Constraint Satisfaction Problem

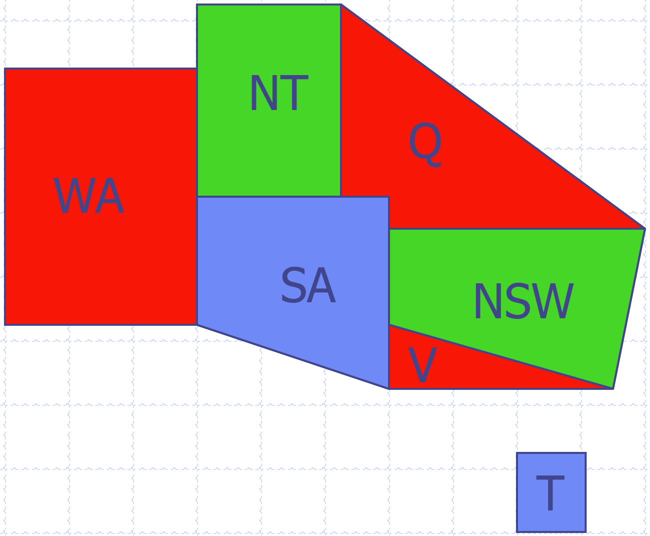
Constraint Satisfaction Problem

- Set of variables $\{X_1, X_2, \ldots, X_n\}$
- Each variable X_i has a domain D_i (also denoted D_{X_i}) of possible values
  - Usually D_i is discrete and finite
- Set of constraints $\{C_1, C_2, \ldots, C_p\}$
  - Each constraint C_k involves a subset of variables, e.g. we may write $C_k(X_i, \ldots, X_j)$ to indicate the variables involved.
  - and specifies the allowable combinations of values of these variables
- Assign a value to every variable such that all constraints are satisfied

Example: 8-Queens Problem

- 8 variables X_i, $i = 1$ to 8
- Domain for each variable {1,2,…,8}
- Constraints are of the forms:
  - $X_i = k \rightarrow X_j \neq k$ for all $j = 1$ to 8, $j \neq i$
  - $X_i = k_i, X_j = k_j \rightarrow |i-j| \neq |k_i - k_j|$
    - for all $j = 1$ to 8, $j \neq i$

Example: Map Coloring



- 7 variables {WA,NT,SA,Q,NSW,V,T}
- Each variable has the same domain {red, green, blue}
- No two adjacent variables have the same value:

WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

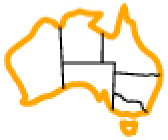
CSP as a Search Problem

- Initial state: empty assignment
- Successor function: a value is assigned to any unassigned variable, which does not conflict with the currently assigned variables
- Goal test: the assignment is complete

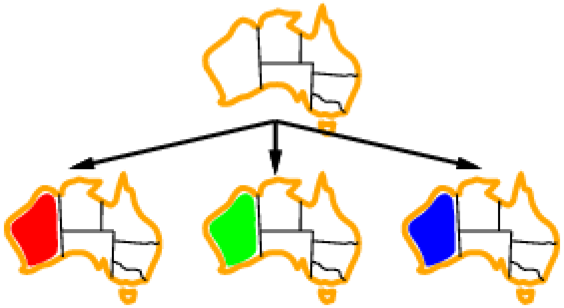
Remark

- Finite CSP include 3SAT as a special case
- 3SAT is known to be NP-complete
- So, in the worst-case, we cannot expect to solve a finite CSP in less than exponential time

