Announcements

- Coding Homework 2 will be released
  - Due 2/8 at 11:59pm
- Written Homework 2 will be released
  - Due 2/8 at 11:59pm

Optimization: CSPs

Forest Agostinelli
University of South Carolina

Topics Covered in This Class

- **Part 1: Search**
  - Pathfinding
    - Uninformed search
    - Informed search
  - Adversarial search
  - Optimization
    - Local search
    - Constraint satisfaction
- **Part 2: Knowledge Representation and Reasoning**
  - Propositional logic
  - First-order logic
  - Prolog
- **Part 3: Knowledge Representation and Reasoning Under Uncertainty**
  - Probability
  - Bayesian networks
- **Part 4: Machine Learning**
  - Supervised learning
    - Inductive logic programming
    - Linear models
    - Deep neural networks
    - PyTorch
  - Reinforcement learning
    - Markov decision processes
    - Dynamic programming
    - Model-free RL
  - Unsupervised learning
    - Clustering
    - Autoencoders

Outline

- Review
- Arc consistency
- Selecting unassigned nodes
- Ordering assignment of values
- Local search
- Special cases

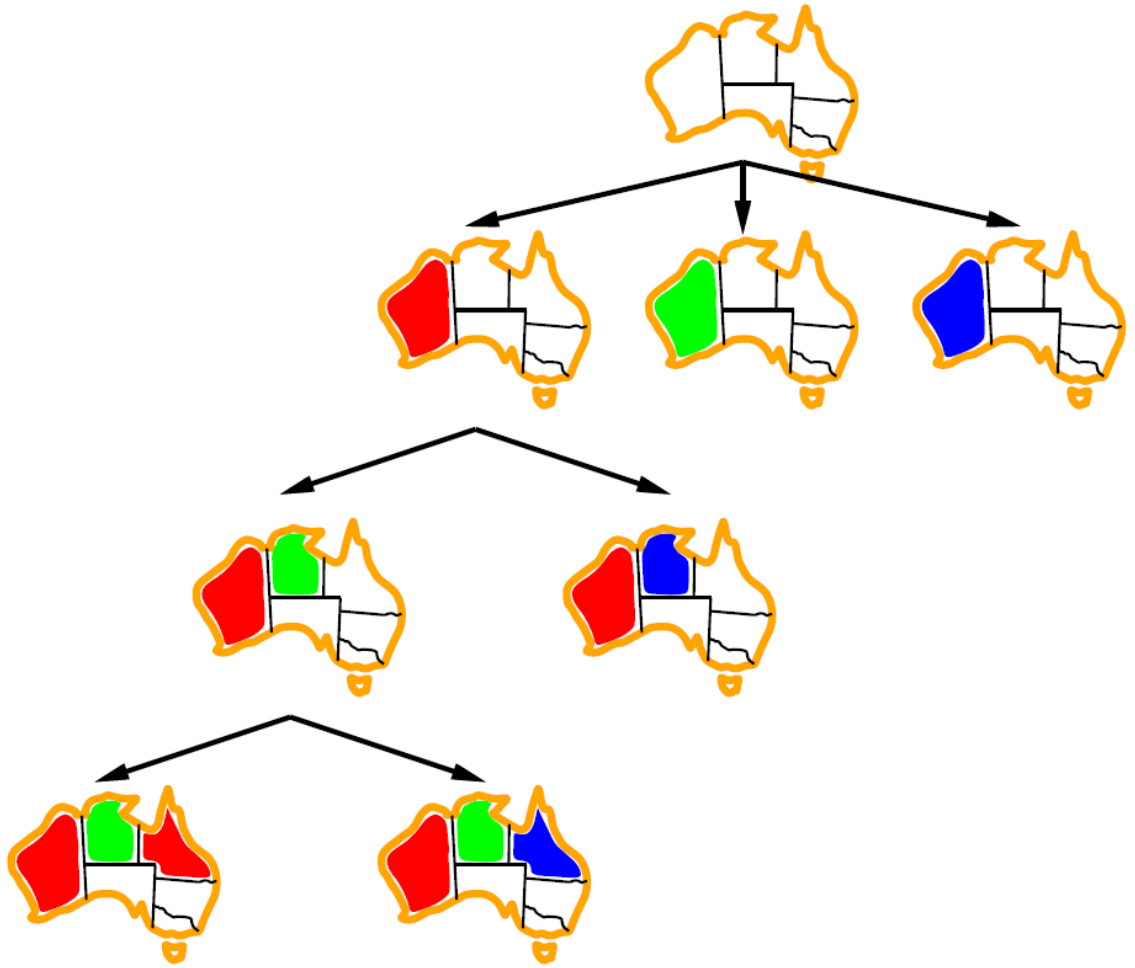
Constraint Satisfaction Problems

- $\mathcal{X}$ is a set of variables
- $\mathcal{D}$ is a set of domains, one for each variable
  - Allowable values
- $\mathcal{C}$ is a set of constraints that specify allowable combinations of variables
  - A tuple

CSPs: Assignments and Solutions

- Consistent assignment: An assignment to variables that does not violate constraints
- Complete assignment: Every variable is assignment a value
- Partial assignment: One that leaves some variables unassigned
- Partial solution: Partial assignment that is consistent
- Solution: A consistent and complete assignment

Backtracking Search



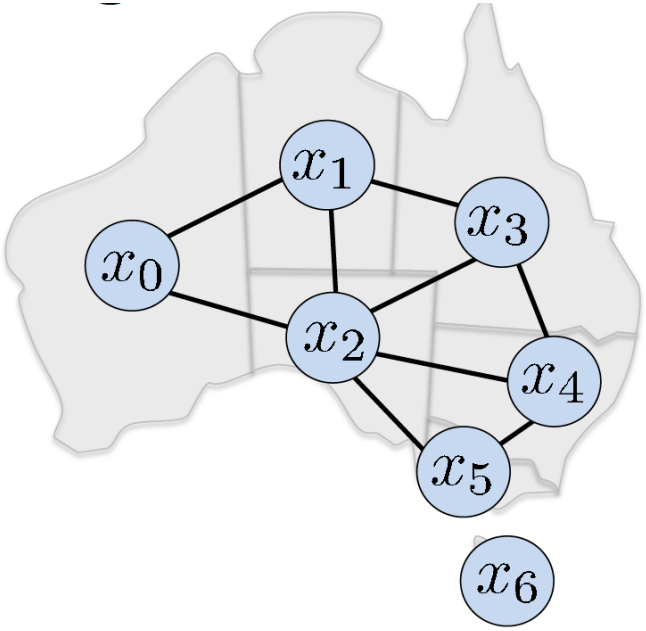
Backtracking Search

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

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

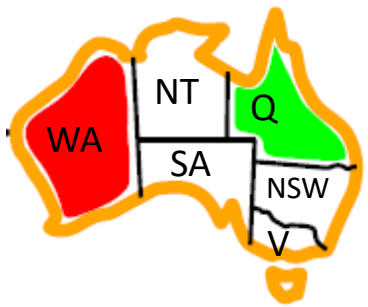
Inference in CSPs

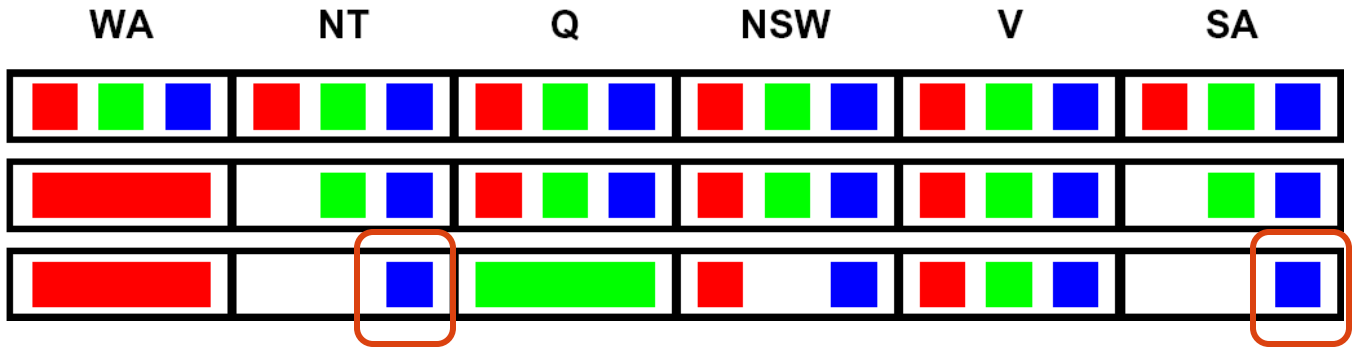
- Make use of a constraint graph
  - Nodes: variables
  - Edges: Connects variables that participate in a constraint
- Gives us an intuitive representation
- Makes it easy to prune large parts of the state space



