CS 4804: Intro to AI

Constraint Satisfaction Problems (CSP)

Virginia Tech CS 4804 Fall 2021
Instructor: Yinlin Chen
Today’s Topics

- Constraint Satisfaction Problems (CSP)
Recap

• Player is maximizing expected estimated value
  1. What is Player’s expected value if she takes the expectimax optimal action?
  2. What is the worst possible payoff she could see from that action?

• Player now only considers actions whose worst-case outcome is 10 or better
  3. Which action does the Player choose for this tree?
  4. What is the expected value for that action?
  5. What is the worst value possible for that action?
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 most important is to archive goal, not path
  - All paths at same depth (for some formulations)
  - CSPs are specialized identification problems
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
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 constraints called a consistent or legal assignment
- A partial assignment has some variables unassigned.
- A complete assignment is that every variable is assigned a value
- A solution to a CSP is a consistent, complete assignment
  - \{(2,1),(3,2)\}
- 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. \(<(SA,WA), \text{SA} \neq \text{WA}>\)
  - \{\(r,g\),(g,b),(r,b),(g,r),\ldots\}\n- **Goal Test**: Does this assignment satisfies all constraints?
- **Solutions** are assignments satisfying all constraints
CSP Example: A Solution
Real-World CSPs

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

- **Variables:**
  - Each (open) square
- **Domains:**
  - \{1,2,...,9\}
- **Constraints:**
  - 9-way alldiff for each column
  - 9-way alldiff for each row
  - 9-way alldiff for each region
  - (or can have a bunch of pairwise inequality constraints)
Example: Cryptarithmetic

Variables: F T U Q R O C₁ C₂ C₃

Domains: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Constraints: alldiff(F, T, U, W, R, O)

\[
\begin{align*}
O + O &= R + 10 \cdot C_{10} \\
C_{10} + W + W &= U + 10 \cdot C_{100} \\
C_{100} + T + T &= O + 10 \cdot C_{1000} \\
C_{1000} &= F,
\end{align*}
\]
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)
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 prefer teaching in the morning
    • Prof. C prefer teaching in the afternoon
    • A schedule that has Prof. R teaching at 2pm would still be an allowable solution, just not be an 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: assign a value to an unassigned variable
  – Goal test: The current assignment is completed 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
  – \(<(SA), \text{SA} \neq \text{green}> \rightarrow \text{SA} \{\text{red, blue}\}\)

• 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 Xm if every assignment \{X_i=a, X_j=b\} consistent with the constraints on \{X_i, X_j\}, there is an assignment to Xm that satisfies the constraints on \{X_i, Xm\} 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

- Occur 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

- 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

5 uncolored states
Filtering: Forward Checking
Filtering: Forward Checking

5 uncolored states
Filtering: Forward Checking

5 uncolored states
Filtering: Forward Checking

5 uncolored states
Filtering: Forward Checking

5 uncolored states

WA
NT
Q
NSW
V
SA
T
Filtering: Forward Checking

5 uncolored states
Filtering: Forward Checking

5 uncolored states
Filtering: Forward Checking

5 uncolored states
Filtering: Forward Checking

4 uncolored states
Filtering: Forward Checking

4 uncolored states
Filtering: Forward Checking

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.
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

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

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

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

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

5 uncolored states

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