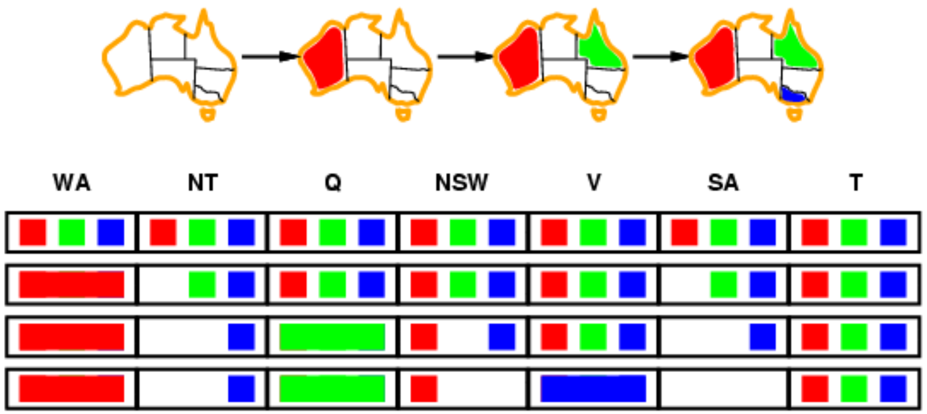
- Idea:
    - Keep track of remaining legal values for unassigned variables
    - Terminate search when any variable has no legal values

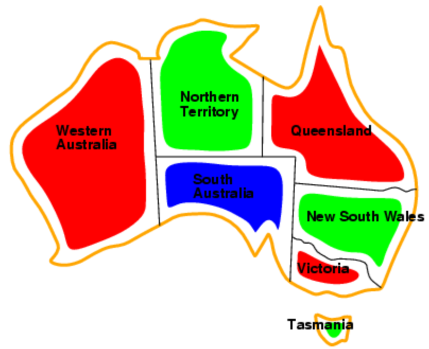




Foward checking

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values





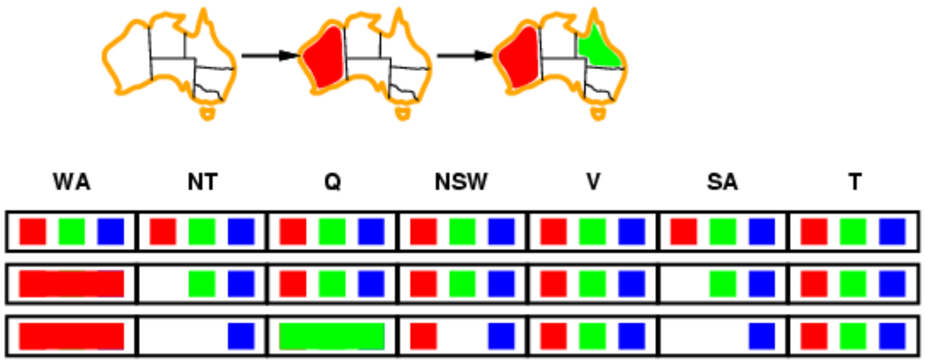
- Instead of V, which would have been a better candidate according to the heuristic views? SA or NT

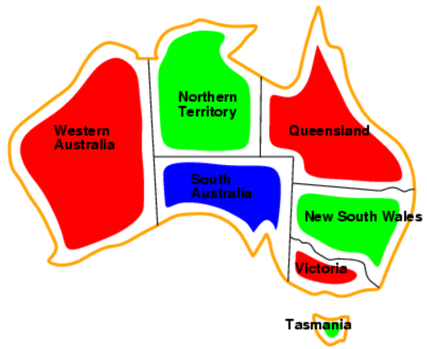
Outline

- Introduction
- Example
- Techniques
  - Bactracking search in CSPs
    - Heuristics
    - Foward checking
    - **Constraint propagation**
  - Local search in CSPs
- Conclusions

Constraint propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:

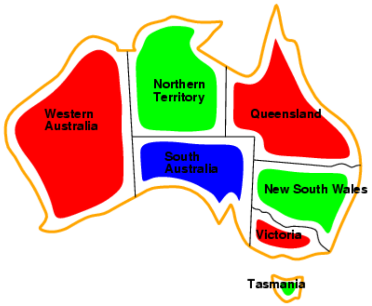


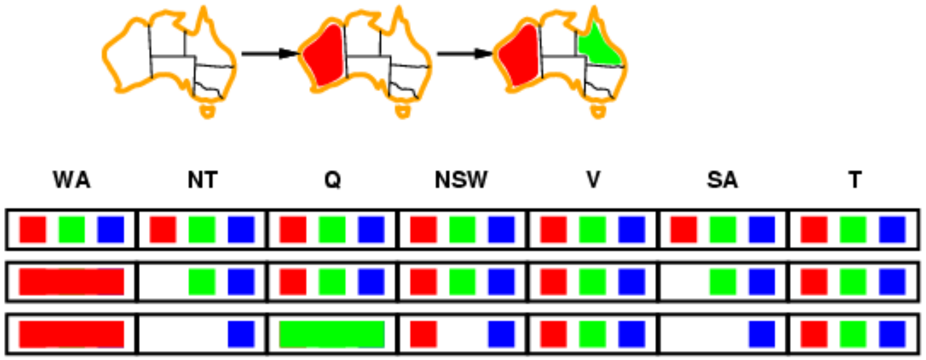


- NT and SA cannot both be blue!
- Constraint propagation repeatedly enforces constraints locally

Constraint propagation

- We need to propagate from WA and Q to NT and SA (as we did in forward checking) and then, in the constraints between NT and SA, to detect the inconsistency

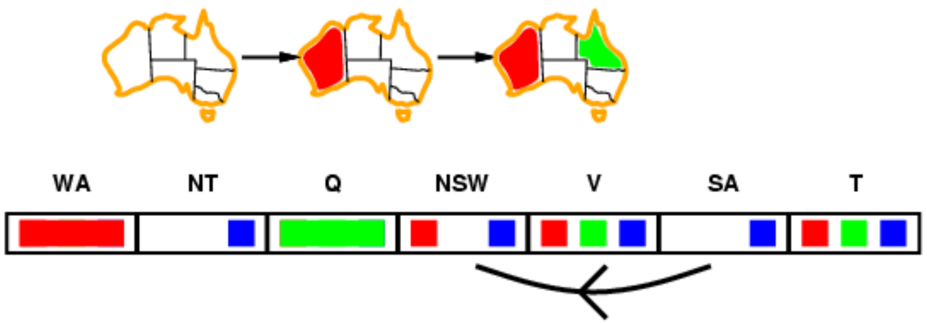


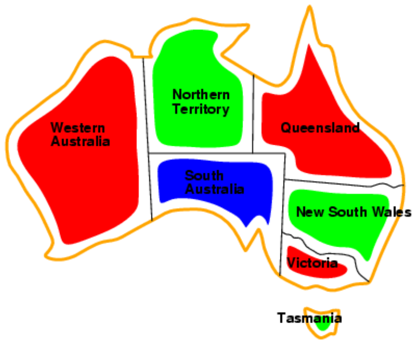


- The way to do it has to be fast: the propagation time cannot be longer than the simple search
- Arc consistency

Arc consistency

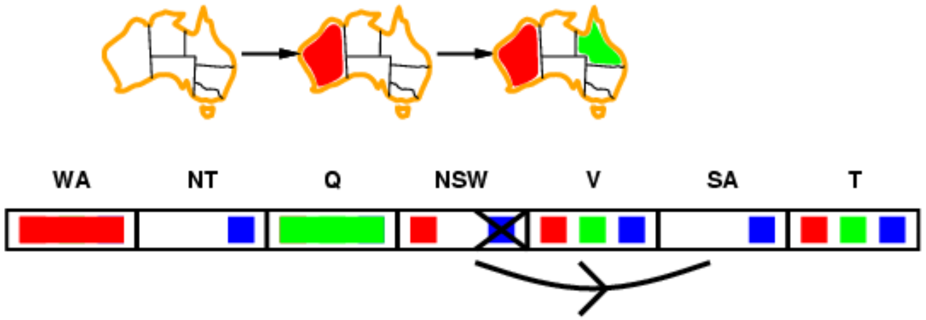
- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

