CSCI 5582
Artificial Intelligence

Fall 2006

Jim Martin

Today 9/14

- Constraint Sat Problems
- Admin/Break
- Constraint Sat as Iterative Improvement

Search Types

- Backtracking State-Space Search
- Optimization-Style Search
- Constraint Satisfaction Search

Constraint Satisfaction

- In CSP problems, states are represented as sets of variables, each with values chosen from some domain
- A goal test consists of satisfying constraints on sets of variable/value combinations
- A goal state is one that has no constraint violations

Examples

- Simple puzzles
- Graph coloring
- Scheduling problems
- Any constrained resource problem

N-Queens

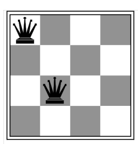
- Place N queens on a chess board such that no queen is under attack from any other queen.

4-Queen Example

- Assume a 4x4 board
- Assume one queen per column
- 4 Variables (Q1, Q2, Q3, Q4)
- 4 possible values (1,2,3,4)
- Constraints…

Constraints

- $V(Q_i) \neq V(Q_k)$
  - Can't be in the same row
- $|V(Q_i) - V(Q_k)| \neq |i - k|$
  - or the same diagonal

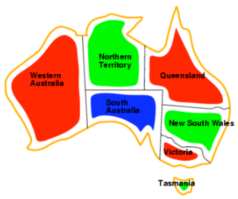


$Q_1 = 1 \quad Q_2 = 3$

Example: Map-Coloring

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

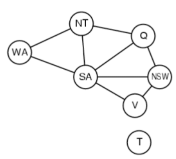
Example: Map-Coloring



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

Constraint graph

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



Varieties of constraints

- Unary constraints involve a single variable,
  - e.g., SA ≠ green
- Binary constraints involve pairs of variables,
  - e.g., SA ≠ WA
- Higher-order constraints involve 3 or more variables,
  - e.g., cryptarithmetic column constraints

Approaches to CSPs

- As a kind of backtracking search
  - Uninformed or informed
- As a kind of iterative improvement

CSCI 5582 Fall 2006

CSP as Backtracking (Dumb)

- Start state has no variables assigned
- Assign a variable at each step
- Apply goal test to completed states
- Where are solutions found?
- What kind of (dumb) search might be applicable?

CSCI 5582 Fall 2006

Less Dumb

- What it means to be a goal (or not) can be decomposed
- What the heck does that mean?
  - In CSPs a state is a goal state if all of the constraints are satisfied.
  - A state fails as a goal state if any constraint is violated
  - So…

CSCI 5582 Fall 2006

Less Dumb

- Check to see if any constraints are violated as variables are assigned values.
- This is backward checking since you're checking to see if the current assignment conflicts with any past assignment

Standard search formulation (incremental)

Let's start with the straightforward approach, then fix it

States are defined by the values assigned so far

- Initial state: the empty assignment { }
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment
  → fail if no legal assignments
- Goal test: the current assignment is complete

1. This is the same for all CSPs
2. Every solution appears at depth *n* with *n* variables
   → use depth-first search
3. Path is irrelevant, so can also use complete-state formulation

Backtracking search

- 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 $n \approx 25$

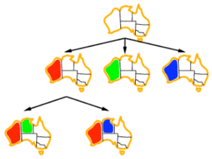
Backtracking search

```
function BACKTRACKING-SEARCH( csp) returns a solution, or failure
   return RECURSIVE-BACKTRACKING({}, csp)
function RECURSIVE-BACKTRACKING( assignment,csp) returns a solution, or
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 according to Constraints[csp] then
         add { var = value } to assignment
         result ← RECURSIVE-BACKTRACKING(assignment, csp)
         if result ≠ failue then return result
         remove { var = value } from assignment
   return failure
```

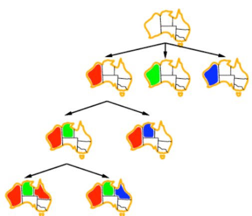

Backtracking example

Backtracking example

Backtracking example



Backtracking example

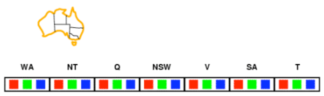


Even Better

- Add forward checking
  - When you assign a variable check to see if it still allows future assignments to the remaining variables
- Using forward checking and backward checking roughly doubles the size of N-queens problems that can be practically solved (from 15 to 30).

