Constraint Satisfaction Problems (CSP)

- Reading: Russell and Norvig, ch. 6
  - The material here is an edited version of Russell's slides
- Intro: map coloring example
- Domains
- Varieties of constraints
- Algorithms
  - Backtracking search

CSP intro

- Can use CSP when:
  - State defined as set of variables with specific values
  - Check for goal state by checking if given combination of values is acceptable
  - By limiting problem, supports different algorithms
- For N-Queens,
  - What would the state be?
  - How would you check if a state is a goal state?
- 8-puzzle doesn’t really work because the steps are part of the solution
  - Cannot just swap positions of tiles

Example: Map coloring

- Goal: color areas (ex: provinces, countries) of a map so that adjacent areas always have different colors
- State: color for each area
- Can convert to a graph:
  - Each area is a vertex
  - Connect vertex to vertices representing adjacent areas
  - Example of *binary* CSP: constraint relates at most 2 variables
- Graph theory result: need at most 4 colors
  - see, for example: https://www.mathsisfun.com/activity/coloring.html

Example: Map coloring

- https://en.wikipedia.org/wiki/Graph_coloring#Applications :
  - Scheduling:
    - vertices represent jobs
    - edges represent jobs that conflict – ex: require same resource or same person
    - “coloring” = assigning time slot to job
  - Register allocation:
    - When compiler generates code, more efficient to store program variables in registers
    - If program variables needed at same time, cannot use same register
    - “coloring” = assigning register to program variable
    - C/C++ `register` keyword is a <u>hint</u> to use a register, if possible

Domain of problem variables

- Discrete
  - In general, harder than continuous domain!
  - Many problems for even finite domains are NP-Complete
    - Ex: Boolean satisfiability
  - In infinite domains (integers, strings)
    - Linear constraints are still solvable
    - Nonlinear constraints lead to undecidable problems
- Continuous
  - Linear constraints: solvable in polynomial time by linear programming algorithms
  - Fortunately, real world problems tend to be continuous

Varieties of constraints

- Unary: constraints on only one variable
  - Ex: vertex for California must be colored gold
- Binary: constraints apply to at most 2 variables
  - Ex: map coloring
- Higher order:
  - Ex: Cryptarithmetic column constraints (next example)
- *Preferences:* soft constraints
  - Ex: try to schedule different sections of Calc 1 on MWF and TuTh
  - Often modeled as costs for assignments that violate constraints

Cryptarithmetic example

- Constraints defined by equation with numbers encoded by words
- Each letter represents a distinct digit (0-9)
  - Goal: assign digits to letters to get valid equation
- Ex:

```
  TWO
+ TWO
-----
 FOUR
```

Algorithms

- Start with straightforward, dumb approach
- Depth-first search
  - Assign value to unassigned variable that does not conflict with existing assignments
  - Backtrack if cannot find valid assignment for variable
  - Fail if failure with assigning values in all permutations of variables
- Branching factor = $(n-L)d$ for n variables at level L, domain size = d
  - $n!d^n$ leaves: can assign variables in any order, exponential in size of domain

