Constraints Satisfaction Problems

Dr. Sandareka Wickramanayake

Reference

- Artificial Intelligence - A Modern Approach – Chapter 6 – Sections 1 and 4.

Constraint Satisfaction Problems (CSPs)

- In a standard search problem:
  - State is a "black box“ to the search algorithm – it is not aware of the internal structure of the states.
  - Internal data structure of states can only be accessed by problem-specific functions.
    - Successor function, heuristic function, and goal test
- CSP:
  - States and goal test of a CSP conforms to a standard structure and a simple representation
    - This allows search algorithms to take advantage of the structure of states and use general-purpose heuristics instead of problem-specific ones.

Constraint Satisfaction Problems(CSPs)

- CSP is defined by
  - A set of variables $X = \{X_1, X_2, \ldots, X_n\}$, where each X_i, can take values from domain D_i
  - A set of constraints, $C = \{C_1, C_2, \ldots, C_m\}$
- *A* domain, D_i, consists of a set of allowable values, $\{v_1, v_2, \ldots, v_k\}$ for variable X_i
- E.g., if X_i is Boolean the domain is *{true, false}*
- Different variables can have different domains of different sizes.

Constraint Satisfaction Problems(CSPs)

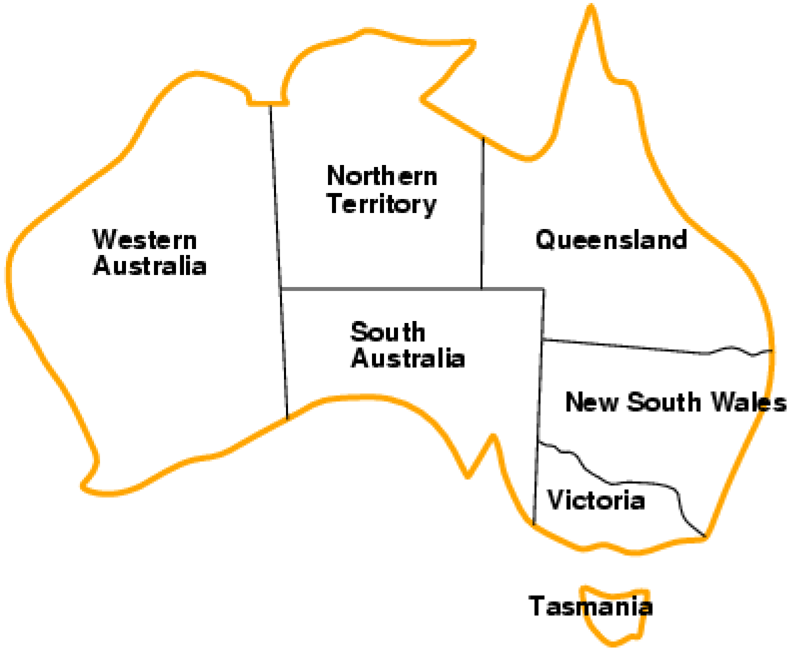
- Each constraint C_j involves a subset of X and specifies legal combinations of values for that subset
- A state is defined by an assignment of values to all or some of the variables, $\{X_i = v_i, X_j = v_j, \dots\}$
  - E.g., If X_1 and X_2 both have the domain {1,2,3}, then the constraint saying that X_1 must be greater than X_2 can be written as $\langle (X_1, X_2), \{(3,1), (3,2), (2,1)\} \rangle$ or $\langle (X_1, X_2), X_1 > X_2 \rangle$