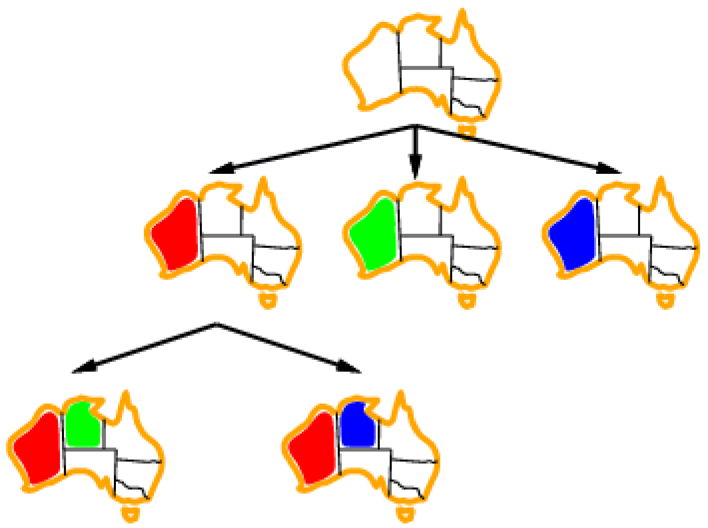
Backtracking example



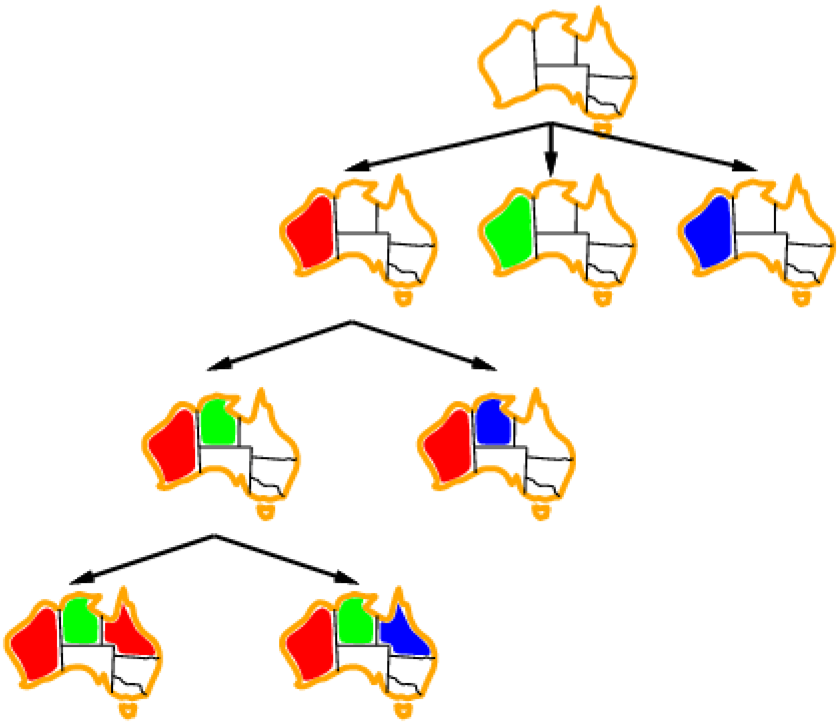
Backtracking example



Backtracking example



Backtracking illustration



Backtracking Algorithm

CSP-BACKTRACKING(PartialAssignment a)

- If a is complete then return a
- X ← select an unassigned variable
- D ← select an ordering for the domain of X
- For each value v in D do
  - If v is consistent with a then
    - Add (X= v) to a
    - result ← CSP-BACKTRACKING(a)
    - If result ≠ *failure* then return result
    - Remove (X= v) from a
- Return *failure*

Start with CSP-BACKTRACKING({})

Improving backtracking efficiency

- Which variable should be assigned next?
- In what order should its values be tried?
- Can we detect inevitable failure early?

Most constrained variable

- Most constrained variable:
  choose the variable with the fewest legal values

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

Most constraining variable

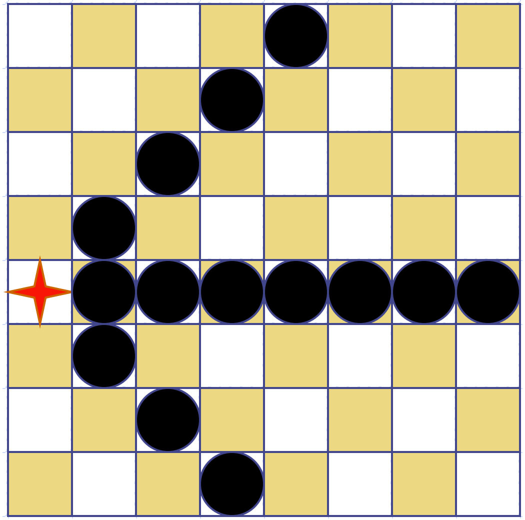
- Tie-breaker among most constrained variables
- Most constraining variable:
  - choose the variable involved in largest # of constraints on remaining variables

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 queens feasible

Forward Checking

After a variable X is assigned a value v, look at each unassigned variable Y that is connected to X by a constraint and deletes from Y's domain any value that is inconsistent with v



Constraint propagation

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

Arc-consistency

A constraint C(x,y) is said to be arc-consistent w.r.t. x if for each value v of x, there is an allowed value of y.

Similarly, we define that C(x,y) is arc-consistent w.r.t. y.

A binary CSP is arc-consistent iff every constraint C(x,y) is arc-consistent w.r.t. x as well as w.r.t. y.

Enforcing/Maintaining arc-consistency

Enforcing arc-consistency:

When a CSP is not arc-consistent, we can make it arc-consistent using an arc-consistency algorithm.

Maintaining arc-consistency:

During backtrack search, arc-consistency is maintained:

- every time when a value is assigned to a variable;
- every time a value is rejected (e.g. empty domain is generated by enforcing arc-consistency);

Of course, we can also perform arc-consistency as a pre-process.

Example

- Consider constraints:

  $X > Y, \ Y > Z$

  Domains:

  $Dx = Dy = Dz = \{1,2,3\}$

1 in Dx is removed by enforcing arc consistency, w.r.t. the constraint $X < Y$.

You can work out the rest. The resulting domains are

$D'x = \{1\}, D'y = \{2\}, D'z = \{3\}$

No search is needed in this case.

Solving a CSP

- Search:
  - can find solutions, but must examine non-solutions along the way
- Constraint Propagation:
  - Prune search space by domain reduction.
- Interleave constraint propagation and backtrack search
  - Perform constraint propagation at each search step.

Summary

- Constraint Satisfaction Problems (CSP)
- CSP as a search problem
  - Backtracking algorithm
  - General heuristics
- Forward checking
- Constraint propagation: AC
- Interweaving CP and backtracking