Backtracking search

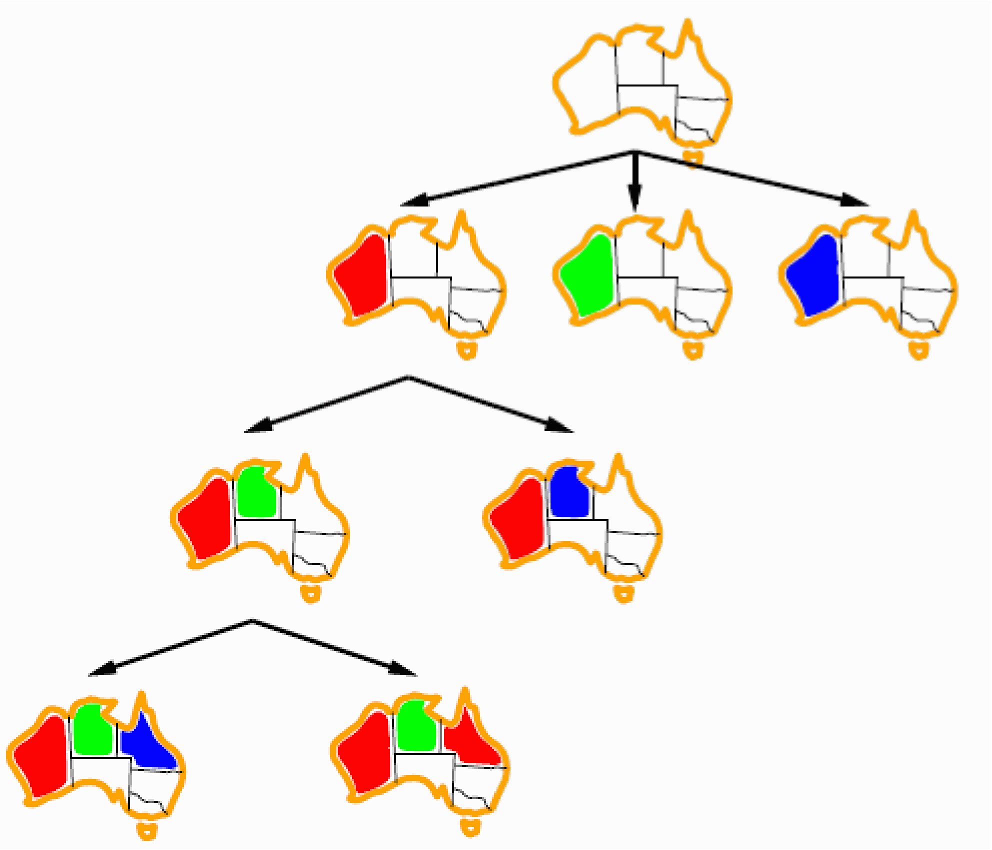
- Variable assignments are *commutative* if swapping values still satisfies constraints
  - Ex: for map coloring, solution groups areas into sets with same color
    - Can use any permutation of colors
- Branching factor reduced to *d*
  - At each level, only consider one variable