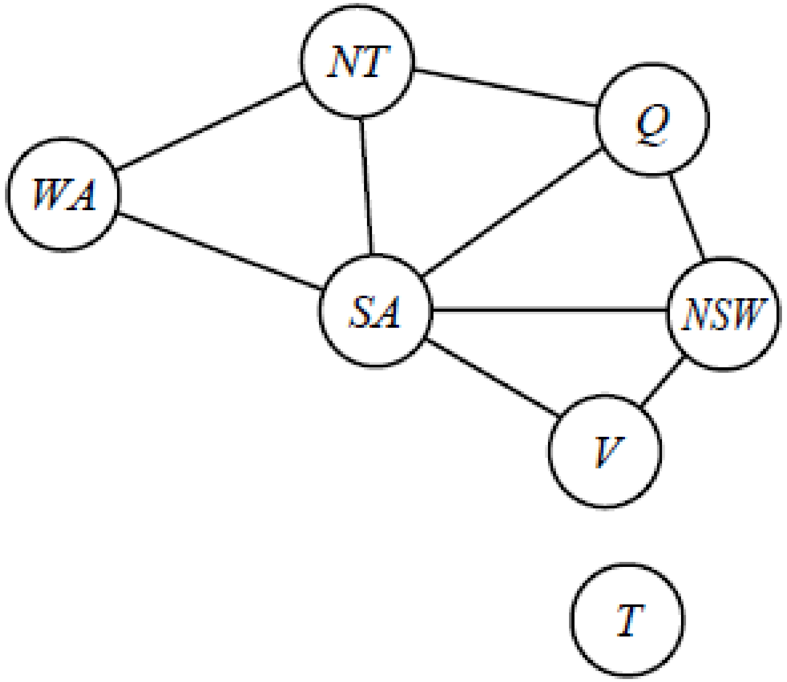
Constraint Satisfaction Problems(CSPs)

- An assignment that doesn't violate any constraint is called a consistent or legal assignment.
- If every variable is assigned a value, it is a complete assignment.

- A solution to a CSP is a complete and consistent assignment.
  - E.g., One that has all variables assigned with values and satisfies all the constraints

Example: Map-Coloring



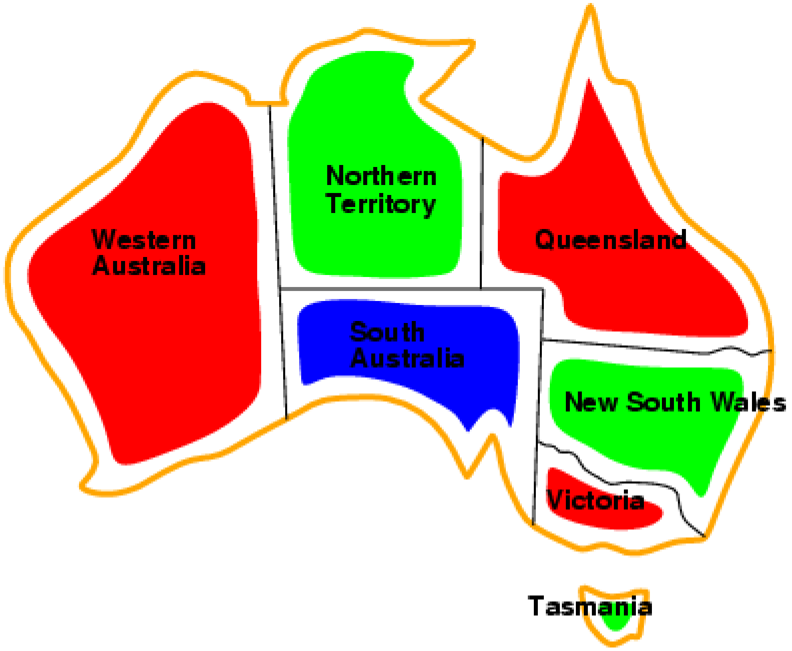


Constraint graph

Nodes – Variables
Edges – Connect any two variables that participate in a constraint.

- 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, Q ≠ NW, … etc.
  - Legal values under the constraint WA ≠ NT are;
    (WT,NT) ϵ {(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.

Why Formulate a Problem as a CSP?

- Provide a natural representation for a wide variety of problems.
- CPS solvers are fast and efficient.
- Can quickly eliminate a large portion of the search space that violates the constraints which an atomic state-space searcher cannot.
  - E.g., Once we have chosen SA = blue in the Australia problem, we can conclude that none of the five neighboring variables can take on the value.