Forward Checking Limitations

- NT and SA cannot both be blue
- Forward checking does not recognize this
- Constraint propagation



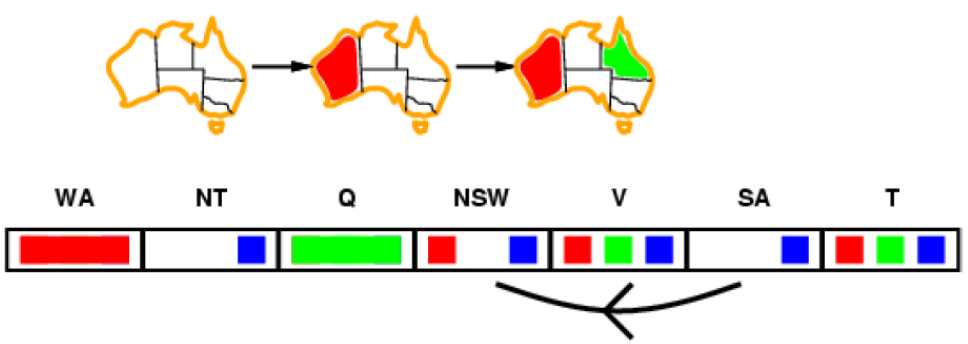


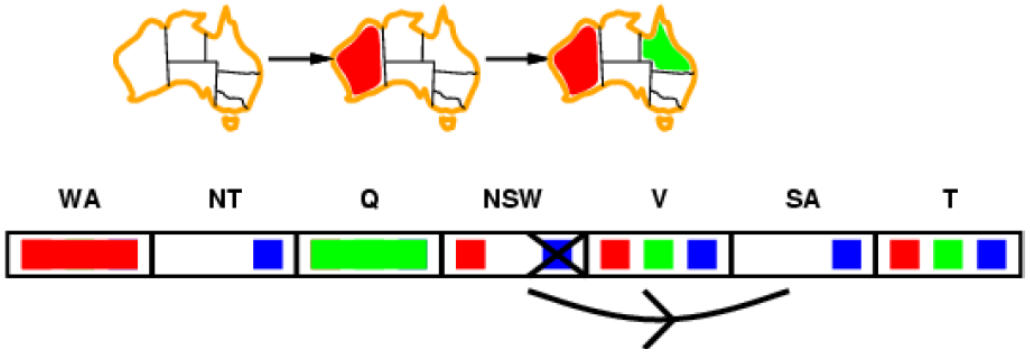
Outline

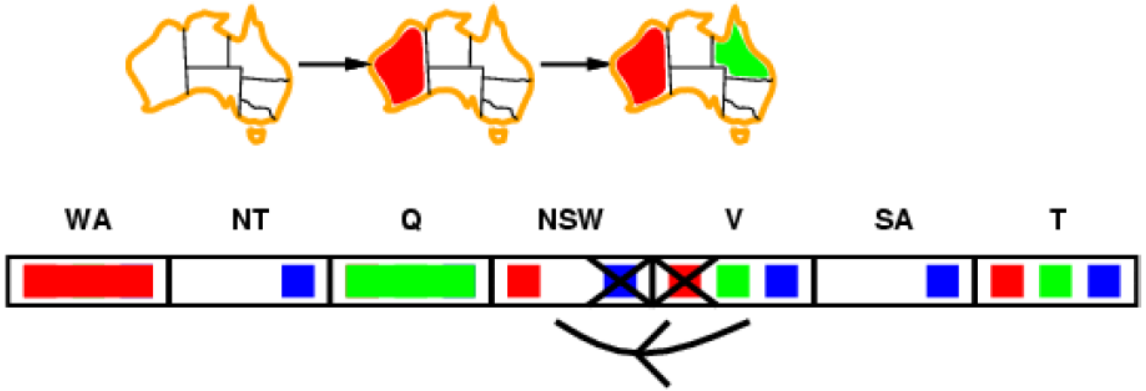
- Review
- Arc consistency
- Selecting unassigned nodes
- Ordering assignment of values
- Local search
- Special cases

Arc-Consistency

- An arc X -> Y is consistent iff for every x in the tail there is some y in the head which could be assigned without violating a constraint
  - If not, delete that value from the tail
- Forward checking: Arc consistency only for assigned variable







AC-3: Achieve Arc-Consistency

- All pairs X->Y and Y->X go on the queue
- Only modify the domain of the tail!
- If domain of tail is modified, then put it back in the queue as the head with its neighbors as the tail
- AC-3
  - $O(dn^2)$
- Revise
  - $O(d^2)$

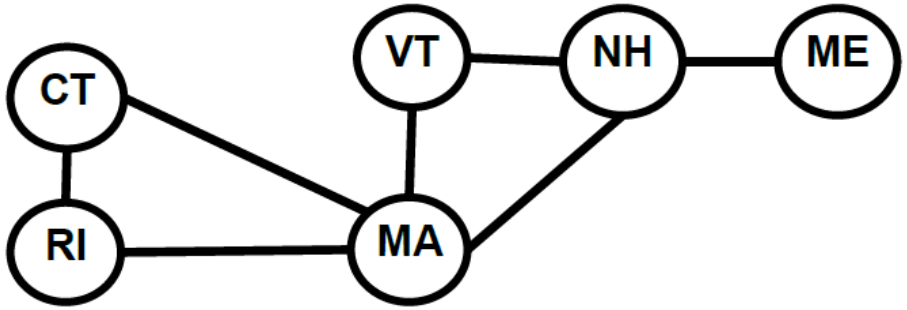
function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  queue ← a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    (X_i, X_j) ← POP(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add (X_k, X_i) to *queue*
  return *true*

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i
  revised ← *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised ← *true*
  return *revised*

Quick Quiz

5. (10 points total, 2 pts each) Constraint Satisfaction Problems.



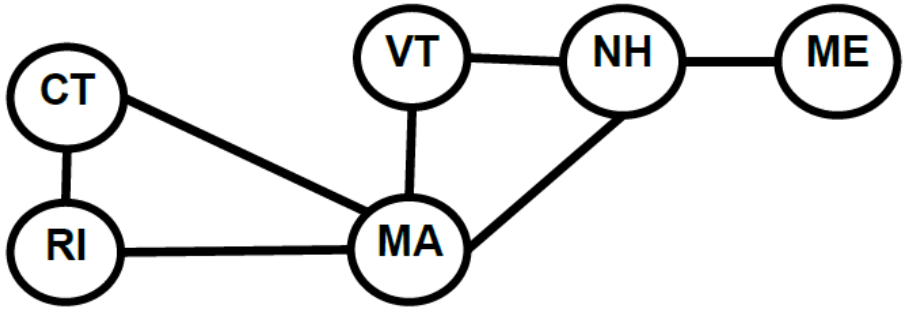
You are a map-coloring robot assigned to color this New England USA map. Adjacent regions must be colored a different color (R=Red, B=Blue, G=Green). The constraint graph is shown.

1) MA is assigned to R, what are the domains of all the other variables after forward checking has been performed?

2) CT has been assigned to R and RI has been assigned to G, what are the domains of all other variables after arc-consistency has been performed?

Quick Quiz

5. (10 points total, 2 pts each) Constraint Satisfaction Problems.



You are a map-coloring robot assigned to color this New England USA map. Adjacent regions must be colored a different color (R=Red, B=Blue, G=Green). The constraint graph is shown.

