CS440/ECE448: Intro to Artificial Intelligence

Lecture 5: Constraint satisfaction problems

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440

Thursday's key concepts

Heuristic search:
Actions and solutions have costs
Heuristic function: estimate of future cost
Uniform cost, best-first, A*

Local search:
Agent only sees the next steps.
Features of the state space landscape
Hill-climbing, random restart, beam, simulated annealing

Constraint satisfaction problems

Today's topics/questions

Different kinds of CSPs:

- Binary vs. global constraints
- Small finite vs. large/continuous domains

How do constraints interact?

- The structure of CSPs

Underdetermined CSPs (multiple solutions):

- Interleaving search and CSP inference

CSP 1:
Map coloring
(Binary constraints)

Map coloring

Task:

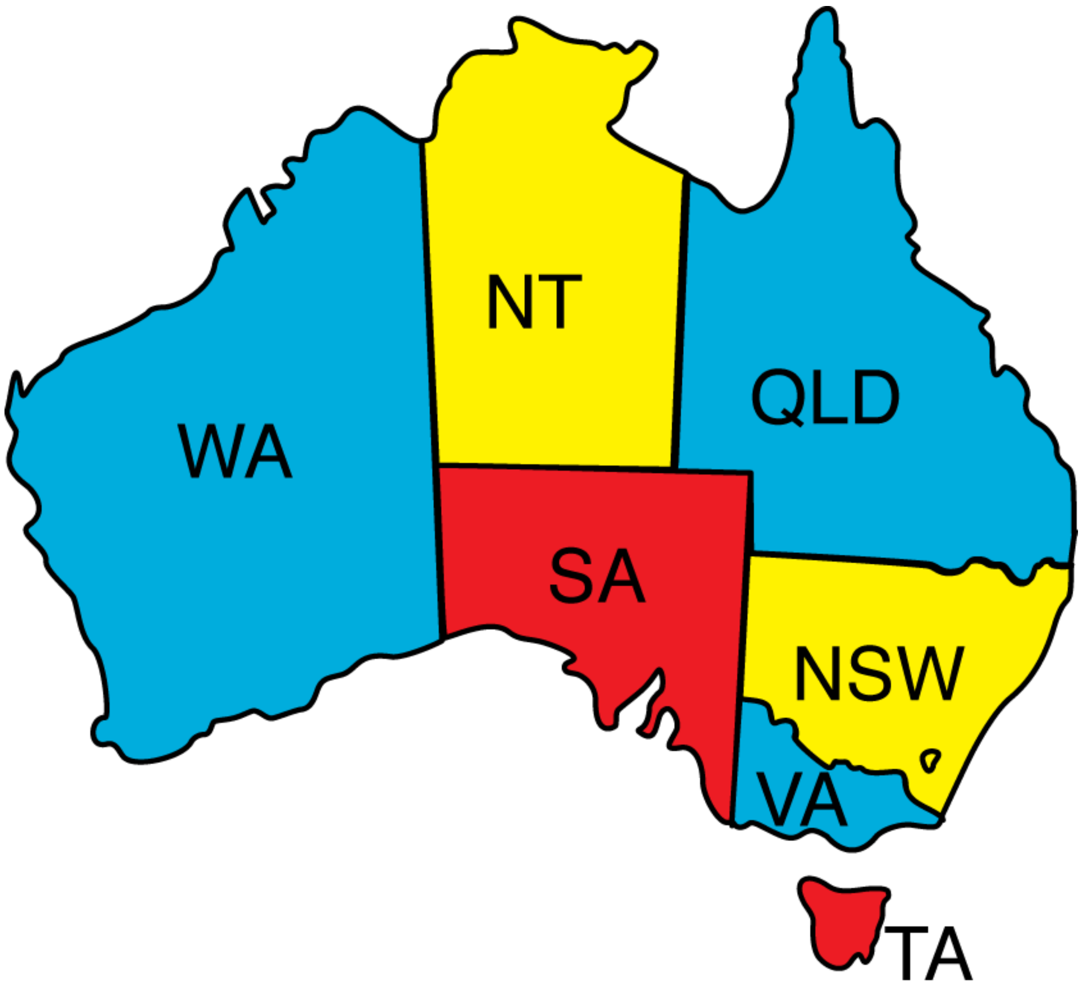
Color each region of the map with one of N colors.

Constraints:

No neighboring regions have the same color

Map coloring

Map coloring: a solution for N=3



Map coloring is defined by…

- the **regions** on the map:

  {WA, NT, QLD, NSW, VA, SA, TA}

- the **colors**:

  {red, blue, green}

- the neighbor **constraints**:

  {WA ≠ NT, WA ≠ SA, NT ≠ QLD,
  NT ≠ SA, QLD ≠ NSW, QLD ≠ SA,
  NSW ≠ VA, NSW ≠ SA, VA ≠ SA}

Map coloring as search

The state space is defined by:

– a set of **variables** (the regions):
  {WA, NT, QLD, NSW, VA, SA, TA}

– the **domain** of each variable
  (its set of possible values):
  D_{WA}= *{red, blue, green}*
  D_{NT}= *{red, blue, green}*
  …

