CSL020U4E: Artificial Intelligence
Lecture-08 (Constraint Satisfaction)

Sumit Kumar Pandey

November, 2025

Constraint Network or Constraint Satisfaction Problem

Constraint Network or Constraint Satisfaction Problem

A constraint network $\mathcal{R}$ or a CSP (constraint satisfaction problem) is a triple,

$$\mathcal{R} = \langle X, D, C \rangle$$

where

1. X is a set of variable names, $X = \{x_1, x_2, \ldots, x_n\}$.
2. D is a set of domains, one for each variable, $D = \{D_1, D_2, \ldots, D_n\}$ and
3. C is a set of constraints on some subsets of variables. $C = \{C_1, C_2, \ldots, C_m\}$ where
    - $C_i = \langle S_i, R_i \rangle$, $S_i = \{x_{i_1}, x_{i_2}, \ldots, x_{i_l}\} \subseteq X$ and $R_i \subseteq D_{i_1} \times D_{i_2} \times \ldots \times D_{i_l}$.

Constraint Satisfaction Problem

We will confine ourselves to finite domain CSPs in which the domain of each variable is discrete and finite.

Constraint Satisfaction Problem

Assignment or Instantiation

An assignment $\mathcal{A}$ is set of variable-value pairs, for example, $\{x_2 = a_{21}, x_4 = a_{45}, x_7 = a_{72}\}$. We also say that the assignment is an instantiation of the set of variables.

Partial Assignment

An assignment to a subset of variables is a partial assignment.

An Assignment Satisfies a Constraint

An assignment satisfies a constraint C_i if $S_i \subseteq \{x_1, x_2, \ldots, x_p\}$ and $\mathcal{A}_{S_i} \in R_i$ where $\mathcal{A}_{Si}$ is the projection of $\mathcal{A}$ onto S_i.

Consistent Assignment

An assignment $\mathcal{A}$ with scope S is consistent if it satisfies all the constraints whose scope is covered by $\mathcal{A}$.

Constraint Satisfaction Problem

Solution to a CSP

A solution to a CSP is consistent assignment over all the variables in X.

Constraint Graph and Matching Diagram

- Every CSP can be depicted as a constraint graph.
- The nodes in the graph are the variables in the CSP and an edge between two nodes says that the two variables participate in a constraint.
- This is true even when the constraint is ternary or higher.
- Another diagram that is useful is the **matching diagram**.
- An edge in the matching diagram connects two values in two variables that together participate in some constraint.

Backtracking

```
Backtracking$(X, D, C)$
  $\mathcal{A} \leftarrow [\,]$
  $i \leftarrow 1$
  $D'_i \leftarrow D_i$
  while $1 \leq i \leq N$
    $a_i \leftarrow$ SelectValue$(D'_i, \mathcal{A}, C)$
    if $a_i =$ null then
      $i \leftarrow i - 1$
      $\mathcal{A} \leftarrow$ tail $\mathcal{A}$
    else
      $\mathcal{A} \leftarrow a_i : \mathcal{A}$
      $i \leftarrow i + 1$
      if $i \leq N$
        then $D'_i \leftarrow D_i$
  return Reverse$(\mathcal{A})$
```

Backtracking

```
SelectValue($D'_i, \mathcal{A}, \mathcal{C}$)
  while $D_i$ is not empty
    $a_i \leftarrow$ head $D'_i$
    $D'_i \leftarrow$ tail $D'_i$
    if Consistent($a_i : \mathcal{A}$) then
      return $a_i$
  return null
```

Constraint Propagation

- A typical CSP describes the constraints in parts. A search algorithm wades through the constraints looking for an assignment.
- Backtracking happens when a partial assignment that satisfies some constraints cannot be extended to another variable and another constraint.
- **Constraint propagation** or **consistency enforcement** is the endeavour to tighten the CSP so that some kinds of dead ends do not arise.
- This can be done by pruning domains of variables in the simplest case, or by adding constraints to limit the choices to values that can be part of a solution.
- Done to an extreme, consistency enforcement can make search backtrack free.

Arc Consistency

Arc Consistent (AC)

A variable X is said to be arc consistent (AC) with respect to variable Y if there is an edge (X, Y) in the constraint graph and for every value $a \in D_X$, there exists a value $b \in D_Y$ such that $\langle a, b \rangle \in R_{XY}$.

A simple algorithm $\textsc{Revise}((X), Y)$ makes X arc consistent to Y.

Revise

Algorithm Revise prunes the domain of variable X, removing any value that is not paired to a matching value in the domain of variable Y.

```
Revise((X), Y)
  for every $a \in D_X$
    if there is no $b \in D_Y$ s.t. $\langle a, b \rangle \in R_{XY}$
      then delete $a$ from $D_X$
```

The worst-case time complexity of Revise is $O(k^2)$ where k is the size of each domain.

Arc Consistent Constraint Network

Arc Consistent Constraint Network

A constraint network $\mathcal{R}$ is said to be arc consistent if all edges in the constraint graph are arc consistent.

- A node is said to be 2-consistent if an assignment to any variable can be extended to a consistent assignment to any other variable.
- Clearly, if a network is 2-consistent, it must be arc consistent as well.
- A simple brute force algorithm AC-1 cycles through all edges in the constraint graph until no domain changes

AC-1

Algorithm AC-1 cycles through all edges repeatedly even if one value is removed from one variable.

```
AC-1($X, D, C$)
  repeat
    for each edge $(x, y)$ in the constraint graph
      REVISE($(x), y$)
      REVISE($(y), x$)
  until no domain changes in the cycle
```

Time Complexity of AC-1

- Let there be n variables, each with domain of size k.
- Let there be e edges in the constraint graph.
- Every cycle then has complexity $O(ek^2)$.
- In the worst case, the network is not AC, and in every cycle exactly one element in one domain is removed. Then there will be nk cycles.
- The worst case complexity of AC-1 is therefore $O(nek^3)$.

Thank You