5a. (2pts total, -1 each wrong answer, but not negative) FORWARD CHECKING.
Cross out all values that would be eliminated by Forward Checking, after variable MA has just been assigned value R as shown:

CT	RI	MA	VT	NH	ME
~~R~~ G B	~~R~~ G B	R	~~R~~ G B	~~R~~ G B	R G B

5b. (2pts total, -1 each wrong answer, but not negative) ARC CONSISTENCY.
CT and RI have been assigned values, but no constraint propagation has been done. Cross out all values that would be eliminated by Arc Consistency (AC-3 in your book).

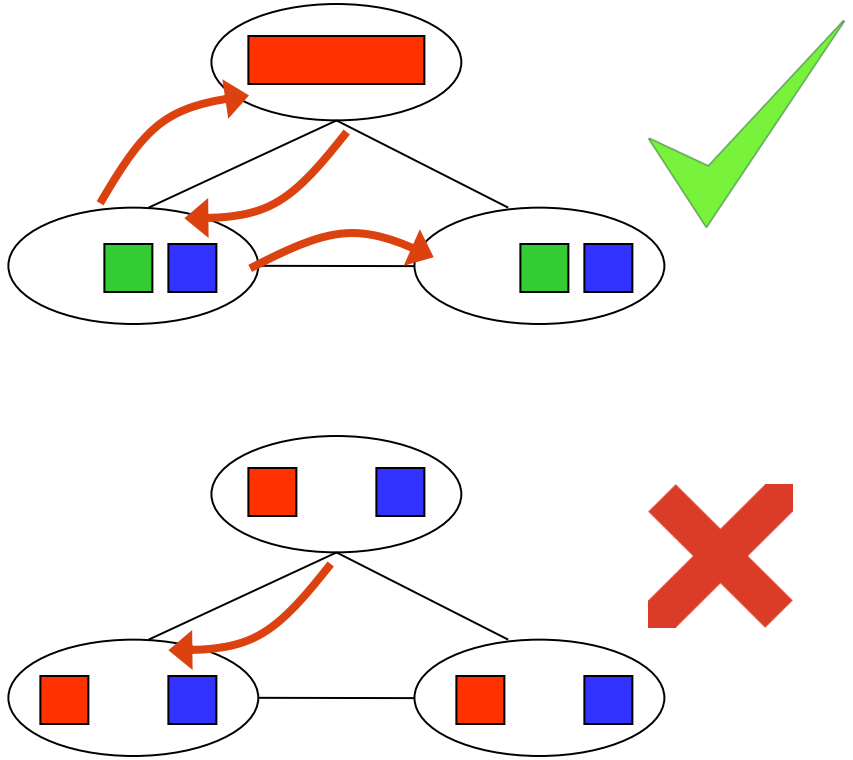
CT	RI	MA	VT	NH	ME
R	G	~~R~~ ~~G~~ B	R G ~~B~~	R G ~~B~~	R G B

Demo

- UC Berkeley CSP Demo: https://inst.eecs.berkeley.edu/~cs188/fa19/assets/demos/csp/csp_demos.html

Limitations of Arc Consistency

- K-consistency: ensure consistency of nodes in groups of K



Outline

- Review
- Arc consistency
- Selecting unassigned nodes
- Ordering assignment of values
- Local search
- Special cases

How to Select Unassigned Variables

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

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

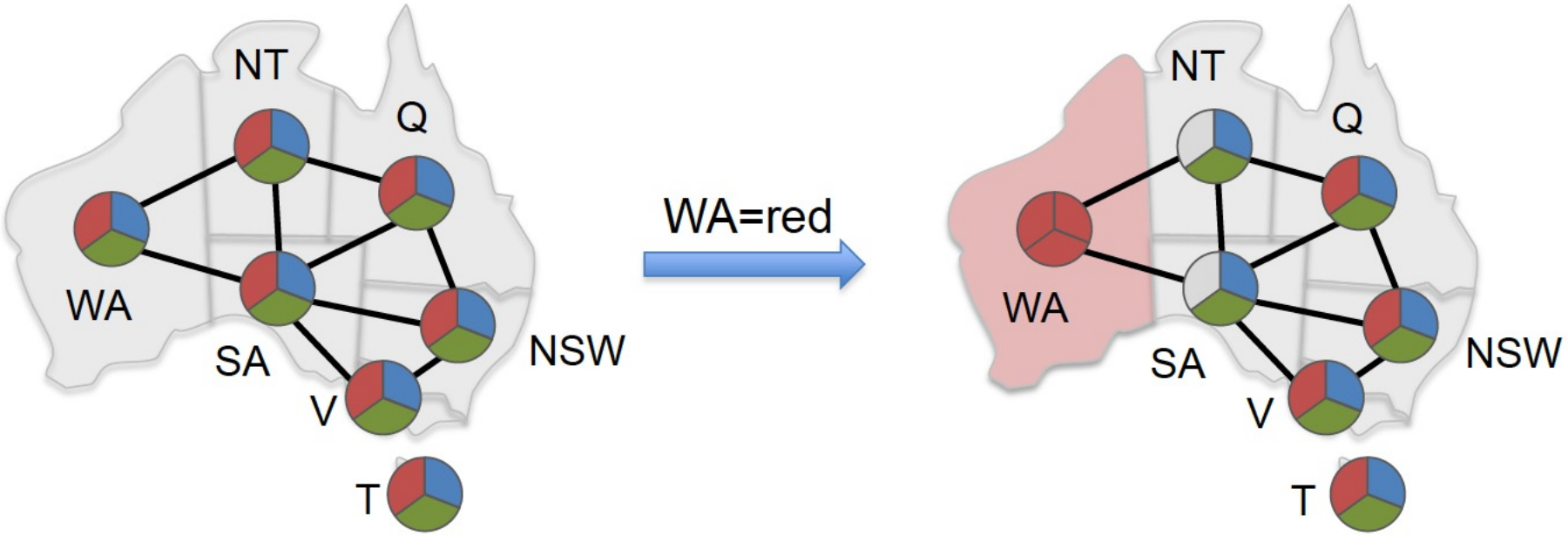
- Minimum remaining values
- Degree

Minimum Remaining Values (MRV) Heuristic

- A heuristic for selecting the next variable
  - a.k.a. most constrained variable heuristic
  - a.k.a. fail-first heuristic
  - Here “heuristic” is different than what was used in pathfinding problems
- Choose the variable with the fewest legal values
- Intuition
  - To find a solution, we have to assign values to all variables
  - If there is a variable that is more likely to cause a problem, we should encounter the problem sooner rather than later

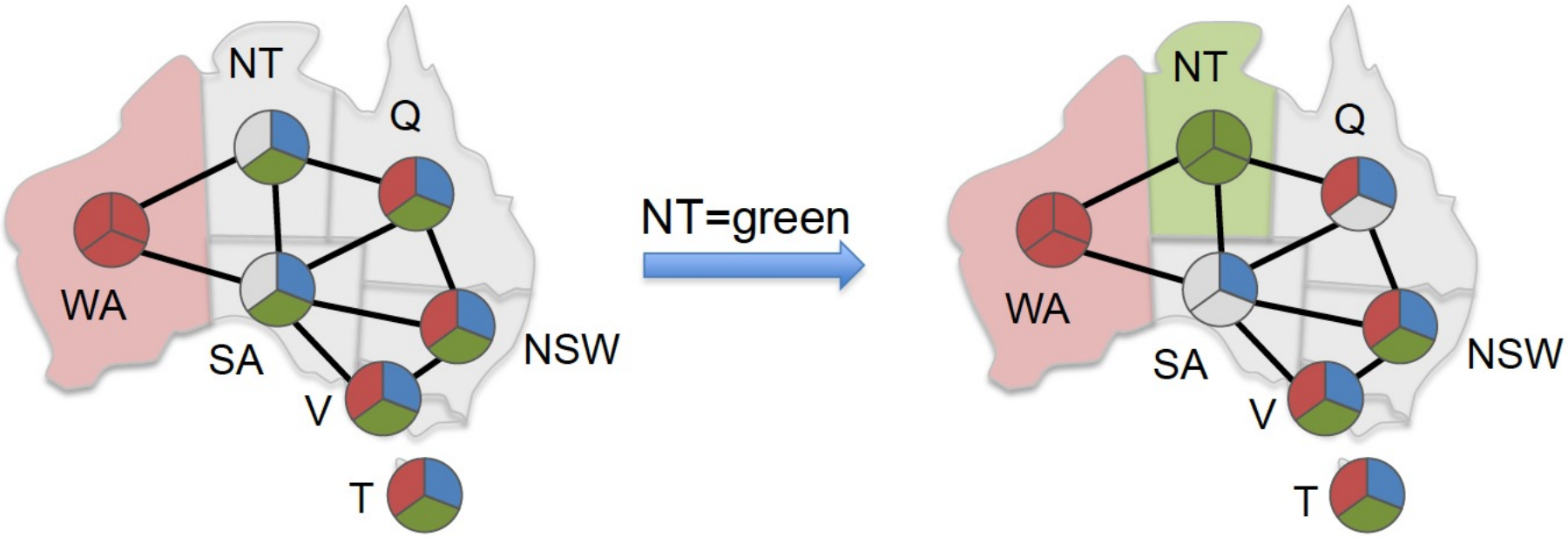
MRV: Example

- Do forward checking after each assignment
- NT and SA are tied with two
  - Pick one randomly



