Artificial Intelligence

6. CSP

Shashi Prabh

School of Engineering and Applied Science
Ahmedabad University

Contents

Goal: use factored representation of agents to solve problems.

Topics
- Constraint Satisfaction Problem
- Constraint Propagation
- Backtracking Search
- Local Serach

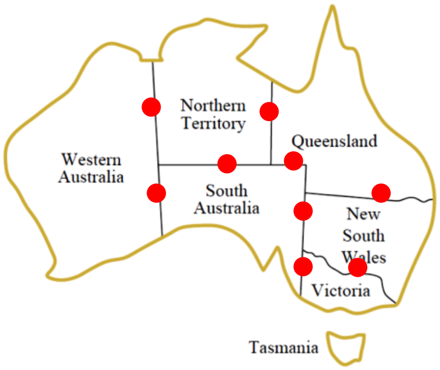
Constraint Satisfaction Problems (CSP)

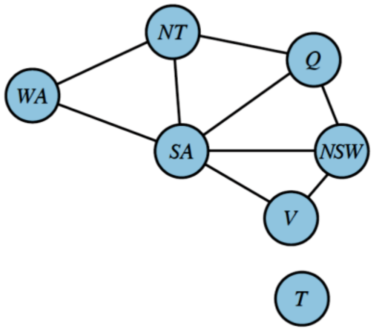
- We consider factored representation of states
  - A state is a set of variables
- A problem solution is an assignment of values to the state variables where all the constraints on the variables are satisfied
- Why CSP?
  - CSP is a natural formulation in many problems
    - Scheduling, planning, resource allocation, temporal models, control etc.
  - Significant reduction of search space, availability of fast solvers
  - Insight into the problem structure can be used for search speed-up
    - Some intractable atomic search-space problems can be quickly solved as CSP formulation
  - Actions and transition model can be deduced from the formulation

Constraint Satisfaction Problems (CSP)

- A CSP consists of three components (X, D, C):
- Variables $X = \{ x_1, x_2, \ldots, x_n\}$
- Domains $D = \{D_1, D_2, \ldots, D_n\}$
- Constraints $C = \{c_1, c_2, \ldots, c_m\}$
  - Domain D_i consists of the set of allowable values $\{v_1, \ldots, v_k\}$ for each x_i
    - {T, F} for a Boolean variable
  - Constraint c_j consists of a pair ⟨scope, relation⟩
    - $\langle (x_1, x_2), x_1 \neq x_2 \rangle$ or just $x_1 \neq x_2$
- Goal: Assign values to the variables from their respective domains such that all the constraints are satisfied
  - An assignment that does not violate any constraint is called consistent or legal assignment
  - A solution to a CSP is a complete and consistent assignment

Map coloring





- X = {W, N, S, Q, NSW, V, T}
- D = {r, g, b}
- C = { W ≠ N, S ≠ N, Q ≠ N, W ≠ S, S ≠ Q, etc}
  - W ≠ N means {(r, g), (r, g), (g, r), (g, b), (b, r), (b, g)}
  - Note the reduced search space due to the constraints: 2^5 instead of 3^5
- Can you find one solution?
- In a CSP constraint graph, two variables are connected by an edge if there is a constraint that involves both

Job-Shop Scheduling – Car Assembly

- X is the set of tasks

$$\{Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, Nuts_{RF}, Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\}$$

- Values are the start times of tasks: D_i = {0, 1, …, 30}
- Constraints: precedence constraints and completion times
  - It takes 10 minutes to install an axle:

$$Axle_F + 10 \leq Wheel_{RF}; \quad Axle_F + 10 \leq Wheel_{LF};$$
$$Axle_B + 10 \leq Wheel_{RB}; \quad Axle_B + 10 \leq Wheel_{LB}.$$

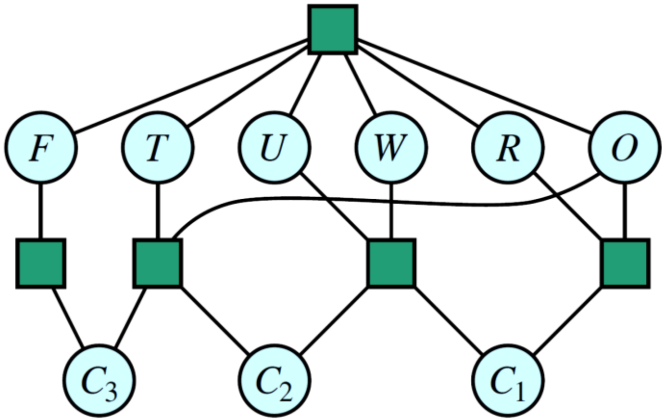
  - Axle installations must not overlap in time:

$$(Axle_F + 10 \leq Axle_B) \quad \textbf{or} \quad (Axle_B + 10 \leq Axle_F)$$

  - Exercise: CSP formulation of 8-Queens problem

Cryptarithmetic Puzzles

$$
\begin{array}{cccc}
 & T & W & O \\
+ & T & W & O \\
\hline
F & O & U & R
\end{array}
$$