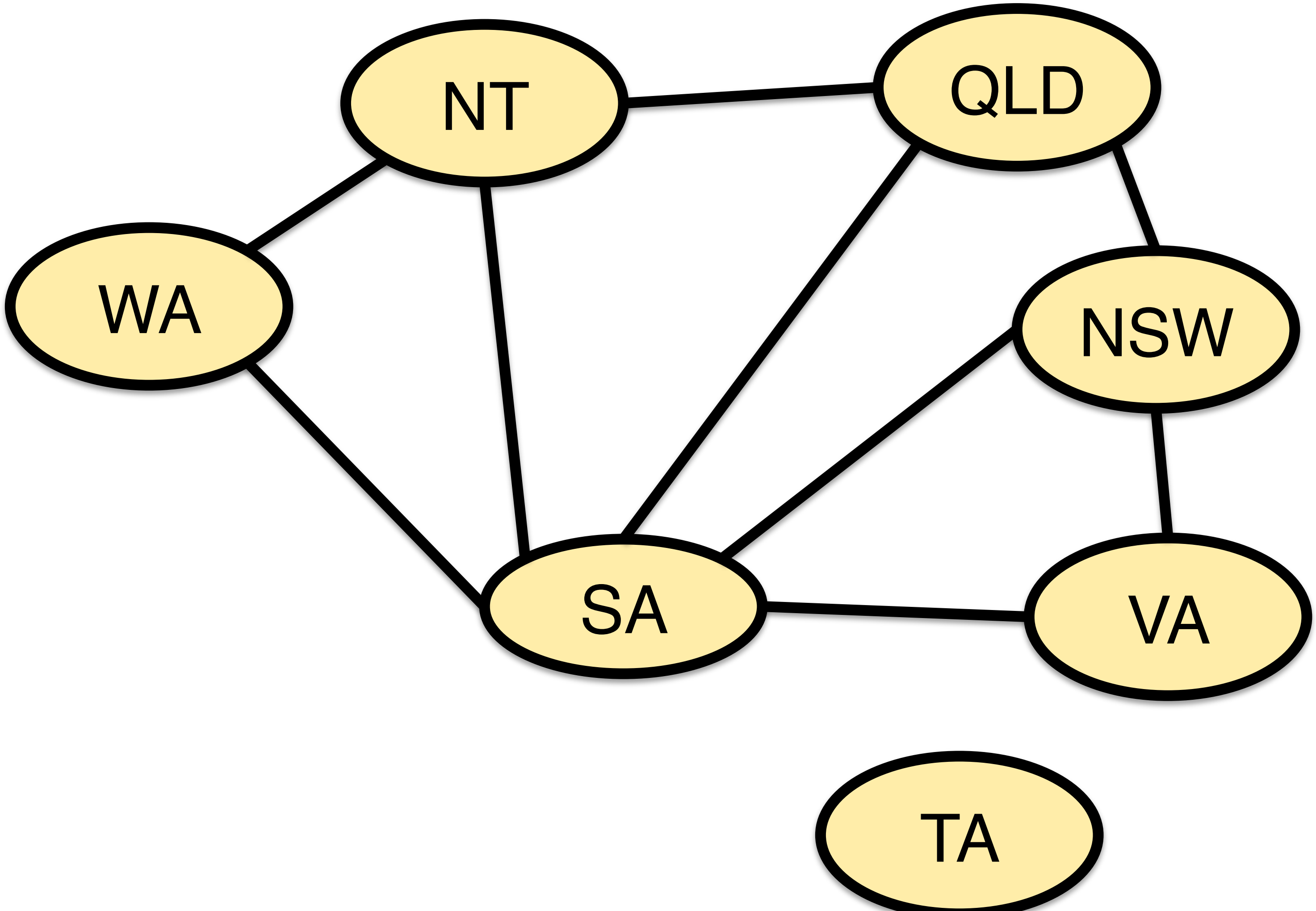
Map coloring as search

Each **state** is a complete or partial assignment of values to variables:

state35 = {WA=red, NT=blue, QLD= green, NSW= red,
VA= green, SA= blue, TA= red};
state23 = {WA = red}

Legal assignments don't violate any constraints.
Solutions are complete legal assignments

Binary constraints: constraint graph



Constraint satisfaction problems are defined by…

- **a set of variables** X:

  $\{WA, NT, QLD, NSW, VA, SA, TA\}$

- **a set of domains** D_i (possible values for variable x_i):

  $D_{WA} = \{red, blue, green\}$

- **a set of constraints** C:

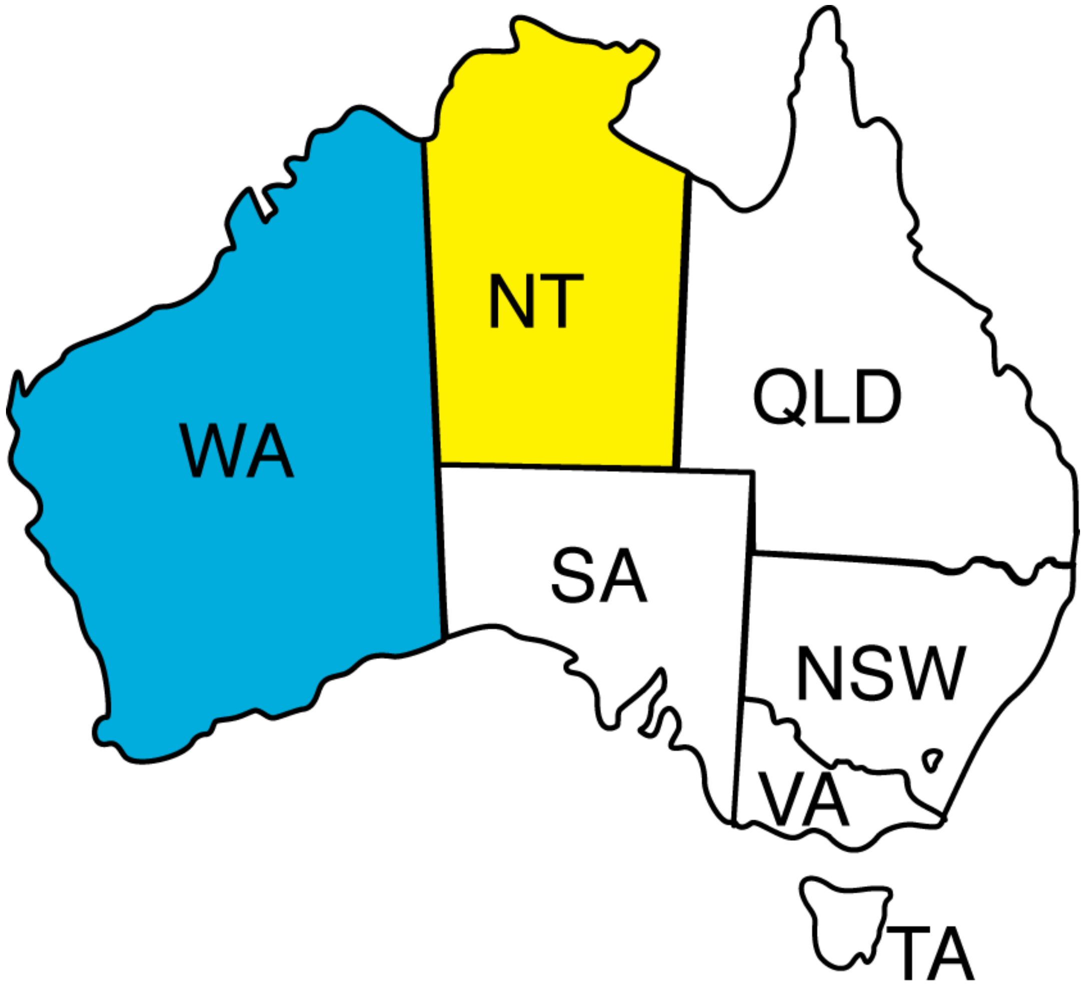
  $\{\langle (WA,NT), WA \neq NT \rangle, \langle (WA,QLD), WA \neq QLD \rangle, \ldots\}$

  scope *relation*

Inference in CSPs: Constraint propagation

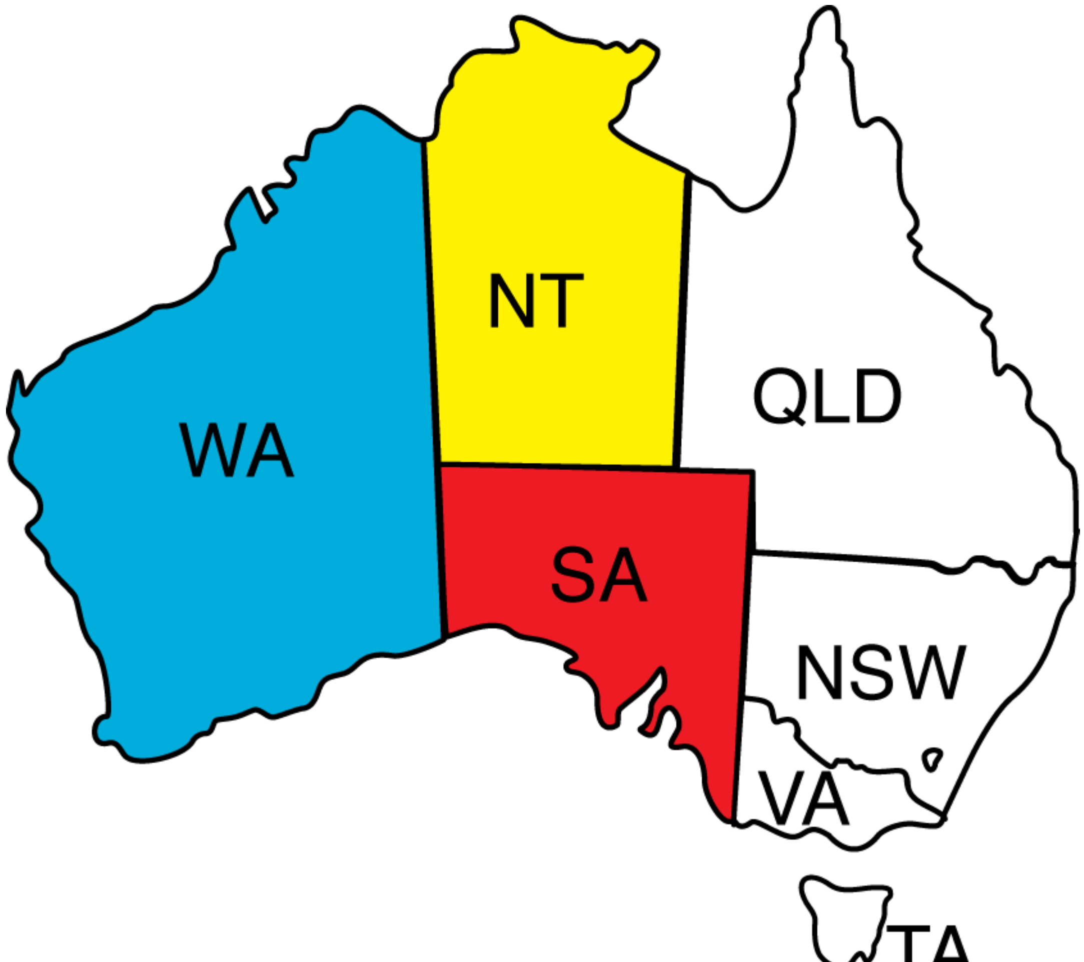
Map coloring

$D_{WA} = \{B\}$
$D_{NT} = \{Y\}$
$D_{SA} = \{R,B,Y\}$
$D_{QLD} = \{R,B,Y\}$
$D_{NSW} = \{R,B,Y\}$
$D_{VA} = \{R,B,Y\}$
$D_{TA} = \{R,B,Y\}$