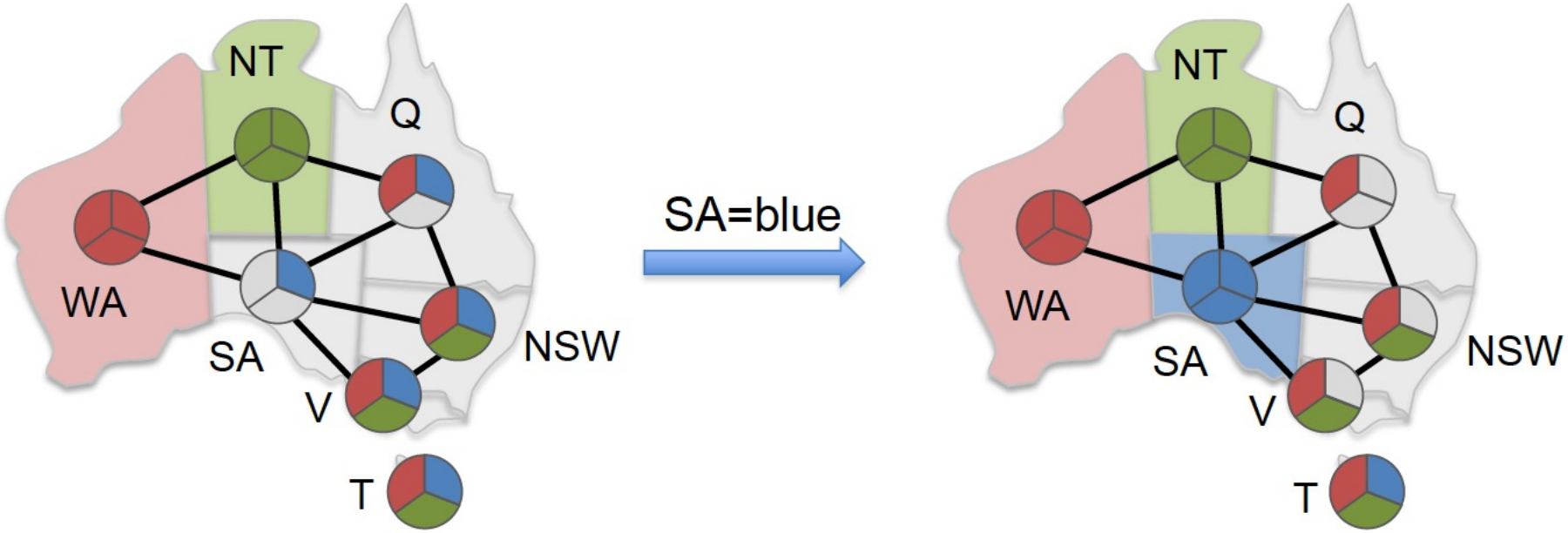
MRV: Example

- SA has the smallest



MRV: Example

- Q has the smallest
- Forward checking will lead us to a solution

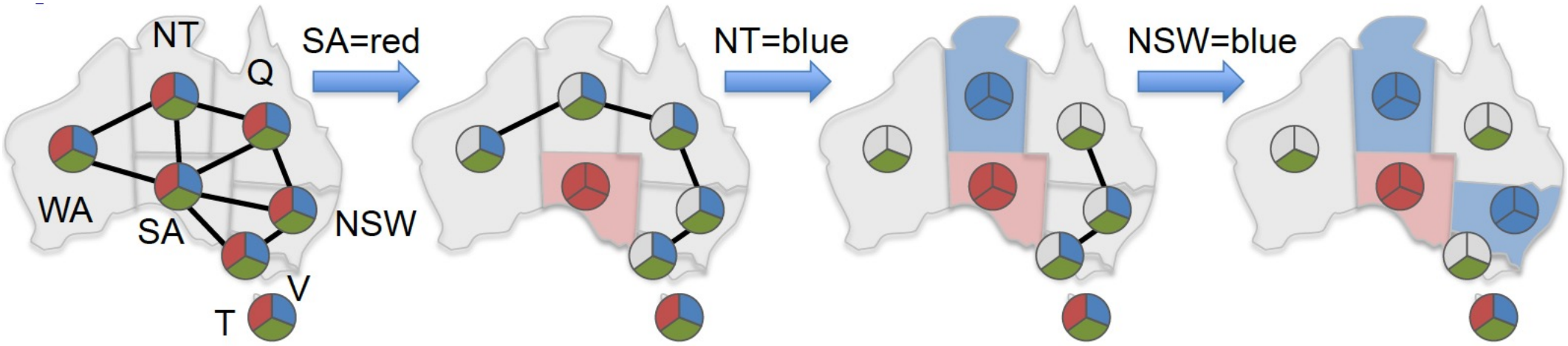


Degree Heuristic

- Another heuristic for selecting the next variable
  - a.k.a. most constraining variable heuristic
- Select the variable with the highest degree
  - Count the number of **unassigned** neighbors
- Can be a good tie-breaker for MRV
  - MRV: Which variable is the most constrained?
  - Degree: Which variable will impose the most constraints on other **unassigned** variables?

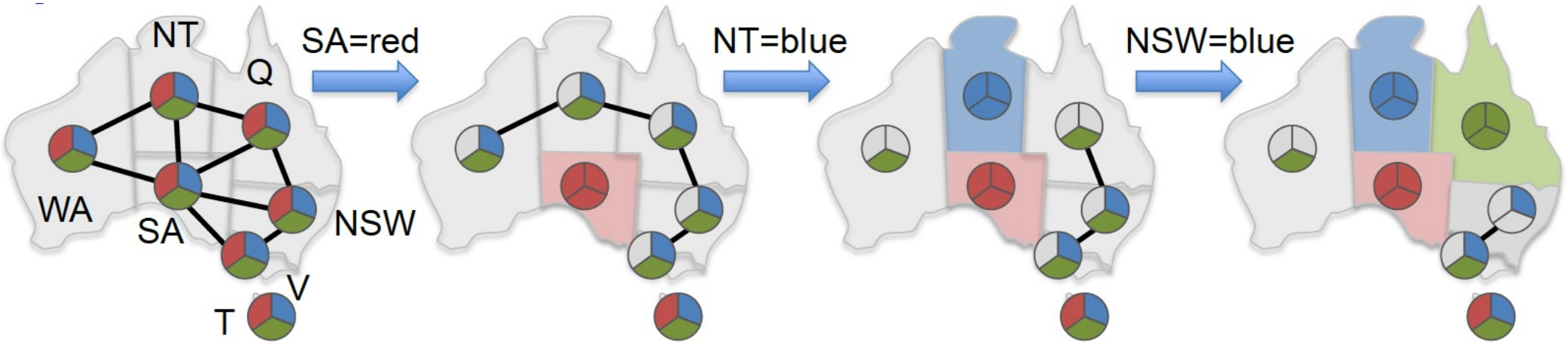
Degree Heuristic: Example

- Degree heuristic by itself
- Edges from assigned variables are removed to show how the degree heuristic counts the degree of a node



Degree Heuristic: Example

- Degree heuristic as a tiebreaker for MRV
- Edges from assigned variables are removed to show how the degree heuristic counts the degree of a node



Outline

- Review
- Arc consistency
- Selecting unassigned nodes
- Ordering assignment of values
- Local search
- Special cases