Constraint hypergraph

- Constraints: AllDiff (F, T, U, W, R, O), F ≠ 0 and

$$O + O = R + 10 \cdot C_1$$
$$C_1 + W + W = U + 10 \cdot C_2$$
$$C_2 + T + T = O + 10 \cdot C_3$$
$$C_3 = F \,,$$

Inference

- State-space search, generating successors as new assignments
- Constraint propagation is an alternative where constraints are enforced locally on the constraint graph
    - Local consistency shrinks the search space by eliminating the inconsistent assignments
    - Used along-with search and/or as a preprocessing step
- Types of local consistency
    - Node consistency
    - Arc consistency
        - Path and K-Consistency
- Global constraints, bounds propagation

Node Consistency

- A node in the constraint graph is node-consistent if all the values in the variable's domain satisfy the variable's unary constraints.
- Example: consider a unary constraint SA ≠ {green}
  - The variable SA with initial domain {red, green, blue} can be made node consistent by eliminating green from its domain, leaving SA with the reduced domain {red, blue}.
- A graph is node-consistent if every variable in the graph is node-consistent.
- Instead of node consistency, one can eliminate domain values inconsistent with unary constraints

Arc Consistency

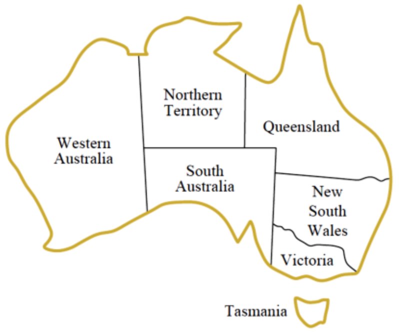
- A variable is arc-consistent if for every value in its domain, there is some value in the domains of all the variables connected by a binary constraint
- Example: consider the constraint $Y = X^2$, $D_X = \mathbb{N}$, $D_Y = \{ 0, 1, 4, 9\}$
  - X is made arc-consistent with Y by restricting $D_X = \{ 0, 1, 2, 3\}$
- However, arc-consistency is ineffective in the map coloring example
- Algorithm called AC-3 is a widely used arc-consistency algorithm

AC-3 (Mackworth, 1977)