Real-world CSPs

- Class Assignment problems
  - E.g., who teaches what class
- Timetabling problems
  - E.g., which class is offered when and where?
- Transportation Scheduling
- Factory Scheduling

Notice that many real-world problems involve real-valued variables

Variations on the CSP Formalism

- Type of variables
  - Discrete variables
    - Finite domains:
      - n variables, each having a domain of size d, leads to $O(d^n)$ possible complete assignments
      - E.g., Map coloring problems and 8-queens
    - Infinite domains:
      - Integers, strings, etc.
      - E.g., job scheduling, where variables are start/end days for each job
  - Continuous variables
    - E.g., start/end times for Hubble Space Telescope observations
    - Liner programming.

Variations on the CSP Formalism

- Type of constraints
  - Unary constraints involving a single variable.
    - E.g., SA ≠ green
  - Binary constraints involving pairs of variables.
    - E.g., SA ≠ WA
  - Global constraints involving an arbitrary number of variables.

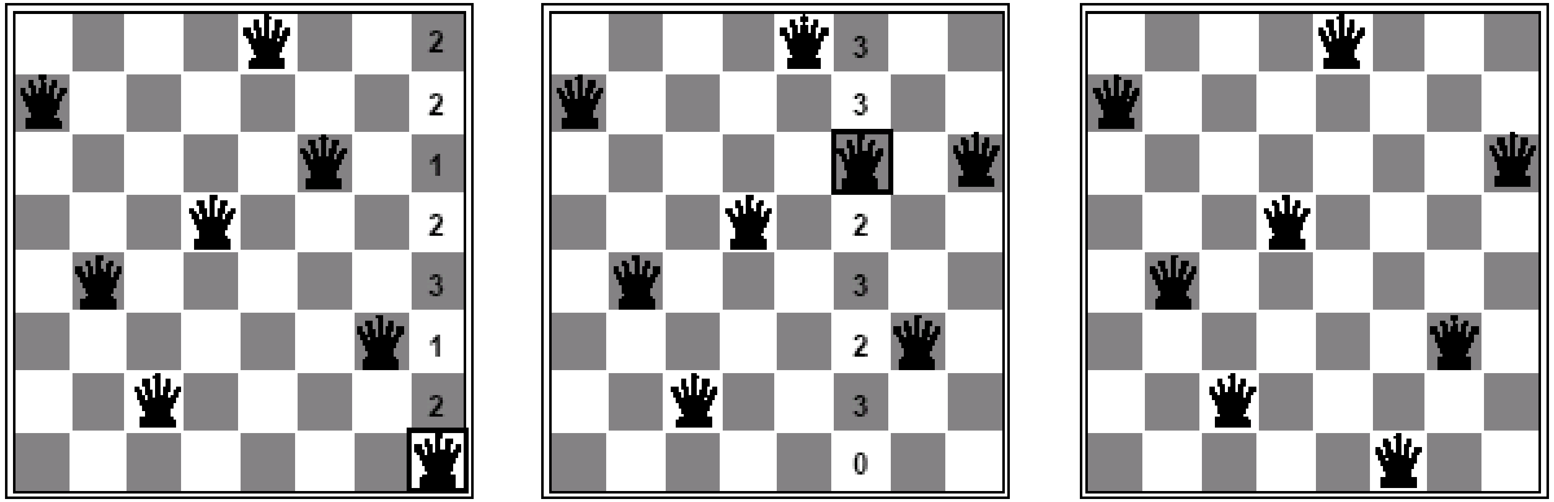
Local Search for CSPs

- Hill-climbing, simulated annealing, and others can be used for CSPs
  - Typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
  - Allow states with unsatisfied constraints
  - Operators reassign variable values

- Initial state: Some assignment to all variables. E.g., random
- Successor function: Usually changes the value of a single variable
- Variable selection: Randomly select any conflicted variable
- Value selection by min-conflicts heuristic:
  - Choose a value that violates the fewest constraints
    - E.g., hill-climb with $h(n)$ = total number of violated constraints

Min-conflicts Example

- A two-step solution for an 8-queens problem using min-conflicts heuristic
- At each stage a queen is chosen for reassignment in its column
- The algorithm moves the queen to the min-conflict square, breaking ties randomly.



Local Search for CSPs

```
function MIN-CONFLICTS(csp, max_steps) return 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 VARIABLES[csp]
     value  ←  the value v for var that minimize
 CONFLICTS(var,v,current,csp)
     set var = value in current
 return failure
```