Ordering Domain Variables

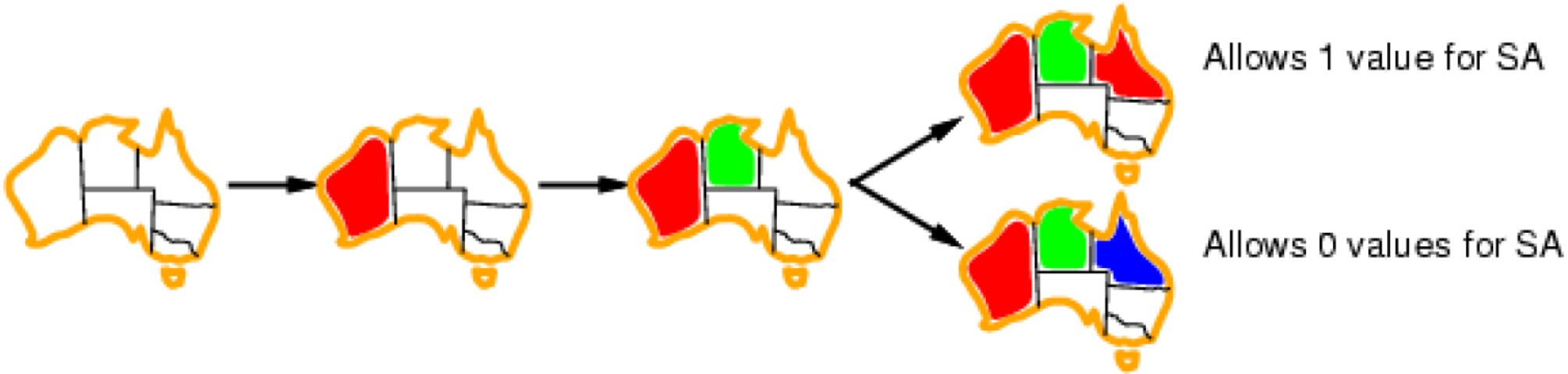
```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
    return BACKTRACK(csp, { })

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

- Least constraining value

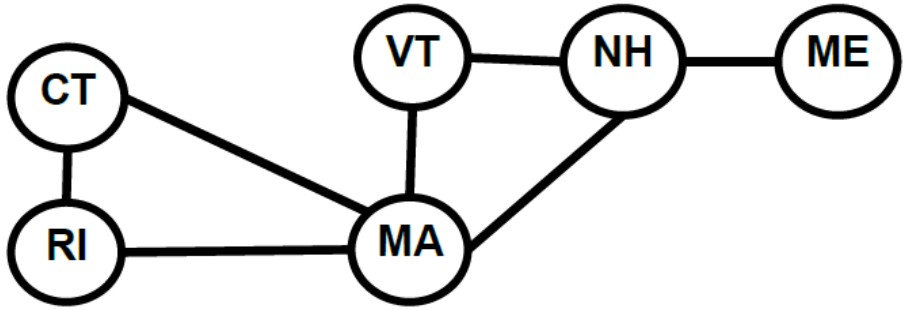
Least Constraining Value

- Heuristic for selecting which variable to try next
- Choose the value that rules out the **fewest** number of values from neighboring nodes
- Intuition
  - We can choose any value
  - We should choose the one that we think is the most promising for finding a solution



Quick Quiz

5. (10 points total, 2 pts each) Constraint Satisfaction Problems.



You are a map-coloring robot assigned to color this New England USA map. Adjacent regions must be colored a different color (R=Red, B=Blue, G=Green). The constraint graph is shown.

5c. (2pts total, -1 each wrong answer, but not negative) MINIMUM-REMAINING-VALUES HEURISTIC. Consider the assignment below. RI is assigned and constraint propagation has been done. List all unassigned variables that might be selected by the Minimum-Remaining-Values (MRV) Heuristic:_____.

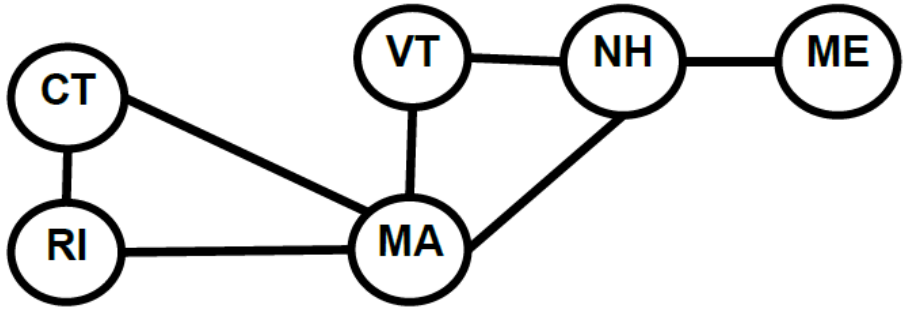
CT	RI	MA	VT	NH	ME
R B	G	R B	R G B	R G B	R G B

5d. (2pts total, -1 each wrong answer, but not negative) DEGREE HEURISTIC. Consider the assignment below. (It is the same assignment as in problem 5c above.) RI is assigned and constraint propagation has been done. List all unassigned variables that might be selected by the Degree Heuristic:_____.

CT	RI	MA	VT	NH	ME
R B	G	R B	R B	R G B	R G B

Quick Quiz

5. (10 points total, 2 pts each) Constraint Satisfaction Problems.



You are a map-coloring robot assigned to color this New England USA map. Adjacent regions must be colored a different color (R=Red, B=Blue, G=Green). The constraint graph is shown.

5c. (2pts total, -1 each wrong answer, but not negative) MINIMUM-REMAINING-VALUES HEURISTIC. Consider the assignment below. RI is assigned and constraint propagation has been done. List all unassigned variables that might be selected by the Minimum-Remaining-Values (MRV) Heuristic: **CT, MA** .

CT	RI	MA	VT	NH	ME
R B	G	R B	R G B	R G B	R G B