Map coloring

$D_{WA} = \{B\}$
$D_{NT} = \{Y\}$
$D_{SA} = \{R, \cancel{B}, \cancel{Y}\}$
$D_{QLD} = \{R,B,Y\}$
$D_{NSW} = \{R,B,Y\}$
$D_{VA} = \{R,B,Y\}$
$D_{TA} = \{R,B,Y\}$



Map coloring

$D_{WA} = \{B\}$

$D_{NT} = \{Y\}$

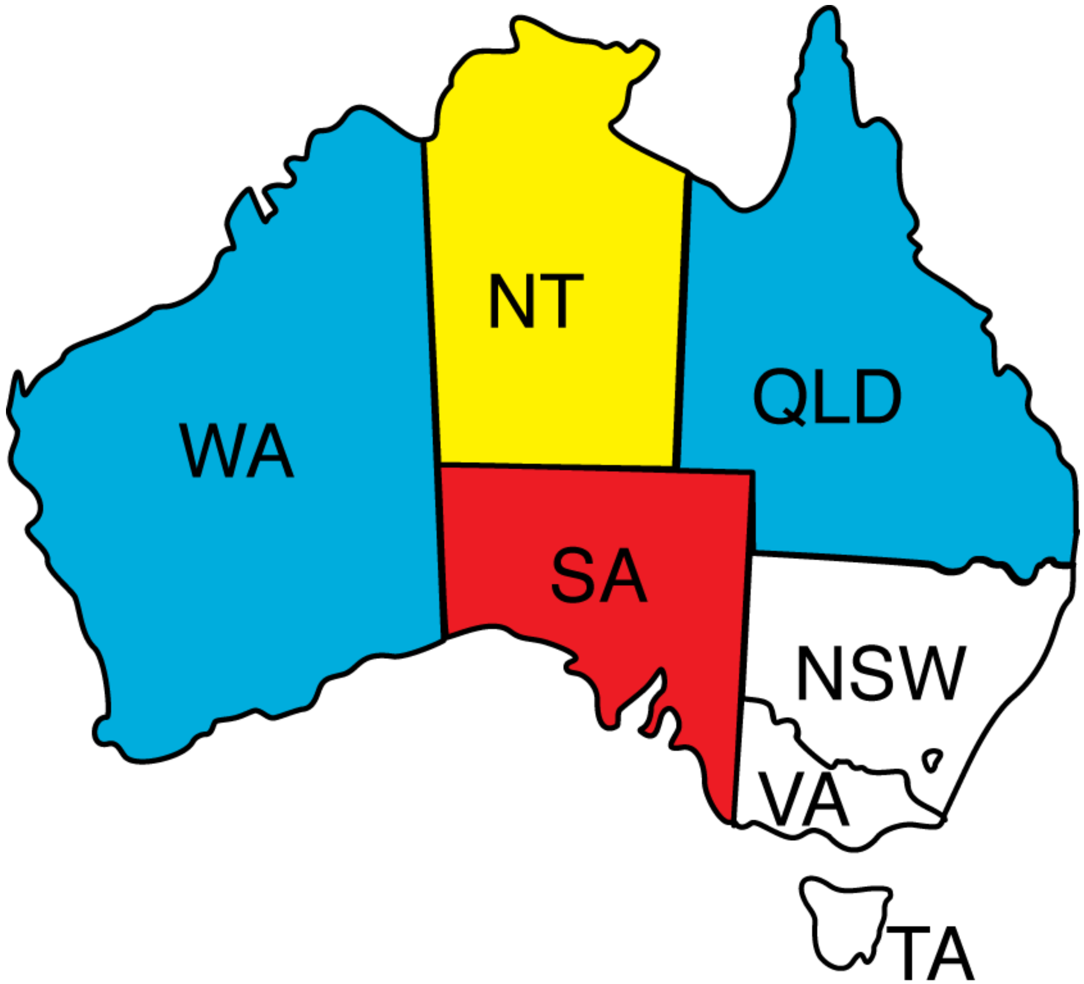
$D_{SA} = \{R\}$

$D_{QLD} = \{\sout{R},B,\sout{Y}\}$

$D_{NSW} = \{R,B,Y\}$

$D_{VA} = \{R,B,Y\}$

$D_{TA} = \{R,B,Y\}$



Map coloring

$D_{WA} = \{B\}$

$D_{NT} = \{Y\}$

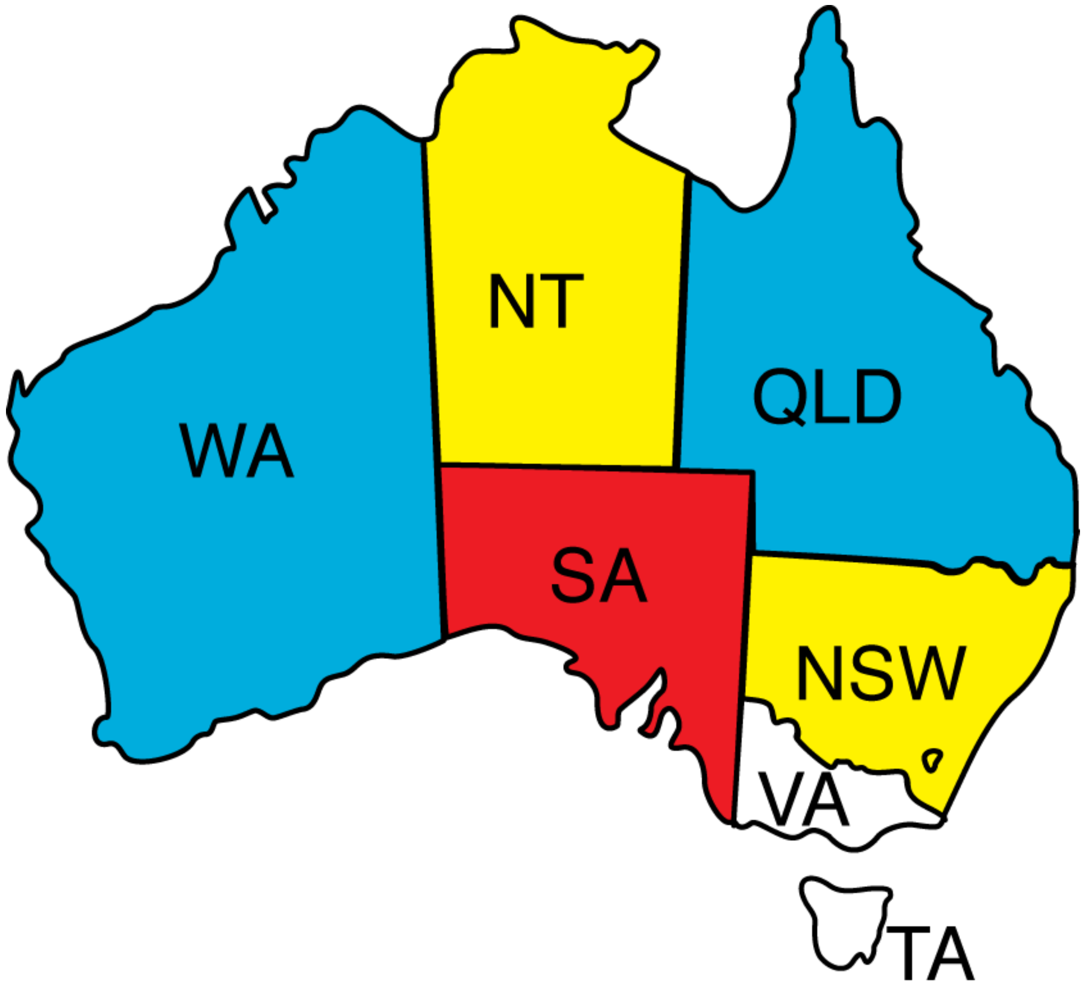
$D_{SA} = \{R\}$

$D_{QLD} = \{B\}$

D_{NSW} = {~~R~~,~~B~~,Y}

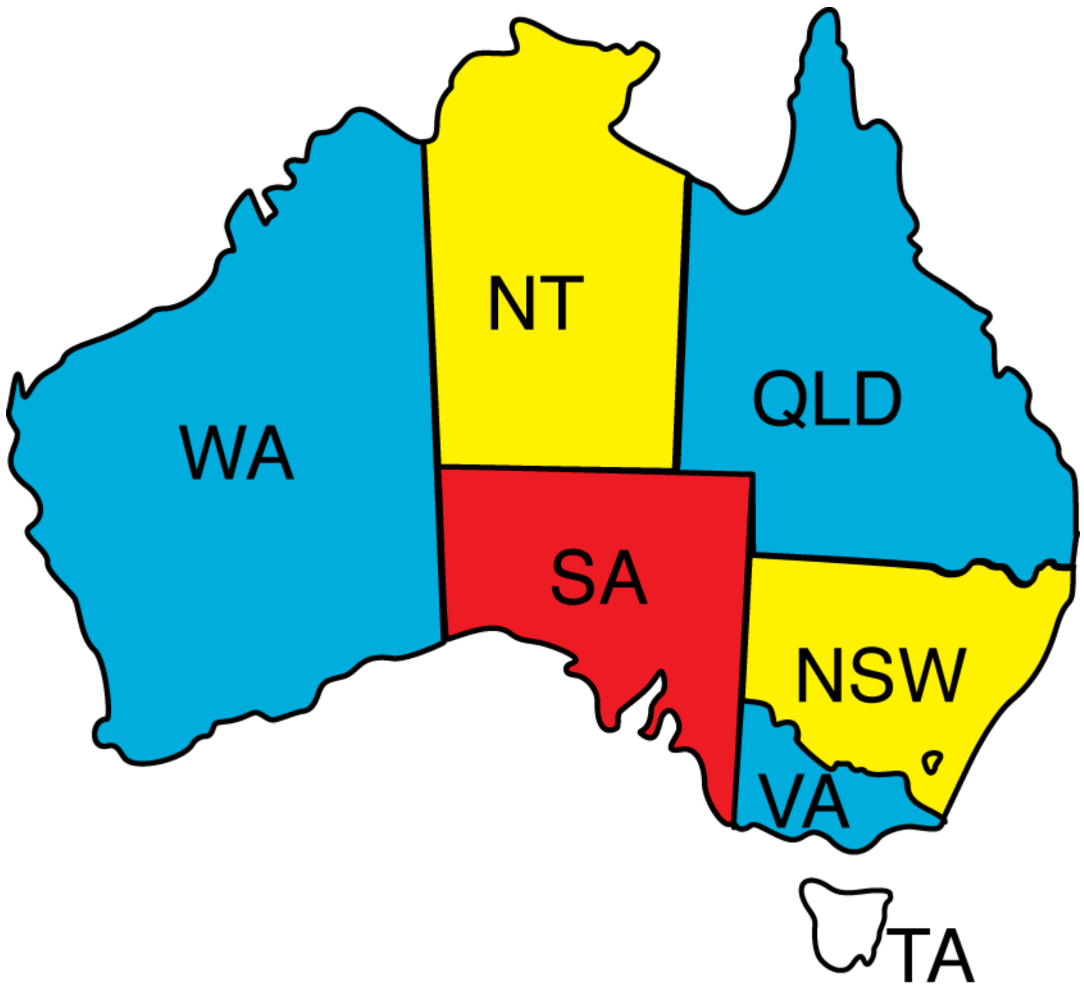
$D_{VA} = \{R,B,Y\}$

$D_{TA} = \{R,B,Y\}$



Map coloring

D_{WA} = {B}
D_{NT} = {Y}
D_{SA} = {R}
D_{QLD} = {B}
D_{NSW} = {Y}
D_{VA} = {~~R~~,B,~~Y~~}
D_{TA} = {R,B,Y}



Unary constraints: Node consistency

A unary constraint:
Western Australia is blue.
The final inspection takes 10 minutes

Expressed as constraint:
WA = blue; FI + 10 ≤ *5:00pm*

Expressed as restriction on the domain:
$D_{WA} = \{blue\}, \; D_{FI} = \{8{:}00am \ldots 4{:}50pm\}$

A single variable is node-consistent if all the values in its domain satisfy all its unary constraints.

Binary constraints: Arc consistency

A variable x_i is **arc-consistent** if and only if for *every* value $d_i \in D_i$ in its own domain and for every binary constraint $C(x_i, x_j)$, there is a value $d_j \in D_j$ in x_j's domain such that C is satisfied.

$D_X = D_Y = \{0,1,2,3,4,5,6,7,8,9\}$,
Constraint: $C(X, Y)$: $Y = X^2$
Arc-consistency ➔ $D_X = \{0, 1, 2, 3\}$, $D_Y = \{0,1,4,9\}$

The AC-3 algorithm:

constraint propagation for binary CSPs with finite domains

Revise(CSP c, var X, var Y)

```
//does C(X,Y) require a change in domain(X)?
Function revise(CSP c, var X, var Y)
   local: boolean revised ← false;
   foreach x in domain(X) do:
      // assigning x to X can't be legal
      if no value in domain(Y)
         satisfies C(X,Y):
         // side effect: change domain(X)
         delete x from domain(X);
         revised ← true;
   return revised;
```

AC-3

```
// Is the CSP c arc-consistent?
function AC3(CSP c)
   input: CSP c = (X,D,C)
   local: queue q ← all arcs C(X,Y) in c
   while q ≠ () do:
      // Can C(X,Y) be satisfied?
      (X,Y) = pop(q);
      // Change domain(X) if necessary
      if revise(c,X,Y):
         // Exit if CSP can't be solved:
         if domain(X) == () return false;
         // Are X's neighbors still okay?
         foreach Z in X.NEIGHBORS\{Y}:
            q ← push(q,(Z, X));
   return true;
```

Complexity of AC3: $O(cd^3)$

CSP with *n* variables, domains of size ≤ *d*, and *c* constraints:

There are *c* constraints C(X,Y).
Each *C(X,Y)* can be revised up to *d* times
(D$_X$ has at most *d* values)
So: each *C(Z,X)* can be added up to *d* times to queue

The consistency check of C(X,Y) is $O(d^2)$ time

Binary constraints interact

Can we color Australia with 2 colors?
No – SA, VA and NSW all border each other.