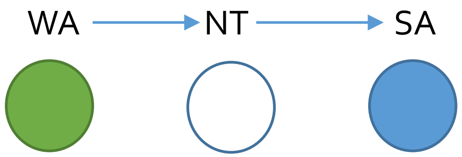
```
function AC-3(csp) returns false if an inconsistency is found and true otherwise
  queue ← a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← POP(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true

function REVISE(csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

- Initially, each binary constraint inserts two arcs
- X_i is being made consistent with X_j
- O $(c\ d^3)$ worst case complexity

Path Consistency



- AC does not help with map coloring
  - Does not object to 2-coloring the map
- A two-variable set $\{X_i, X_j\}$ is path-consistent with respect to a third variable X_m if, for every assignment $\{X_i = a, X_j = b\}$ consistent with the constraints (if any) on $\{X_i, X_j\}$, there is an assignment to X_m that satisfies the constraints on $\{X_i, X_m\}$ and $\{X_m, X_j\}$.
  - Refers to the overall consistency of the path from X_i to X_j with X_m in the middle
- Can infer no valid 2-coloring of the Australia map



K-Consistency

- A CSP is k-consistent if, for any set of k−1 variables and for any consistent assignment to those variables, a consistent value can always be assigned to any k^{th} variable
  - 1-consistency says that, given the empty set, we can make any set of one variable consistent: this is what we called node consistency
  - 2-consistency is the same as arc consistency
  - 3-consistency (binary constraints) is the same as path consistency
- A CSP is strongly k-consistent if it is k-consistent and is also (k−1)-consistent, (k−2), . . . all the way down to 1-consistent
  - Can design a greedy algorithm
- CSP is NP-complete
  - K-consistency requires exponential time and space

Global constraints

- A global constraint involves an arbitrary number of variables. It is more efficient to handle these by special-purpose algorithms
- AllDiff: if m variables are involved in an AllDiff constraint, and if n possible distinct values altogether are available, then the constraint cannot be satisfied if m > n
- Atmost: resource constraint
  - Example: no more than 10 personnel are scheduled in total
  - We can detect an inconsistency simply by checking the sum of the minimum values of the current domains

Global constraints

- Bounds propagation: For problems with large integer domains it is usually not efficient to represent the domain of each variable as a large set of integers.
  - Domains can be represented by upper and lower bounds and managed by bounds propagation
- Example:
  - Consider two flights, F1 and F2, for which the planes have capacities 165 and 385, respectively
  - The initial domains for the numbers of passengers are then D1 = [0, 165] and D2 = [0, 385]
  - The additional constraint that the two flights together must carry 450 people can be handled by propagating bounds constraints as D1 = [65, 165] and D2 = [285, 385]

Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

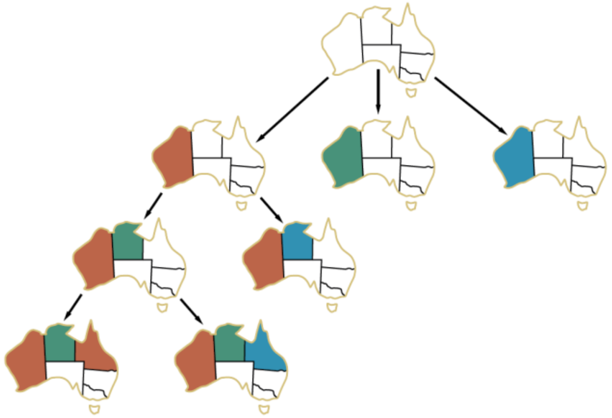
	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

Exercise: Write CSP formulation!

Backtracking Search

- Search for solution is needed when after constraint propagation there exist variables with multiple possible vlaues
- For a CSP with n variables of domain size d results in a search tree where all the complete assignments are ~~n!~~ d^n leaf nodes at depth n
  - The branching factor at the top would be nd, at the next level (n-1) d and so on, but the order of assignments does not matter

Backtracking Search



- Backtracking search progresses via a recursive call
- An unassigned variable is (repeatedly) chosen, a value is assigned and the search progresses to another variable and so on
  - If the search succeeds, the solution is returned
  - If the search fails, the assignment is restored to the previous state, and the next value is tried
- BACKTRACKING-SEARCH keeps only a single representation of a state (assignment) and alters that representation rather than creating new ones

Backtracking Search

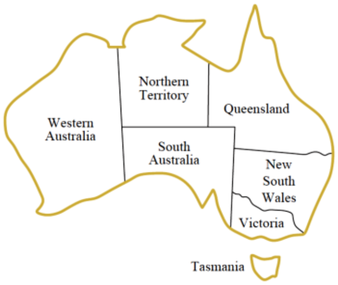
```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