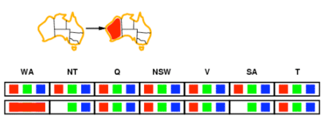
Forward checking

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



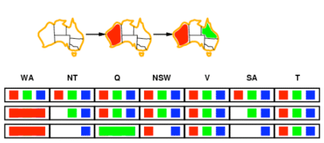
Forward checking

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



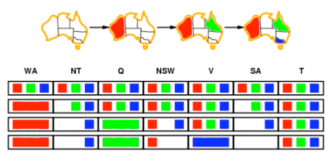
Forward checking

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



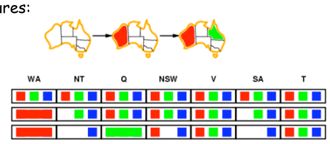
Forward checking

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



Constraint propagation

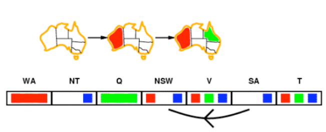
- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:



- At this point all variables have possible values. But NT and SA cannot both be blue! Backtracking should occur here, not at the next step.

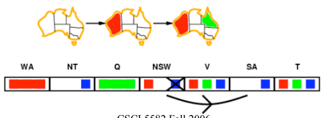
Arc consistency

- Simplest form of propagation makes each arc consistent
- *X* → *Y* is consistent iff
  For every value *x* for *X* there is some allowed value *y* for Y



Arc Consistency

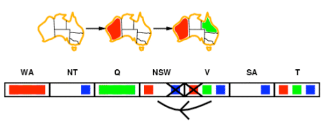
- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff
  for every value x of X there is some allowed y *for Y*



CSCI 5582 Fall 2006

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

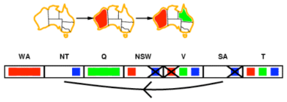


- If X loses a value, neighbors of X need to be rechecked

CSCI 5582 Fall 2006

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

CSCI 5582 Fall 2006

Informed Backtracking CSP Search

- The previous discussion didn't use any notion of heuristic.
- There are two places heuristics can help
  - Which variable to assign next
  - Which value to assign to a variable

CSCI 5582 Fall 2006

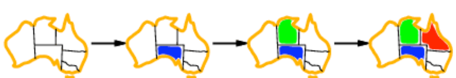
Minimum Remaining Values

- The variable with the min remaining values is the most constrained variable:

CSCI 5582 Fall 2006

Degree Heuristic

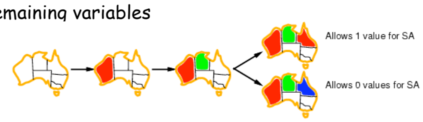
- Tie-breaker among most constrained variables (or at the start).
- Most constraining variable:
  - choose the variable with the most constraints on remaining variables



CSCI 5582 Fall 2006

Least constraining value

- 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 N-queen puzzles feasible

Admin/Break

- Questions?

Iterative Improvement

- CSPs permit a complete-state framework
- Sometimes it's better to look at these problems as optimization problems.
- Where you want to optimize (minimize) the number of constraints violated (to zero would be good)

How?

- Randomly assign values to all the variables in the problem (from their domains)
- Iteratively fix the variables (reassign values) that are conflicted.
- Continue until there are no conflicts or no progress

CSCI 5582 Fall 2006

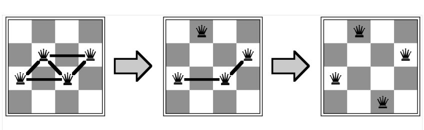
Min Conflict Heuristic

- Randomly choose a variable from among the problematic ones.
- Reassign its value to be the one that results in the fewest conflicts overall
- Continue until there are no conflicts

CSCI 5582 Fall 2006

Min Conflict Example

- States: 4 Queens, 1 per column
- Operators: Move queen in its column
- Goal test: No attacks
- Evaluation metric: Total number of attacks



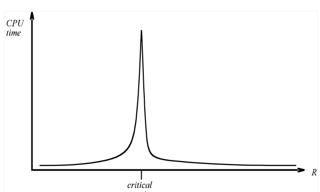
CSCI 5582 Fall 2006

Min Conflict Performance

- Amazing factoid: Min Conflict often has astounding performance.
- For example, it's been shown to solve arbitrary size (in the millions) N-Queens problems in constant time.
- This appears to hold for arbitrary CSPs with the caveat…

Min Conflict Performance

- Except in a certain critical range of the ratio constraints to variables.



Search Review

- Backtracking search
- Optimization search
- Constraint sat search

Next Time

- On to game playing
- Read Chapter 6

CSCI 5582 Fall 2006