Backtracking search 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
```

Backtracking example

- Edited version of Russell's figure:
  - (J. Neal Richter)
  - Bottom two leaf nodes swapped
  - What happens at the bottom left?



Backtracking efficiency

- General methods can greatly improve efficiency
- To consider:
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?
  - Can we take advantage of problem structure?

Minimum remaining values

- In backtracking example, assign to South Australia (bottom center) before Queensland (top right) because 2 of its neighbors have already been given a color
- Sudoku row
  - Obviously, if 8 columns assigned, last column only has 1 legal value left
  - If 7 columns assigned, assigning to last 2 columns may require backtracking, but just once

Degree heuristic

- Tiebreaker for minimum remaining value variables
  - Choose variable with the most constraints on remaining variables
  - Ex: for map coloring, the area with the most neighbors that remain uncolored

Least constraining value

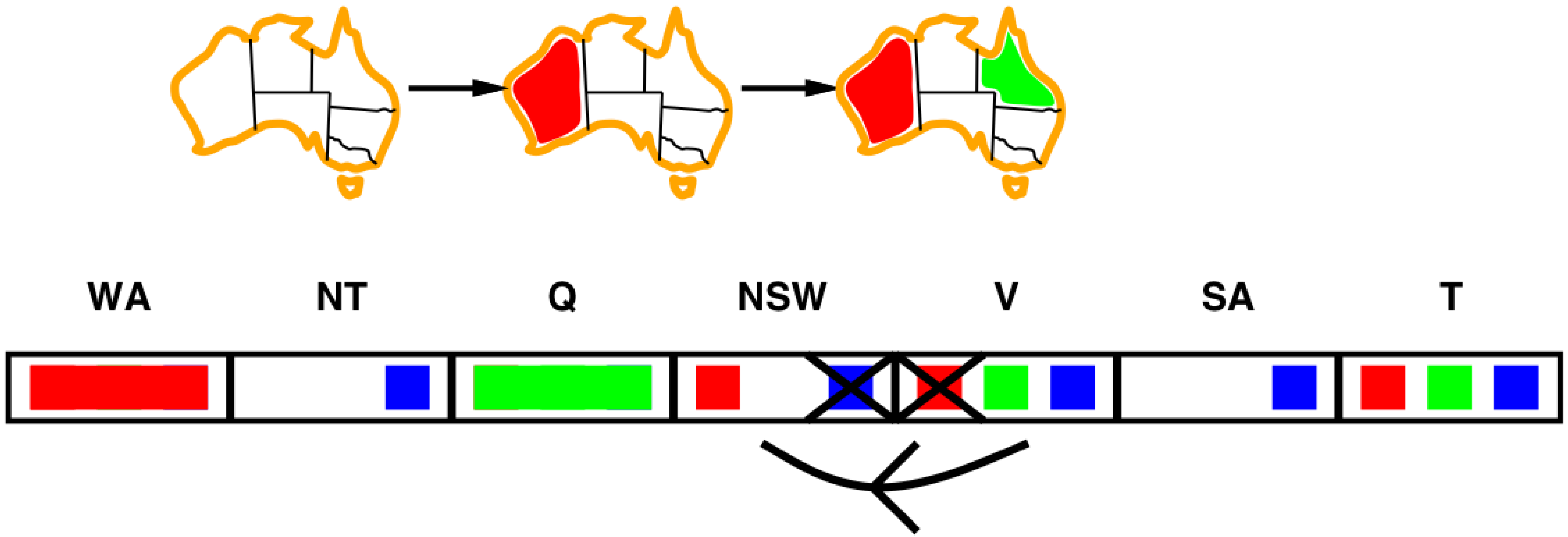
- Choose value that rules out the fewest values in remaining variables
- In Cryptarithmetic example (TWO + TWO = FOUR):
  - We know F = 1
  - O, U, R must be even
  - Therefore, try to pick odd values for T and W
- Using minimum remaining value, the degree heuristic and least constraining value, can solve N-queens problem for problem with N about 40 times greater

Early detection of failure

- Forward checking
  - Keep track of remaining legal values for each variable
  - Can backtrack as soon as any variable has no legal values
  - Ex: In Sudoku, if guesses lead to row only missing a 1, but with a 1 in same column as empty square

Early detection of failure

- Forward checking does not detect all inevitable failures
- Ex: Northern Territory and South Australia both can only be blue



Arc consistency

- For arc (edge) from X to Y,
  - Arc is consistent iff for every value of X, there is at least one legal value of Y
- If variable assignment removes any value from X, X's neighbors need to be rechecked
  - Ex: for coloring Australia, making Queensland green removes that color option from New South Wales

Arc consistency algorithm

```
function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp
  while queue is not empty do
    (Xi, Xj)←Remove-First(queue)
    if Remove-Inconsistent-Values(Xi, Xj) then
      for each Xk in Neighbors[Xi] do
        add (Xk, Xi) to queue

function Remove-Inconsistent-Values(Xi, Xj) returns true iff succeeds
  removed←false
  for each x in Domain[Xi] do
    if no value y in Domain[Xj] allows (x,y) to satisfy the constraint Xi ↔ Xj then
      delete x from Domain[Xi]; removed←true
return removed
```

Problem structure

- For coloring Australia,
  - Tasmania is an island, so can consider it separately from other territories
  - In general, disconnected parts of constraint graph allow them to be considered separately
- In general, if possible to divide problem into subproblems, each with c of the variables, can reach linear cost in n ($n/c * d^c$)
  - For n=80, d=2, c=20,
    - Execution time goes from 4 billion years to .4 seconds

Tree-structured problems

- If no cycles in constraints, problem is solvable in $O(n\ d^2)$ vs $O(d^n)$ for general CSP
- Algorithm:
  1. Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering
  2. For j from n down to 2, apply `RemoveInconsistent`(Parent(Xj),Xj)
  3. For j from 1 to n, assign Xj consistently with Parent(Xj)

Nearly tree-structured problems

- *Conditioning*: assign a value to a variable, prune neighbor's domains
    - Goal is to get a tree
    - *Cutset conditioning*: for a subset, find all assignments that result in trees
        - Cutset cannot be too big
        - How might you choose the cutset?