Today’s Topics

• Constraint Satisfaction Problems (CSP)
Recap

- **CSP**
  - $X$ is a set of **Variables**
  - $D$ is a set of **Domains**, one for each variable and consist of a set of allowable **values**
  - $C$ is a set of **Constraints** that specify allowable combinations of values

- An **assignment** that fulfills all constrains called a **consistent** or legal assignment

- A **complete assignment** is that every variable is assigned a value

- A **solution** to a CSP is a **consistent, complete** assignment

- **K-consistency**
  - 1-consistency: Node consistency
  - 2-consistency: Arc consistency
  - 3-consistency: Path consistency

- **Global Constraint**
Recap: Forward Checking

- When a variable X is assigned, for each unassigned variable Y that is connected to X by a constraint, delete any value in Y’s domain that is inconsistent with the value chosen for X.
Backtracking Search

• Basic uninformed algorithm for solving CSPs
• **One variable at a time**
  – Variable assignments are commutative (WA = red and NT = blue is the same as NT = blue, WA = red)
  – The order of any given set of actions does not matter
  – Only need to consider assignments to a single variable at each step
• **Check constraints as you go**
  – Select values that don’t conflict with any previously assigned values (assignments)
  – If no such value exist, **backtrack** and return to the previous variable and change its value
  – Might need computation to check the constraints
• **DFS** with above two improvements is called **backtracking search**
Backtracking Example
Backtracking Search Algorithm

- **Backtracking** = DFS + Variable-ordering + Fail-on-violation
- **Select-unassigned-variable** and **order-domain-values** implement the general-purpose heuristics
- **Inference** imposes forward checking, arc-, path-, or k-consistency

```plaintext
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 ← INERENCE(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

• AC-3 can reduce the domains of variables before the search
• Inference: Infer new domain reductions on the neighboring variables every time when making a choice of a value for a variable
• Forward checking:
  – One of the simplest forms of inference
  – When a variable X is assigned, for each unassigned variable Y that is connected to X by a constraint, delete any value in Y’s domain that is inconsistent with the value chosen for X
Improving Backtracking

- General-purpose ideas give huge gains in speed
- Better ordering
  - Which variable should be assigned next?
  - Which value should be tried first?
- Filtering: Can we detect failure early?
- Structure: Can we exploit the problem structure?
Ordering

- **Minimum Remaining Values (MRV)** heuristic: Choose the **variable** with the fewest legal left values in its domain
  - “Most constrained variable” heuristic
  - “Fail-first” heuristic
- **Degree** heuristic: Select the variable that is involved in the largest # of constraints
- **Least Constraining Value (LCV)** heuristic: Choose the **value** that rules out the fewest choices for the neighboring variables in the constraint graph
  - “Fail-last” heuristic
Forward checking

- Forward checking + MRV: After assigned \{WA = \text{red}\}, what next? NT or SA
- Maintaining Arc Consistency (MAC):
  - Variable $X_i$ is assigned a value
  - Inference procedure calls AC-3
  - Only the arcs $(X_j, X_i)$ for all $X_j$ that are unassigned variables that are \textit{neighbors} of $X_i$ are stored in the queue
  - More strictly than forward checking
Local Search

• Find a good state without worrying about the path to get there
• Iterative improvement
  – Start with some random assignment to values
  – Iteratively select a random conflicted variable and reassign its value to the one that violates the fewest constraints until no more constraint violations exist
• A policy known as the min-conflicts heuristic
• Generally much faster and more memory efficient (but incomplete and suboptimal)
Min-conflicts heuristic
Hill Climbing Search

- Most basic local search technique
- At each step, the current node is replaced by the best neighbor
- Greedy local search

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← problem.INITIAL
    while true do
        neighbor ← a highest-valued successor state of current
        if VALUE(neighbor) ≤ VALUE(current) then return current
        current ← neighbor
```
Hill Climbing Diagram

- Global maximum: The highest peak.
- Local maximum: Higher than its neighboring state but lower than the global maximum.
- Plateaus: A flat area of the state-space landscape.
  - A flat local maximum: No uphill exit exists.
  - Shoulder: It is a plateau that has an uphill edge.
- Current state: The region of state space diagram where we are currently present during the search.
Simulated annealing