5d. (2pts total, -1 each wrong answer, but not negative) DEGREE HEURISTIC. Consider the assignment below. (It is the same assignment as in problem 5c above.) RI is assigned and constraint propagation has been done. List all unassigned variables that might be selected by the Degree Heuristic: **MA, NH** .

CT	RI	MA	VT	NH	ME
R B	G	R B	R B	R G B	R G B

Backtracking Summary

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

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

- Minimum remaining values
- Degree
- Least constraining value
- Forward checking
- K-Consistency

Quick Quiz

For each of the following terms on the left, write in the letter corresponding to the best answer or the correct definition on the right.

	Minimum Remaining Values Heuristic	A	Specifies the allowable combinations of variable values
	Solution to a CSP	B	The values assigned to variables do not violate any constraints
	Least Constraining Value Heuristic	C	Set of allowed values for some variable
	Domain	D	Every variable is associated with a value
	Constraint	E	Nodes correspond to variables, links connect variables that participate in a constraint
	Consistent Assignment	F	Chooses the next variable to expand to have the fewest legal values in its domain
	Complete Assignment	G	A complete and consistent assignment
	Constraint Graph	H	Prefers to search next the value that rules out the fewest choices for the neighboring variables in the constraint graph

Quick Quiz

For each of the following terms on the left, write in the letter corresponding to the best answer or the correct definition on the right.

F	Minimum Remaining Values Heuristic	A	Specifies the allowable combinations of variable values
G	Solution to a CSP	B	The values assigned to variables do not violate any constraints
H	Least Constraining Value Heuristic	C	Set of allowed values for some variable
C	Domain	D	Every variable is associated with a value
A	Constraint	E	Nodes correspond to variables, links connect variables that participate in a constraint
B	Consistent Assignment	F	Chooses the next variable to expand to have the fewest legal values in its domain
D	Complete Assignment	G	A complete and consistent assignment
