CSP: Variables, domains, and constraints.

- **States**: partial assignment of values to variables
- **Initial state**: the empty assignment,
- **Successor function**: Assign unassigned variable
- **Goal test**: Complete satisfying assignment.

We’ll remind ourselves of straightforward, naive approach, and then improve.
Backtracking Search

Backtracking search is the basic uninformed algorithm for solving CSPs

Idea 1: One variable at a time.
Variable assignments are commutative, so fix ordering
I.e. [WA = red then NT = green] same as [NT = green then WA = red]
Assign single variable at each step

Idea 2: Check constraints as you go.
I.e. consider values which do not conflict with previous assignments
Might have to do some computation to check the constraints
“Incremental goal test”

Depth-first search with these two improvements is called backtracking search (not the best name)
Can solve n-queens for $n \approx 25$

Backtracking Example

Backtracking Search

function BACKTRACKING-SEARCH(assignment) returns solution/false
return RECURSIVE-BACKTRACKING(assignment)
function RECURSIVE-BACKTRACKING(assignment) returns solution/false
if assignment is complete then return assignment
choose an unassigned variable $v$
for each value $v$ in ORDER-DOMAIN-VALUES($v$, assignment) do
  if value is consistent with assignment then CONTINUE($v$, $v$)
  result $\leftarrow$ RECURSIVE-BACKTRACKING(assignment)
  if result $\neq$ failure then return result
remove ($v$ = value) from assignment
return failure

CSP-Backtracking Search

CSP-Backtracking = DFS + fail-on-violation + variable-ordering
Todo:
Better “fail on violation”. Filtering.
Pick “better” variable orderings and value orderings.

An issue.

Consider the partially completed CSP assignment.
Decisions made bottom-up, left-to-right.
Let X be the decision is obviously doomed in the current assignment.
What is X?
Bonus: How many decisions before CSP-Backtracking search realizes its error?
Filtering

Filtering: Forward Checking

- Filtering:
  - Reduce domains for unassigned variables
- Forward checking:
  - Remove values that violate constraint in existing assignment

Filtering: Constraint Propagation

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

Consistency of A Single Arc

An arc \(X \rightarrow Y\) is consistent iff for every \(x\) in the tail there is some \(y\) in the head which could be assigned without violating a constraint

Video of Demo Coloring – Backtracking with Forward Checking

Video of Demo Coloring – Backtracking with Forward Checking – Complex Graph
Arc Consistency of an Entire CSP

A simple form of propagation makes sure all arcs are consistent:

Arc consistency detects failure earlier than forward checking
Important: If X loses a value, neighbors of X need to be rechecked!
What’s the downside of enforcing arc consistency?

Enforcing Arc Consistency in a CSP

Runtime: $O(n^2d^3)$, can be improved to $O(n^2d^2)$.
.....but detecting all possible future problems is NP-hard – why?

Arc Consistency: Step by step.

Function `Arc-Consistency(csp)` returns the CSP, possible with reduced domain
inputs: csp, a binary CSP with variables $\{X_1, X_2, \ldots, X_n\}$
local variables: $\text{given}$, a queue of arcs, initially all the arcs in `csp`
while $\text{given}$ is not empty do
    $(X_i, X_j)$ ← `Remove-First`($\text{given}$)
    if `Remove-Inconsistency-Value`($X_i, X_j$) then
        for each $X_k$ in `Neighbors`($X_i$) do
            if $X_k$ is $\text{given}$ then
                `Delete`($X_k, X_i$) from `given`
    end if
end while

Function `Remove-Inconsistency-Value`($X_i, X_j$) returns true iff succeeds
returns: false
for each $v_i$ in `Domain`($X_i$) do
    if no value in `Domain`($X_j$) allows $\{v_i\}$ to satisfy the constraint $X_i \rightarrow X_j$
        then delete $\text{from Domain}(X_j)$
returns: true

Video of Demo Arc Consistency – CSP Applet – n Queens

Video of Demo Coloring – Backtracking with Arc Consistency – Complex Graph
Limitations of Arc Consistency

After enforcing arc consistency:
- Can have one solution left
- Can have multiple solutions left
- Can have no solutions left (and not know it)

Arc consistency still runs inside a backtracking search!

K-Consistency

Increasing degrees of consistency
- 1-Consistency (Node Consistency): Each single node’s domain has a value which meets that node’s unary constraints
- 2-Consistency (Arc Consistency): For each pair of nodes, any consistent assignment to one can be extended to the other
- K-Consistency: For each k nodes, any consistent assignment to k-1 can be extended to the kth node.

Higher k more expensive to compute
(You need to know the k=2 case: arc consistency)

Strong K-Consistency

Strong k-consistency: also k-1, k-2, … 1 consistent
Claim: strong n-consistency means we can solve without backtracking!

Why?
- Choose any assignment to any variable
- Choose a new variable
  - By 2-consistency, there is a choice consistent with the first
- Choose a new variable
  - By 3-consistency, there is a choice consistent with the first 2
  - ...

Lots of middle ground between arc consistency and n-consistency!
(e.g. k=3, called path consistency)

Improving Backtracking

General-purpose ideas give huge gains in speed
Filtering:
- Can we detect inevitable failure early?
  - Forward/Arc/K-consistency.

Ordering:
- Which variable should be assigned next?
  - In what order should its values be tried?

Next time. Structure:
- Can we exploit the problem structure?
Ordering: Minimum Remaining Values

Variable Ordering: Minimum remaining values (MRV):
Choose the variable with the fewest legal left values in its domain

Why min rather than max?
Also called “most constrained variable”
“Fail-fast” ordering!

Ordering: Least Constraining Value

Value Ordering: Least Constraining Value.
For a variable, choose the least constraining value.
I.e., rules out the fewest values in the remaining variables
Takes computation to determine this! (E.g., rerunning filtering)
Why least rather than most?
All these ordering ideas makes 1000 queens feasible!

CSP: what to know.

CSP: variables with domains, constraints.
Model problems: coloring, n-queens, cryptoarithmetic.
Generic Algorithm.
Backtracking.
Filtering, Arc Consistency.
Variable Ordering. Minimum Remaining Values.
Value Ordering. Least Constraining Value.

Demo: Coloring – Backtracking + Forward Checking + Ordering