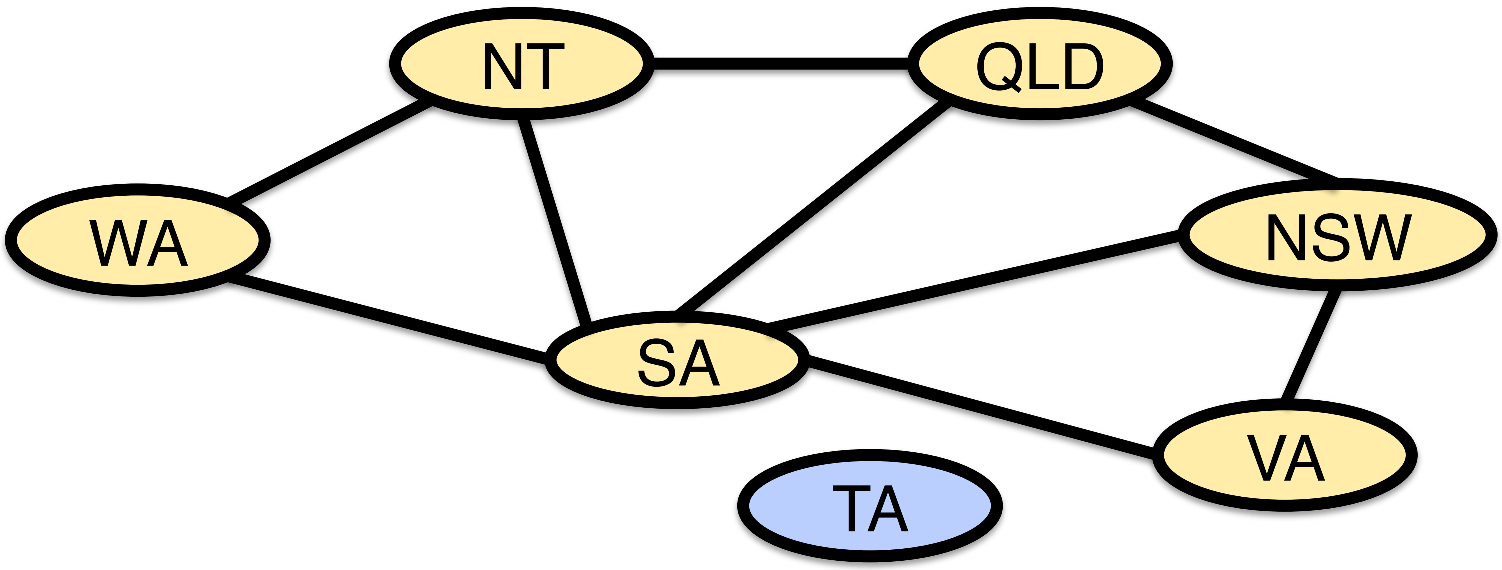
Arc consistency doesn’t catch
interactions between binary constraints

Interactions of binary constraints: Path consistency

A pair of variables {X,Y} is **path consistent** with respect to variable Z if and only if for every consistent x $\in D_X$ and y $\in D_X$ there is a z $\in D_Z$ that satisfies C(X=x,Z=z) and C(Y=y,Z=z)

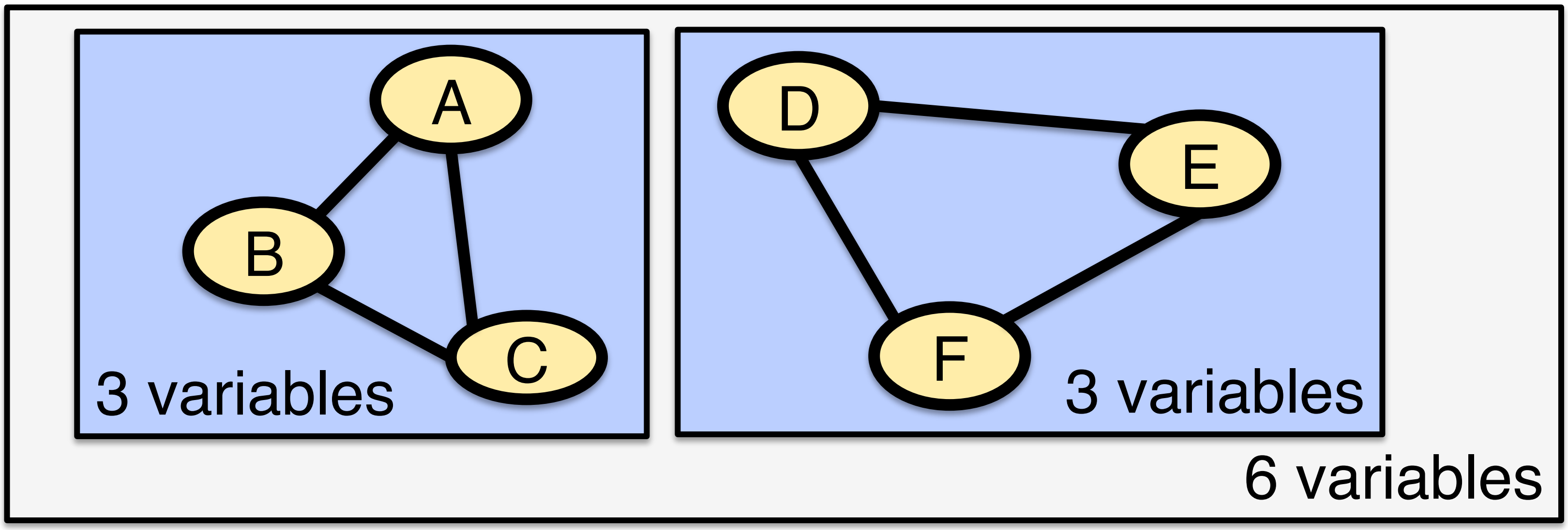
The structure of CSPs

Independent subproblems



This constraint graph consists of two **connected components**.
Each connected component corresponds to an independent subproblem.