E	Constraint Graph	H	Prefers to search next the value that rules out the fewest choices for the neighboring variables in the constraint graph

Outline

- Review
- Arc consistency
- Selecting unassigned nodes
- Ordering assignment of values
- Local search
- Special cases

Local Search

- We previously talked about the benefits of CSPs over search where states were atomic
- However, local search can still be very useful
- Local search can be very fast
  - Especially in scenarios where the state space has a high density of solutions
- In an online setting (i.e. airline scheduling) if the schedule suddenly changes due to weather, local search can help find a new schedule that is good
  - Can be much better than starting from scratch

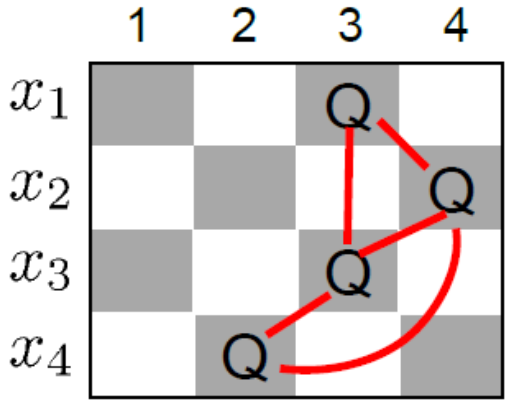
Local Search: Min Conflicts Heuristic

- Start with a complete assignment
- Repeat for N Iterations
  - Choose a variable at random
  - Assign it a value with the smallest number of conflicts

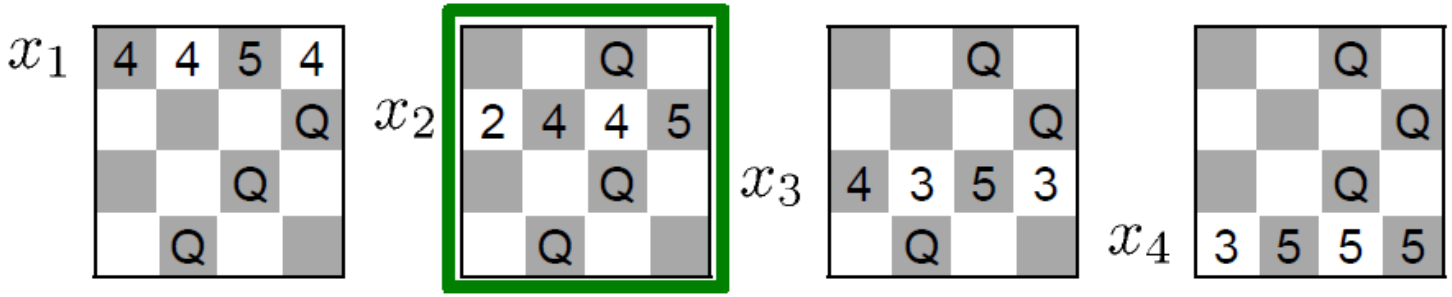
function MIN-CONFLICTS(csp, max_steps) **returns** a solution or *failure*
  inputs: csp, a constraint satisfaction problem
    max_steps, the number of steps allowed before giving up

  $current \leftarrow$ an initial complete assignment for csp
  for $i = 1$ to max_steps **do**
    if $current$ is a solution for csp **then return** $current$
    $var \leftarrow$ a randomly chosen conflicted variable from csp.VARIABLES
    $value \leftarrow$ the value v for var that minimizes CONFLICTS($csp, var, v, current$)
    set $var = value$ in $current$
  return *failure*

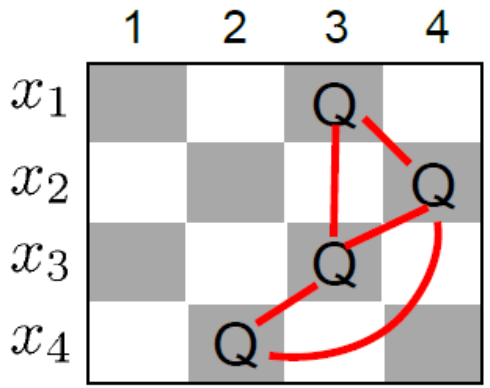
Min Conflicts: Example



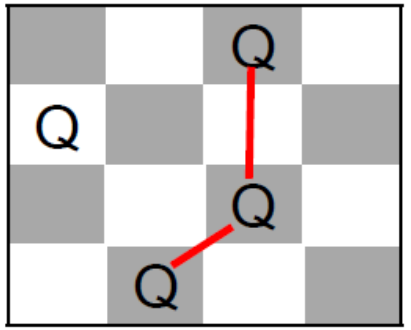
(5 conflicts)



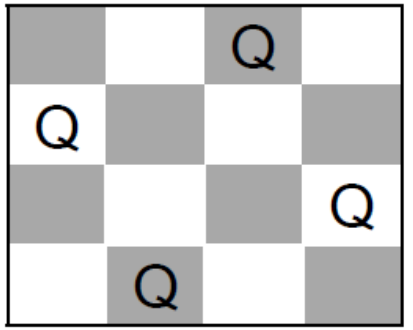
Min Conflicts: Example



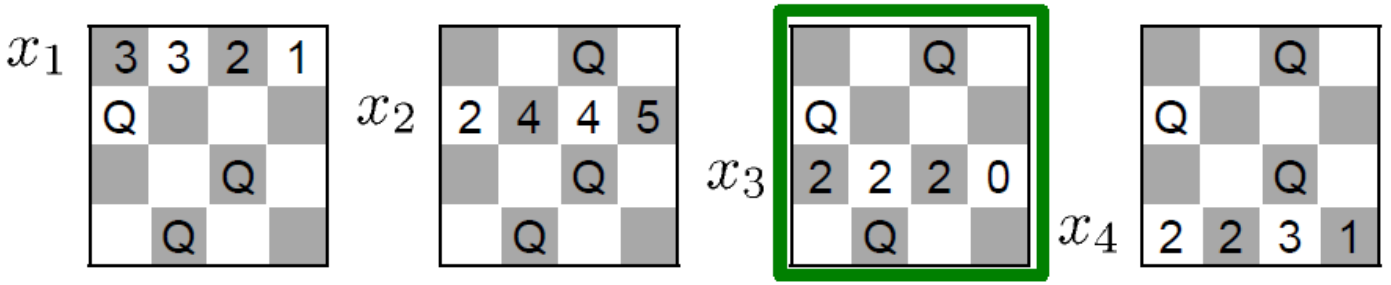
(5 conflicts)



(2 conflicts)



(0 conflicts)



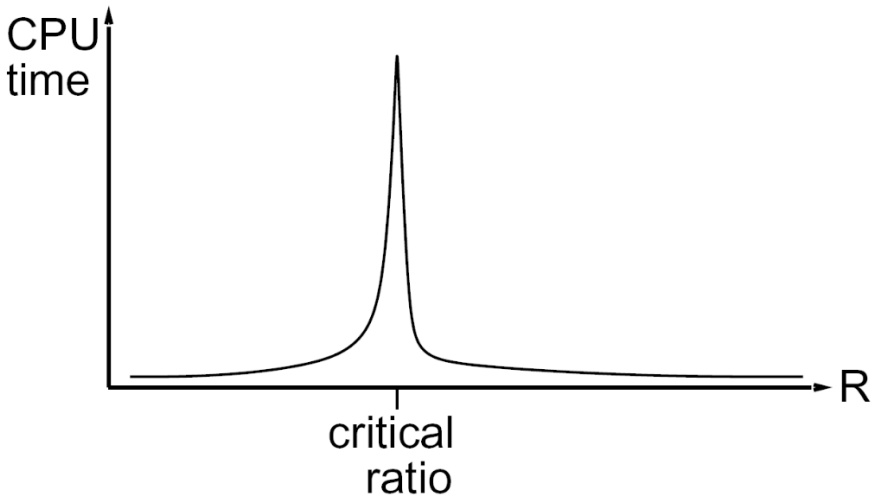
Outline

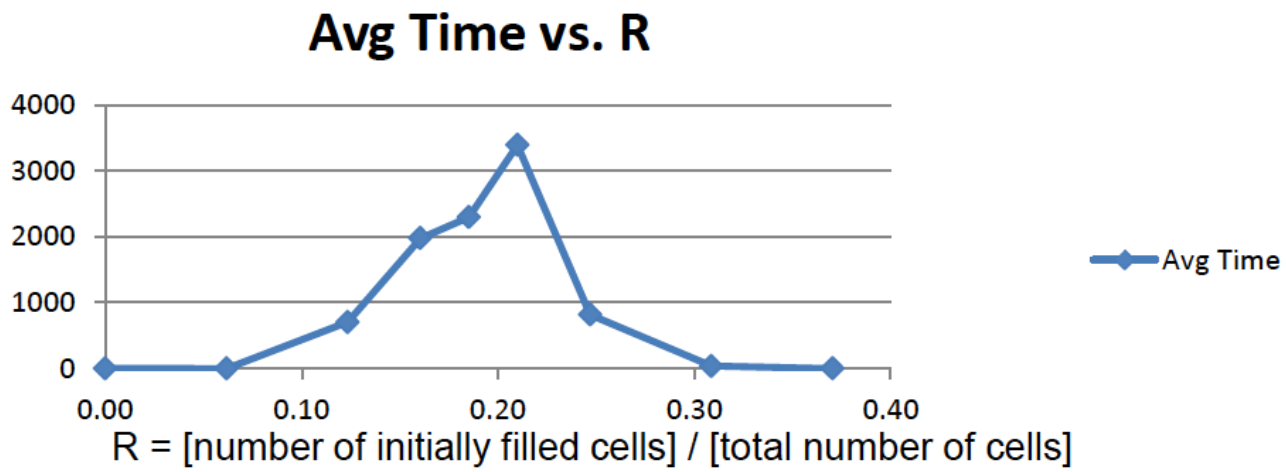
- Review
- Arc consistency
- Selecting unassigned nodes
- Ordering assignment of values
- Local search
- Special cases

Hardness of CSPs

- Boolean satisfiability can be posed as a CSP and is NP-Complete
- How hard for everyday examples?
  - Sudoku example