- Escape local maximum by allowing downhill move
- If temperature T decreased slowly enough, then a property of the Boltzmann distribution, $e^{-\Delta E/T}$, is concentrated on the global maxima

```python
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
    current ← problem.INITIAL
    for t = 1 to ∞ do
        T ← schedule(t)
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← VALUE(current) – VALUE(next)
        if ΔE > 0 then
            current ← next
        else
            current ← next only with probability $e^{-\Delta E/T}$
```
MIN-CONFLICTS local search algorithm for CSPs

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 ← an initial complete assignment for csp

for i = 1 to max_steps do
    if current is a solution for csp then return current
    var ← a randomly chosen conflicted variable from csp.VARIABLES
    value ← the value v for var that minimizes CONFLICTS(csp, var, v, current)
    set var = value in current

return failure
Program Structure

- Extreme case: independent subproblems
  - Example: Tasmania and mainland do not interact
- Independent subproblems are identifiable as connected components of constraint graph
- General CSPs, where worst-case time is $O(d^n)$, $n$ variables and $d$ domains
- Suppose a graph of $n$ variables can be broken into subproblems of only $c$ variables:
  - worst-case is $O((n/c)d^c)$, linear in $n$
  - E.g. $n = 100$, $d = 3$, $c = 20$
  - $3^{100}$ vs $(5)^{3^{20}}$
Tree-Structured CSP

- One that has no loops in its constraint graph
- We can reduce the runtime from $O(d^n)$ to $O(nd^2)$
- Topological sort
TREE-CSP-SOLVER algorithm

function TREE-CSP-SOLVER(csp) returns a solution, or failure
inputs: csp, a CSP with components X, D, C

n ← number of variables in X
assignment ← an empty assignment
root ← any variable in X
X ← TOPOLOGICALSORT(X, root)

for j = n down to 2 do
    MAKE-ARC-CONSISTENT(PARENT(X_j), X_j)
    if it cannot be made consistent then return failure
for i = 1 to n do
    assignment[X_i] ← any consistent value from D_i
    if there is no consistent value then return failure
return assignment
Tree-Structured CSP Algorithm

- Order: Choose a root variable, order variables so that parents precede children (Topological sort)

- **Backward pass** of arc consistency: For $j = n$ down to 2, apply make-arc-consistent($\text{Parent}(X_j), X_j$)

- **Assign forward**: For $i = 1$ to $n$, assign $X_i$ consistently with $\text{Parent}(X_i)$
Improving Structure

- Idea: Reduce general constraint graphs to trees
- Remove nodes: Cutset conditioning
- Collapse nodes together: Tree decomposition
Cutset Conditioning

- Choose a subset S of the CSP’s variables such that the constraint graph becomes a tree after removing S (cycle cutset). Usually the smallest subset.
- For each possible assignment to the S that satisfies all constraints on S:
  - Remove any values that are inconsistent with the assignment for S from the domains of the remaining variables.
  - If there is a solution, return it together with the assignment for S.
- Runtime $O(d^c(n-c)d^2)$.
Tree Decomposition

- Every variable in the original problem appears in at least one of the tree nodes.
- If two variables are connected by a constraint in the original problem, they must appear together (along with the constraint) in at least one of the subproblems.
- If a variable appears in two nodes in the tree, it must appear in every node along the path connecting those nodes.
- Runtime $O(nd^2)$ with Tree-CSP-Solver.
Summary

• CSPs are a special kind of search problem
  – States are partial assignments
  – Goal test defined by constraints

• $X$ is a set of **Variables**

• $D$ is a set of **Domains**, one for each variable and consist of a set of allowable **values**

• $C$ is a set of **Constraints** that specify allowable combinations of values
Summary

- CSPs are represented as Constraint Graphs
- Node is Variable
- Edge: Connects any two nodes that participate in a constraint
- Unary constraint: restricts the value of a single variable \(<(SA), SA \neq \text{green}>\)
- Binary constraint: two variables. \(SA \neq NT\)
- Binary CSP: each constraint relates (at most) two variables
- Higher-order Constraints: Constraints involving three or more variables can also be represented with edges in a CSP graph
Summary

• Backtracking search: Basic uninformed algorithm for solving CSPs
• Inference:
  – Forward Checking
  – Arc Consistency
• Value ordering
  – Minimum Remaining Values (MRV)
  – Degree
  – Least Constraining Value (LCV)
• Local search: min-conflicts heuristic
• Structure:
  – Tree structured CSP
  – Cutset Conditioning
  – Tree Decomposition
Reading and Next Class

• CSP: AIMA 6.3 – 6.5
• Next: Propositional Logic: AIMA 7