Solving subproblems is faster



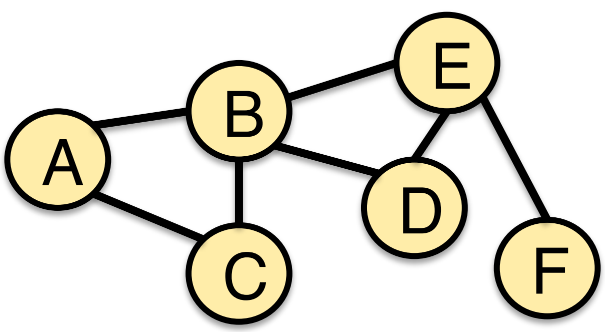
Solving a CSP with domain size *d:*

– with *n* variables: $O(d^n)$

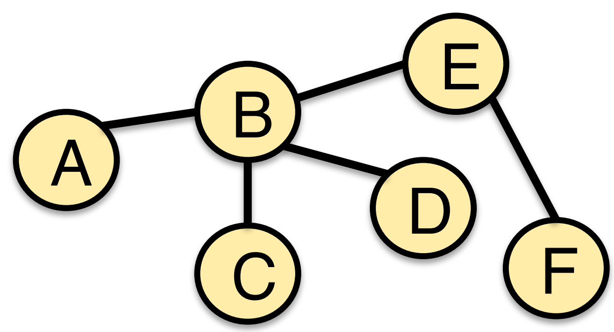
– divided into *n/c* subproblems with c variables: $O(n/c * d^c)$

Tree-structured constraint graphs

- Any two nodes connected by a single path
- With n vertices, there are n–1 edges
- Can be solved in linear time



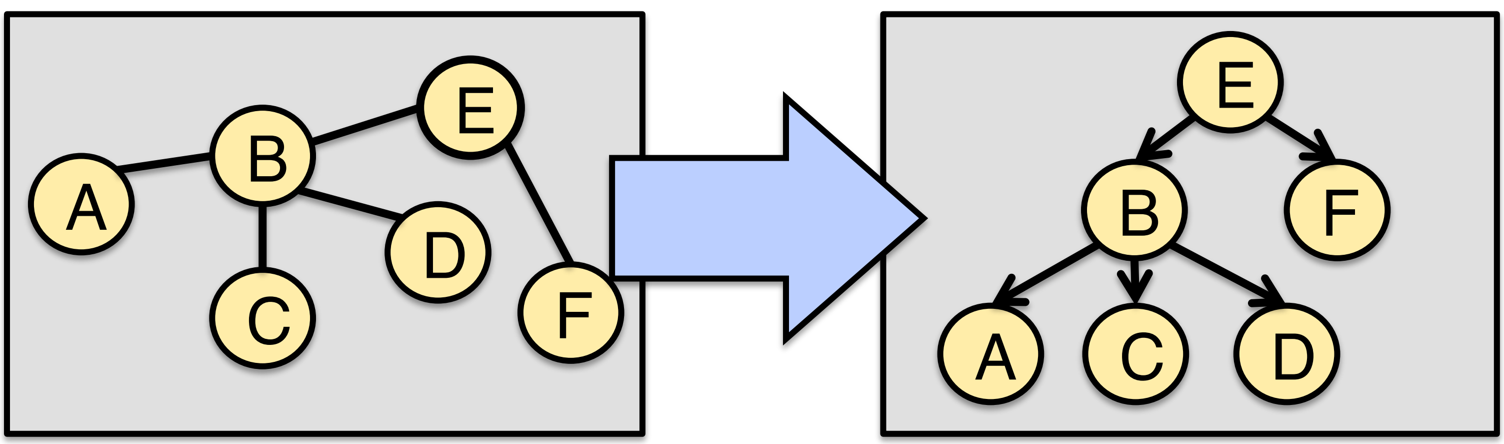
Not a tree



A tree

1. Topological sort
Create an ordered tree

1. Pick an (arbitrary node) v_1 as the root
2. For each undirected edge (v_1, u):
   a) Create directed edge $v_1 \rightarrow u$. Now $v_1 < u$.
   b) Recurse on u.



Directed arc consistency

A CSP is directed-arc consistent (DAC) under an ordering $X_1,\ldots,X_n$ if and only if every X_i is arc-consistent with any X_j for $j>i$

Solving a tree-shaped CSP

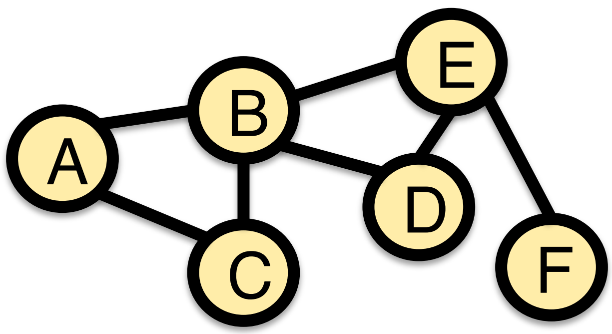
1. Make the tree-shaped CSP DAC: $O(nd^2)$

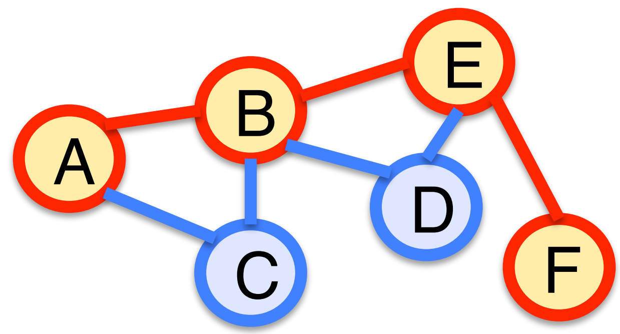
2. Pick a value for the root.

3. Find the corresponding value for its children.

4. Recurse on the children.

What about non-tree-shaped CSPs?





Unless the graph is fully connected, there will be a subset of nodes which form a tree.

Remainder := “cutset”

Cutset conditioning

1. Separate graph into cutset S and tree T
2. For each possible assignment to the variables in S:
   1. Resolve any constraints with variables in T
   2. If the remaining CSP for T has a solution, return it with the assignment to