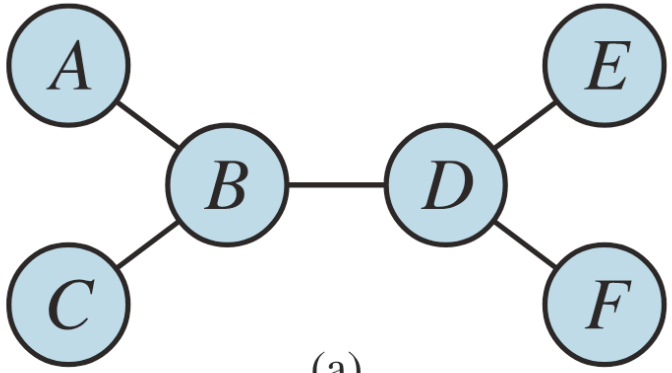
$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$



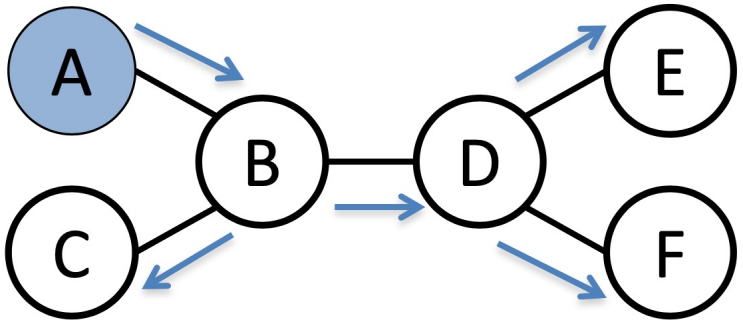


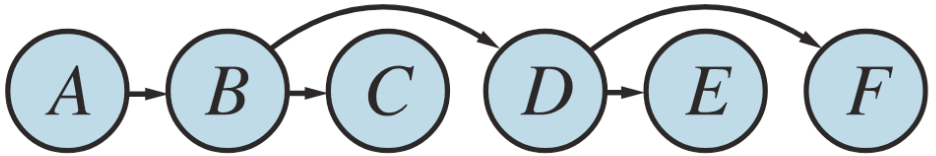
Structure: Tree-Structured CSPs

- The time complexity for solving CSPs, in the worst case, is $O(d^n)$
  - Where d is the size of the domain and n is the number of nodes
- If a constraint graph has no cycles, then the CSP can be solved in $O(nd^2)$ time.
  - This can be achieved by converting the graph to a tree structure
  - Tree: each node has, at most, one parent
- Start at the root and do arc consistency
- Then start at the root and make consistent assignments



(a)

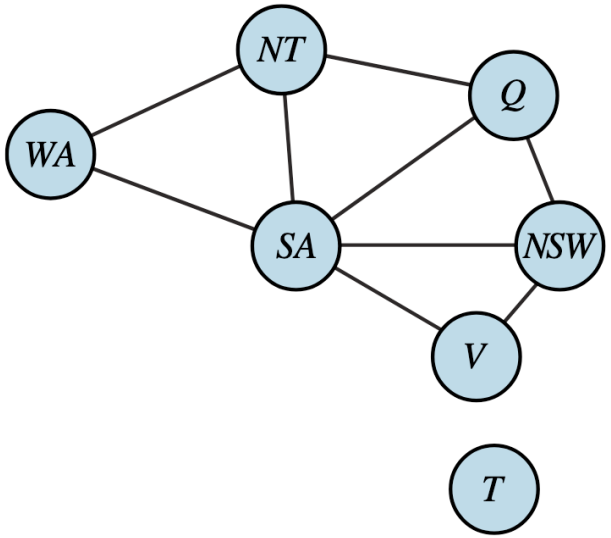


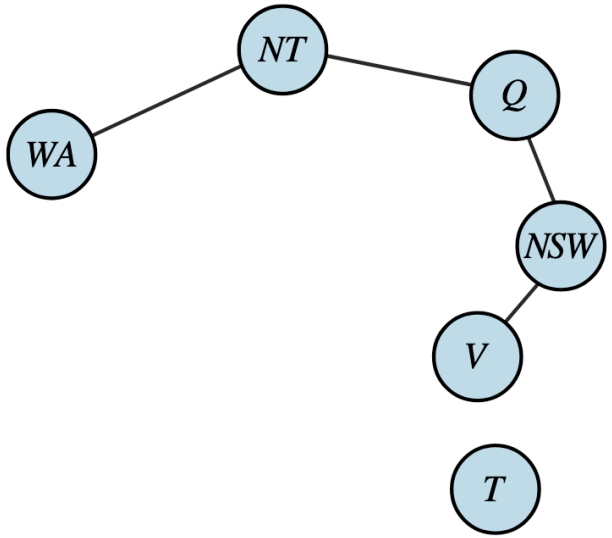


(b)

Structure: Cutset

- What if there are cycles?
- Choose a subset of the CSP variables such that the constraint graph becomes a tree after removal
  - This subset is called a cycle cutset
- If the cycle cutset has size c, then the total runtime is $O(d^c(n-c)d^2)$





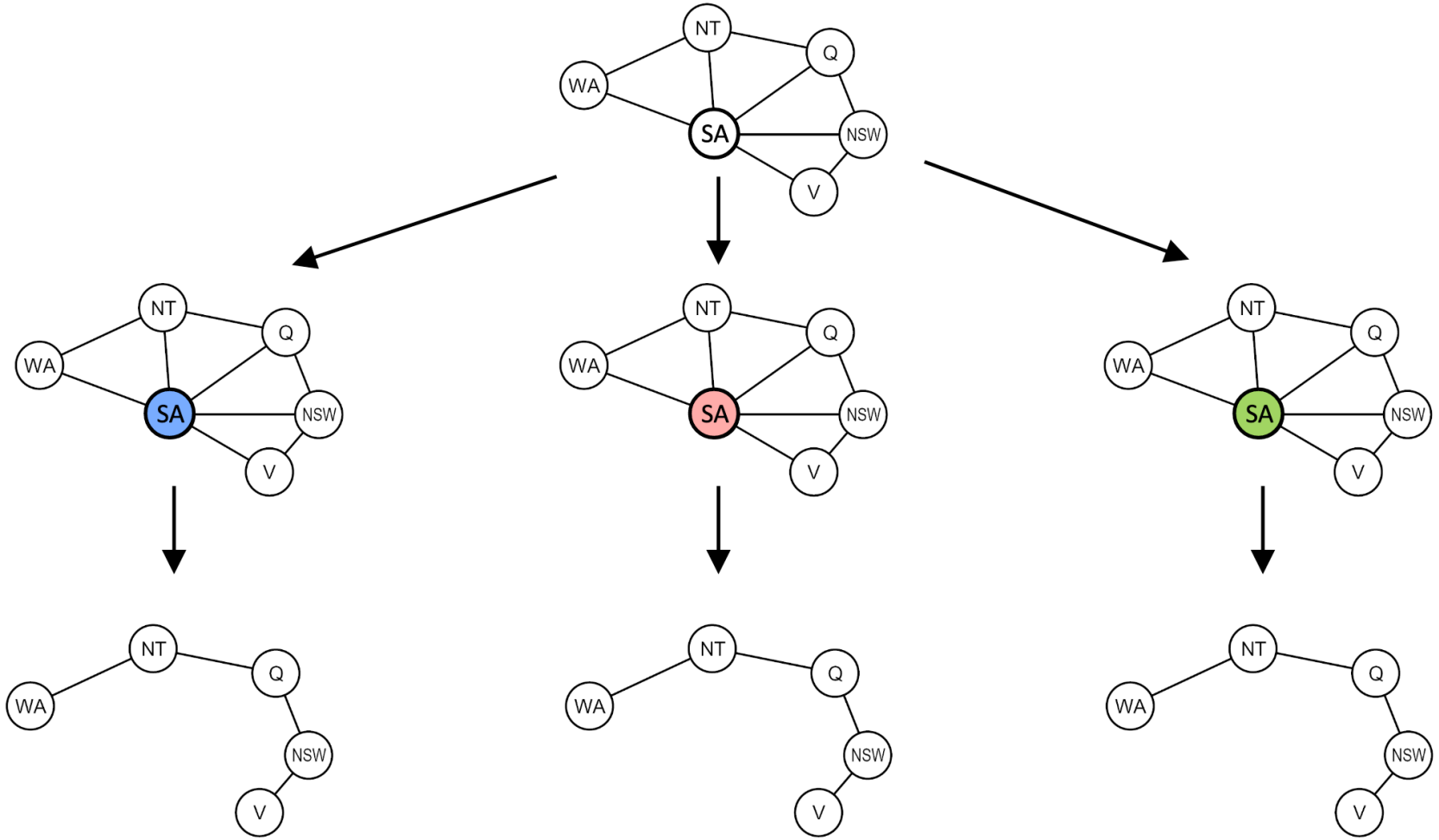
Cutset

Choose a cutset

Instantiate the cutset (all possible ways)

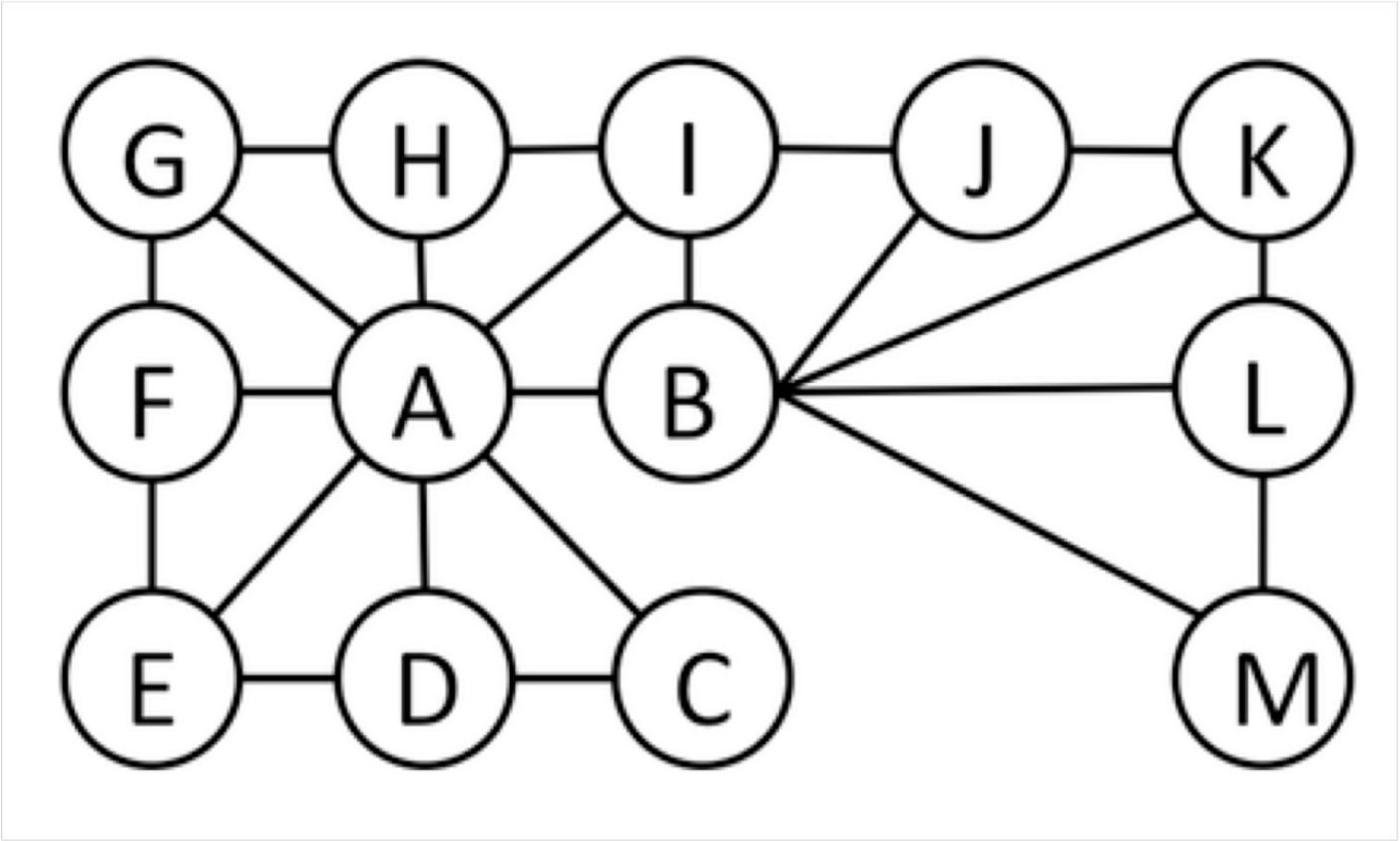
Compute residual CSP for each assignment

Solve the residual CSPs (tree structured)



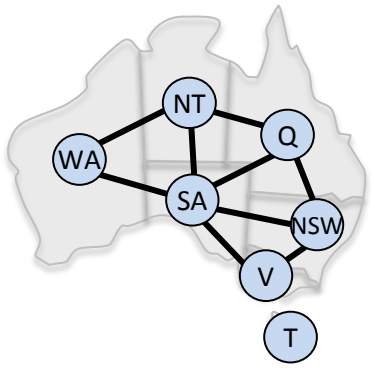
Quick Quiz

- Find the smallest cutset that makes this graph a tree



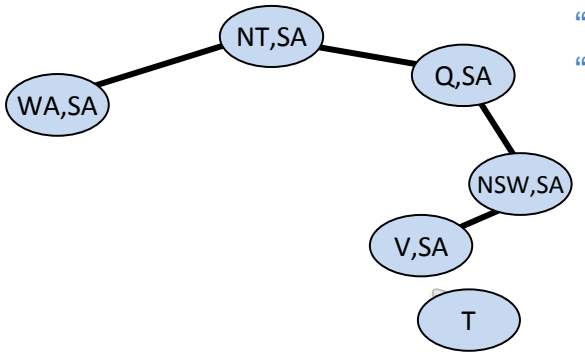
Combining Variables

- Can also convert a CSP into a tree by combining variables



Change "variables"

= color of pair of areas



Now:
"unary" WA-SA constraint
"binary" (WA,SA) – (NT,SA)
require all 3 consistent
…

Summary

- CSPs: Variables, domains, constraints
- Backtracking search: depth-first search that backtracks when there is a conflict
  - Select un-assigned variable
    - Minimum remaining values heuristic
    - Degree heuristic
  - Order domain values
    - Least constraining value
  - Inference
    - Forward checking
    - Arc consistency
    - K-consistency
- Local search
  - In practice, min-conflicts search can help quickly modify solutions in case constraints change
- Computational complexity can be reduced given a tree structure

Next Time

- Propositional Logic