Constraint Satisfaction Problems (CSP)

CS 4804 Fall 2020
Virginia Tech
Today’s Topics

- Constraint Satisfaction Problems (CSP)
Constraint Satisfaction Problem (CSP)

- A special subset of search problems
- State is represented by **factored representation**
  - A vector of attribute values
  - Boolean, real-valued, etc.
- Goal test: When each variable has a value that satisfies all the constraints on the variable
- A type of **identification problem**
CSP Examples

- Assign red, green, or blue color to different states on the Australia map
- Adjacent states must have different colors
Planning vs Identification

- **Planning**: sequences of actions
  - Path to the goal
  - Each path has cost
  - Depths of search tree
  - Domain-specific heuristics that guide agent

- **Identification**: *assignments to variables*
  - The goal itself is important, not the path
  - All paths at the same depth (for some formulations)
  - CSPs are specialized identification problems
Components

• 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
Assignments

- Variable $X_1$ has domain $\{1,2,3\}$
- Variable $X_2$ has domain $\{1,2,3\}$
- Constraint $C <(X_1, X_2), X_1 > X_2>$
- An **assignment** that fulfills all constrains called a **consistent** or legal assignment
- A **complete assignment** is when every variable is assigned a value
- A **solution** to a CSP is a **consistent, complete** assignment
  - $\{(2,1),(3,2)\}$
- A **partial assignment** has some variables unassigned.
- A **partial solution** is a partial assignment that is consistent
- CSP is an **NP-complete** problem, which means that exists no known algorithm for finding solutions in polynomial time
CSP Example: Map Coloring

- **Variables**: WA, NA, Q, NSW, V, SA, T
- **Domains**: \{red, green, blue\}
- **Constraints**: adjacent regions must have different colors
  - e.g. \<(SA,WA), SA != WA >
  - \{(r,g),(g,b),(r,b),(g,r),...\}
- **Goal Test**: Does this assignment satisfies all constraints?
- **Solutions are assignments satisfying all constraints**
Real-World CSPs

- Meeting schedule problem
- Transportation schedule problem
- Hardware configuration problem
- Assignment problem
- Factory job schedule problem
- Sudoku puzzle
- Many many more!
Constraint Graph

- 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
- $T$ is an independent subproblem
CSPs - Varieties of Domains

- Simplest: discrete and finite domains
  - Map coloring, Scheduling with time limits, 8-queens problem
- Discrete with infinite domains (integers, strings, etc.)
  - E.g. Scheduling with no deadline. (Infinite start time)
  - Linear constraints is solvable
  - Nonlinear constraints is undecidable
- Continuous domains
  - E.g. Start/End times for Hubble Telescope observations
  - Linear constraints is solvable in polynomial time by Linear programming
CSPs - Varieties of Constraints

• Simplest: unary constraint
  – Restricts the value of a single variable. E.g. SA != green

• Binary constraints:
  – Relates two variables. E.g. SA != NT

• Higher-order constraints:
  – Involves three or more variables. E.g. <(X,Y,Z), X<Y<Z>

• Global constraint:
  – Arbitrary number of variables
  – Alldiff: All the variables in the constraint must have different values. (Sudoku)
Cryptarithmetic

\[ O + O = R + 10 \cdot C_{10} \]
\[ C_{10} + W + W = U + 10 \cdot C_{100} \]
\[ C_{100} + T + T = O + 10 \cdot C_{1000} \]
\[ C_{1000} = F \]
Absolute vs Preference constraints

• Indicates which solutions are preferred
• Class-scheduling problem
  – Absolute: Professor can only teach one class at one time
  – Preference:
    • Prof. R prefers teaching in the morning
    • Prof. C prefers teaching in the afternoon
    • A schedule that has Prof. R teaching at 2pm would still be an allowable solution, just not the optimal one
    • Often represented as cost for assignment
    • A constrained optimization problem, or COP
Standard Search Formulation

• States defined by the values assigned so far (partial assignments)
  – Initial State: the empty assignment {}  
  – Successor function: assigns a value to an unassigned variable  
  – Goal test: The current assignment is complete and satisfies all constraints
Constraint Propagation

- Atomic state-space search algorithm: Expands a node to visit the successors
- CSP algorithm:
  - Generate successors by choosing a new variable assignment
  - Constraint propagation: a specific type of inference
- Constraint propagation:
  - Uses the constraints to reduce the number of legal values for variable
  - Have fewer choices to consider for the next choice of a variable assignment
  - May be done as a preprocessing step, before search starts
  - Local consistency
Node Consistency

• If all the values in the variable’s domain satisfy the variable’s unary constraints
  – \(<(SA), \text{SA} \neq \text{green}> \Rightarrow \text{SA} \{\text{red}, \text{blue}\}\)

• Each single node’s domain has a value which meets that node’s unary constraints

• A graph is node-consistent if every variable in the graph is node-consistent
  – \(<(SA), \text{SA} \neq \text{green}>, <(WA), \text{WA} \neq \text{red}>, <(NT), \text{NT} \neq \text{red}>, <(Q), \text{Q} \neq \text{red}>, <(NSW), \text{NSW} \neq \text{red}>, <(V), \text{V} \neq \text{red}>, <(T), \text{T} \neq \text{green}>\)
Arc Consistency

• If every value in the variable’s domain satisfies the variable’s binary constraints
• For each pair of nodes, any consistent assignment to one can be extended to the other
  – Variable $X_i$ is arc-consistent with another variable $X_j$
  – For every value in $D_i$, you can find a value in $D_j$ that can satisfy the binary constraint on the arc $(X_i, X_j)$
  – E.g. $X \{0, 1, 2, 3\}$, $Y \{0, 1, 4, 9\}$, Binary constraint $Y = X^2$
• A graph is arc-consistency if every variable is arc-consistency with every other variable.
• [Arc consistency Demo]
Path Consistency

- Tightens the binary constrains by looking at triples of variables
- A set \( \{X_i, X_j\} \) is path-consistent with respect to a third variable \( X_m \) if every assignment \( \{X_i=a, X_j=b\} \) is consistent with the constraints on \( \{X_i, X_j\} \), there is an assignment to \( X_m \) that satisfies the constraints on \( \{X_i, X_m\} \) and \( \{X_m, X_j\} \).
K-consistency

• Increasing degrees of consistency
  – 1-consistency: Node consistency
  – 2-consistency: Arc consistency
  – 3-consistency: Path consistency
• K-Consistency: For each k nodes, any consistent assignment to k-1 node can be extended to the kth node.
• K-consistent CSP: any set of k-1 variable and any consistent assignment
• Strong k-consistent CSP: It is k-consistent, (k-1)-consistent, (k-2)-consistent, …1-consistent
Global Constraint

- Occurs frequency in real problems
- Can be handled by special-purpose algorithms that are more efficient than the general-purpose algorithms
- Alldiff constraint
- Sudoku: Demo
Solving CSPs

• BFS?
• DFS?
• What are the problems?
Filtering: Forward Checking Step by Step

- Filtering: keep track of domains for unassigned variables and cross off bad options
- Forward checking: cross off values that violate the constraint when added to the existing assignment
Filtering: Forward Checking Step by Step

5 uncolored states

WA | NT | Q | NSW | V | SA | T
Filtering: Forward Checking Step by Step

5 uncolored states

WA  NT  Q  NSW  V  SA  T
Filtering: Forward Checking Step by Step
Filtering: Forward Checking Step by Step

5 uncolored states
Filtering: Forward Checking Step by Step

5 uncolored states
Filtering: Forward Checking Step by Step

5 uncolored states
Filtering: Forward Checking Step by Step

5 uncolored states

WA  NT  Q  NSW  V  SA  T
Filtering: Forward Checking Step by Step

5 uncolored states
Filtering: Forward Checking Step by Step
Filtering: Forward Checking Step by Step

5 uncolored states
Filtering: Forward Checking Step by Step

4 uncolored states
Filtering: Forward Checking Step by Step

4 uncolored states

[Diagram showing the states WA, NT, Q, NSW, V, SA, T with colored segments representing uncolored states]
Filtering: Forward Checking Step by Step

4 uncolored states
Filtering: Constraint Propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn’t provide early detection for all failures
- Arc consistency detects failure earlier than forward checking
AC-3 Algorithm

• First stores all arcs in the CSP in a queue $Q$.
  – Each binary constraint becomes two arcs, one in each direction. $Q = [SA \rightarrow V, V \rightarrow SA, ... ]$
• Remove arc $(X_i, X_j)$ from the $Q$ and make $X_i$ arc-consistent with $X_j$
• Continue remove values from the domains of variables until queue is empty
• Output: an arc-consistent CSP that have smaller domains, or no solution exists
• Time complexity: $O(cd^3)$. $c$ is the number of arcs and $d$ is the size of the largest domain.
AC-3 Algorithm

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
Arc Consistency: Step by Step

Q = [SA→V, V →SA, SA→NSW, NSW →SA, SA→NT, NT →SA, V →NSW, NSW →V]

Note: Take out value from the tail of the arc
Arc Consistency: Step by Step

\[ Q = [V \rightarrow SA, SA \rightarrow NSW, NSW \rightarrow SA, SA \rightarrow NT, NT \rightarrow SA, V \rightarrow NSW, NSW \rightarrow V, SA \rightarrow V] \]
Arc Consistency: Step by Step

$Q = [SA \rightarrow NSW, NSW \rightarrow SA, SA \rightarrow NT, NT \rightarrow SA, V \rightarrow NSW, NSW \rightarrow V, SA \rightarrow V]$
Arc Consistency: Step by Step

Q=[\text{NSW} \rightarrow \text{SA}, \text{SA} \rightarrow \text{NT}, \text{NT} \rightarrow \text{SA}, \text{V} \\
\rightarrow \text{NSW}, \text{NSW} \rightarrow \text{V}, \text{SA} \rightarrow \text{V}, \text{SA} \rightarrow \text{NSW}]

5 uncolored states
Arc Consistency: Step by Step

Q = [SA → NT, NT → SA, V → NSW, NSW → V, SA → V, SA → NSW]

5 uncolored states
Arc Consistency: Step by Step

Q=[SA→NT, NT→SA, V→NSW, NSW→V, SA→V, SA→NSW]

Failure detected!
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
Reading and Next Class

- CSP: AIMA 6.1-6.2
- Next: CSP: AIMA 6.3-6.5