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 ← INFERENCE(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
```

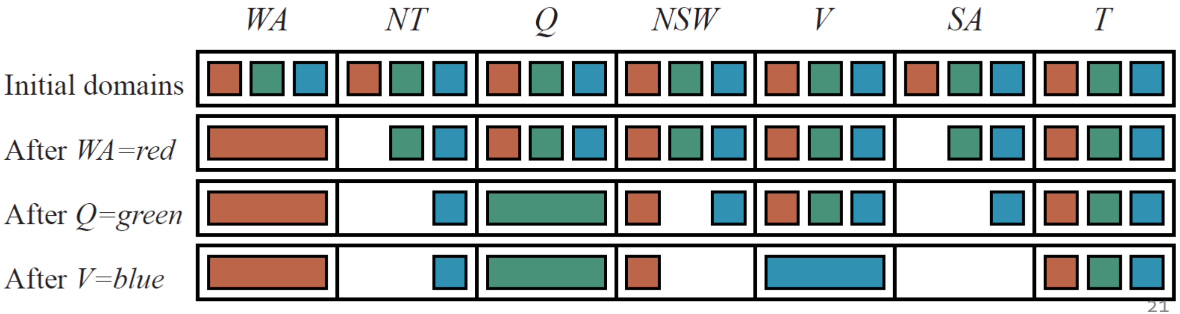
Improving Backtracking Search

- Backtracking search can be improved using domain-independent heuristics that take advantage of the factored representation of states
- Variable and value ordering heuristics
  - Minimum-remaining-values heuristic
    - Start with F in crypatrithmetic puzzle
  - Degree heuristic – largest first
    - Start with SA in Australia map
  - Least constraining value first
    - Values that rule out the fewest choices first

Interleaved Search and Inference

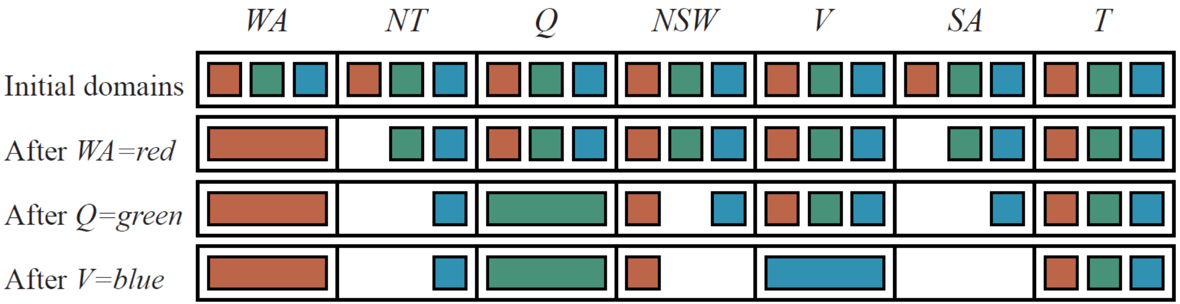
- Forward Checking: Check for arc consistancy upon a variable assignment
  - Upon assignment to X, for each unassigned variable Y that is connected to X by a constraint, delete from Y’s domain any value that is inconsistent with the value chosen for X
    - After assigning V =blue, the domain of SA is empty indicating that the partial assignment {WA=red, Q=green, V =blue} is inconsistent with the constraints. At this point the algorithm backtracks.





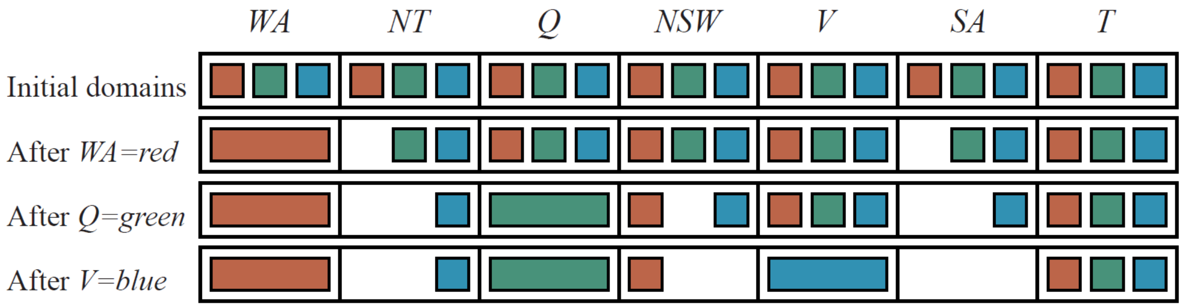
Interleaved Search and Inference

- Combining the MRV heuristic with forward checking is usually more effective
  - After assigning {WA=red} NT and SA each have two values. MRV will choose one of them first and then the other. After that Q, NSW and V.
- Forward checking incrementally computes the information that the MRV heuristic needs…



Interleaved Search and Inference

- Forward checking doesn't detect all inconsistencies since it does not look ahead far enough
  - In the Q=green row, WA and Q arc-consistent, but both NT and SA are left with blue as their only possible value, which is an inconsistency, since they are neighbors.



Interleaved Search and Inference

- Maintaining Arc Consistency (MAC): After a variable X_i is assigned a value, the inference procedure calls AC-3
  - Instead of a queue of all the arcs, it starts with only the arcs (X_j, X_i) for all X_j that are unassigned variables and are neighbors of X_i
  - If any variable has its domain reduced to the empty set, the call to AC-3 fails which triggers backtracking immediately
- We can see that MAC is strictly more powerful than forward checking unlike MAC, forward checking does not recursively propagate constraints

- Reading: Chapter 6
- Assignments: PS 4, csp.ipynb

- Next: Logical Agents, Chapter 7

- Mid-Term Examination coming up