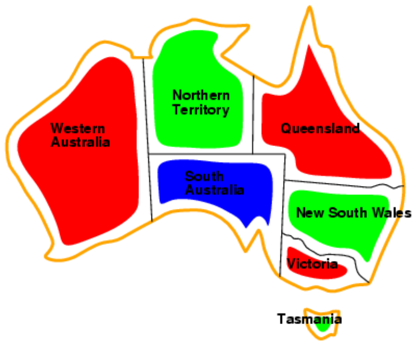




Arc consistency

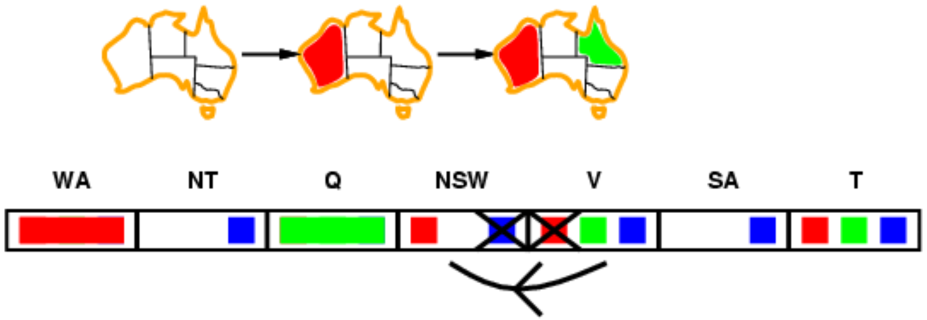
- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

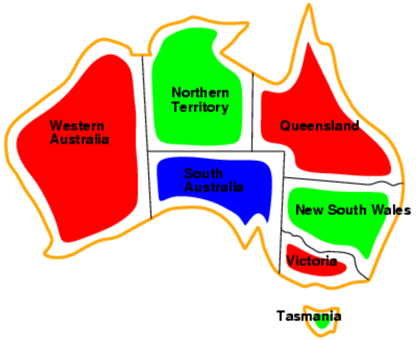




Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y

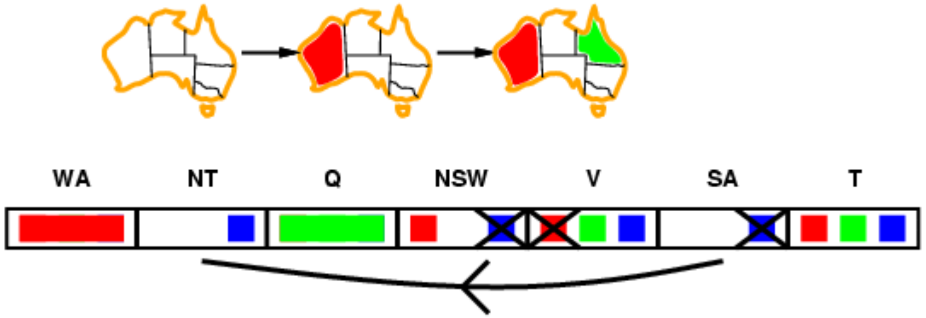


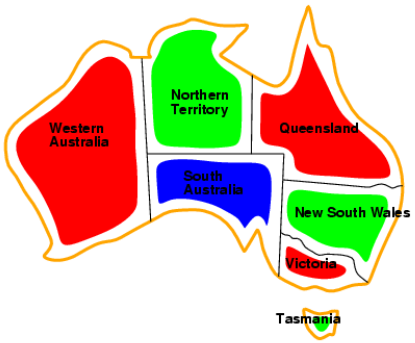


- If X loses a value, neighbors of X need to be rechecked

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y





- If X loses a value, neighbors of X need to be rechecked
- Arc consistency detects failure earlier than forward checking
- Can be run as a preprocessor or after each assignment

Outline

- Introduction
- Example
- Techniques
  - Bactracking search in CSPs
    - Heuristics
    - Foward checking
    - Constraint Propagation
  - **Local search in CSPs**
- Conclusions

Local search in CSPs

- Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
  - Allow states with unsatisfied constraints
  - Operators reassign variable values
- Variable selection: randomly select any conflicted variable
- Value selection by min-conflicts heuristic:
  - Choose value that violates the fewest constraints
  - i.e., hill-climb with $h(n)$ = total number of violated constraints

Outline

- Introduction
- Example
- Techniques
  - Bactracking search in CSPs
    - Heuristics
    - Foward checking
    - Constraint Propagation
  - Local search in CSPs
- **Conclusions**

Conclusions

- CSPs are a special kind of problem:
  - States defined by values of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that guarantee later failure
- Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies