AI & NN
Unit I – Part 4

Topics: Constraint satisfaction, related algorithms, handling uncertainty in terms of probability, measure of performance.

Class: III CYS

Sem : 2025-26, Sem 5

-Dr. S.Durga, M.E., PhD

Constraint Satisfaction Problems (CSP)

- **Definition**: A Constraint Satisfaction Problem is a mathematical problem defined as a set of objects whose state must satisfy several constraints and conditions. CSPs can be formalized as a triple (X, D, C) where:
- **X**: A set of variables (e.g., X1, X2, ..., Xn).
- **D**: A set of domains of values for each variable (e.g., D(X1), D(X2), ...).
- **C**: A set of constraints that specify the allowable combinations of values.

- **Example**: Consider a simple CSP of a scheduling problem where you want to schedule classes in a way that no two classes occur at the same time:
- Variables: {Class1, Class2, Class3}
- Domains: {Morning, Afternoon, Evening} for each class.
- Constraints:
  - Class1 ≠ Class2
  - Class1 ≠ Class3
  - Class2 ≠ Class3

CSP: Sample Problem 2

- A cybersecurity team must assign three analysts (A1, A2, A3) to three different systems (S1, S2, S3). Each analyst has different strengths for each system as follows:
  - A1: (S1: High, S2: Low, S3: Medium)
  - A2: (S1: Medium, S2: High, S3: Low)
  - A3: (S1: Low, S2: Medium, S3: High)
- Constraints:
  - A1 cannot work on S2.
  - A2 cannot work on S3.
  - A3 must work on S1.

How to assign analysts to systems respecting all constraints?

Solution:

- Analysts: A1, A2, A3
- Systems: S1, S2, S3

- **Establish Constraints**:
  - A1 ≠ S2
  - A2 ≠ S3
  - A3 = S1
- **Assign A3 to S1** (from the constraint):
  - Current Assignment: A3 -> S1
- **Possible Assignments for A1 and A2**:
  - Remaining Systems: S2, S3
  - Since A1 cannot be assigned to S2, it must be assigned to S3. So, A1 -> S3
- **Final Assignment**:
  - A2 must take the only system left, which is S2. Thus, A2 -> S2
- **Final Resulting Assignment**:
  - A1 -> S3
  - A2 -> S2
  - A3 -> S1

Related Algorithms

- **Backtracking**:
  - A depth-first search algorithm that assigns values to variables one at a time and checks for constraints. If constraints are violated, it backtracks to try a different value.
- **Example**: In our class scheduling example, if Class1 is scheduled for the Morning, and we find that Class2 cannot be in the Morning due to a constraint, we backtrack to try different values for Class 2.

Related Algorithms

- **Constraint Propagation**:
  - Techniques that reduce the search space of a CSP by updating domains of variables based on constraints.
- **Example**: If Class1 is put in the Morning, then the domain of Class2 can be reduced to {Afternoon, Evening} since it cannot be in the Morning.
- The core idea is to eliminate values from the domains of variables that cannot possibly lead to a valid solution, thereby pruning the search tree significantly.
- Common constraint propagation techniques include Forward Checking, Arc Consistency (AC-3 algorithm), and Mac's Algorithm.

Constraint Propagation: Example

- Let's consider a simple CSP involving the problem of assigning colors to a map of three countries (A, B, and C) such that no two adjacent countries have the same color. The two constraints are:
- **Variables**: A, B, C (the countries)
- **Domains**: {Red, Green, Blue} (the colors)
- **Constraints**:
  - $A \neq B$
  - $A \neq C$
  - $B \neq C$

  Solve using **Constraint Propagation**.

- Let (A, B, C) be the variables.
- Each variable has a domain of possible colors: (D(A) = D(B) = D(C) = {Red, Green, Blue}).
- A: {Red, Green, Blue} B: {Red, Green, Blue} C: {Red, Green, Blue}
- **Choose a Variable to Assign**:
  - For instance, suppose we assign a color to (A) first. Let's say we choose **Red** for (A).
- **Apply Constraints (Forward Checking)**:
  - After assigning (A) with **Red**, we check the adjacent variables (B) and (C):
    - For (B): Remove **Red** from (D(B)); now (D(B) = {Green, Blue}).
    - For (C): Remove **Red** from (D(C)); now (D(C) = {Green, Blue}).

  A: {Red} B: {Green, Blue} C: {Green, Blue}
- **Choose Next Variable**:
  - Next, we can assign a value to (B). Let's choose **Green**.
- **Apply Constraints Again**:
  - After assigning **Green** to (B):
    - Remove **Green** from (D(C)); now (D(C) = {Blue}).
    - The state looks like this A: {Red} B: {Green} C: {Blue}

- **Arc Consistency**:
  - A property of CSP that ensures for every value of variable X, there is a value of variable Y that satisfies the binary constraint. Algorithms like AC-3 can be employed to enforce this property.

Handling Uncertainty in Terms of Probability

- **Probabilistic Models**: In AI, uncertainty can be modeled using probabilities. This includes approaches like Bayesian networks, which allow reasoning under uncertainty.
- **Key Concepts**:
  - **Random Variables**: They represent uncertain quantities. E.g., the weather can be a random variable with possible values like "Rainy," "Sunny," etc.
  - **Joint Probability Distribution**: The probability of every possible combination of values for all variables.
- **Example**: If we have a weather-related variable S (Sunny) and activity A (Outdoor), the joint distribution might look like:
  - P(Sunny, Outdoor) = 0.6
  - P(Rainy, Outdoor) = 0.1
  - P(Sunny, Indoor) = 0.3
- **Bayes' Theorem**: Used to update the probability of a hypothesis as more evidence becomes available.

Handling Uncertainty in Terms of Probability: Eg. problem

Problem statement:

- A company experiences a cyber attack that could either be a phishing attack or a malware attack. The historical data shows the following probabilities:
- Probability of a phishing attack (P(P)) = 0.6
- Probability of a malware attack (P(M)) = 0.4
- If a phishing attack occurs, the probability of a successful breach (B) is 0.9. If a malware attack occurs, the probability of a successful breach (B) is 0.7.

What is the overall probability of a successful breach?

- Let (P): Phishing attack
- Let (M): Malware attack
- Let (B): Successful breach
- Given data : $P(B|P) = 0.9$, $P(B|M) = 0.7$, $P(P) = 0.6$, $P(M) = 0.4$
- The total probability of a successful breach can be calculated as: $P(B) = P(B|P)P(P) + P(B|M)P(M)$
- $P(B) = (0.9)(0.6) + (0.7)(0.4)$
- $P(B) = 0.54 + 0.28$
- $P(B) = 0.82$
- The overall probability of a successful breach is 0.82 (or 82%).

Measure of Performance

- **Performance Measures**: In AI, various metrics are used to evaluate the performance of algorithms and systems:
  - **Accuracy**: The proportion of true results among the total cases examined; important in classification tasks.
  - **Time Complexity**: Often measured in Big O notation, indicating how the time to solve the problem grows with input size. Algorithms for CSP can have complexities influenced by the number of variables and constraints.
  - **Space Complexity**: This measures the amount of memory used by the algorithm in relation to the input size.
  - **Quality of Solutions**: Especially important in optimization problems, not just whether a solution exists, but how "good" that solution is concerning the defined objective (e.g., minimum cost, maximum efficiency).

Example Metrics:

- CSP solutions might be evaluated based on how quickly a valid schedule can be generated.
- In a probabilistic model, the measure could include the likelihood of correct predictions based on the model's probabilities.

AI Applications : Example

Face & Object Detection from Footage	You're reviewing CCTV footage from a suspect location. Use AI to identify faces, weapons, or vehicles in video frames or images.
Intelligence Extraction from Text	A cache of text messages, PDFs, and chat logs were found on a seized phone. Use NLP to extract keywords, names, and identify threat indicators.
Image Tamper & Deep Fake Detection	A suspect claims a photo used as evidence is fake. Use AI-assisted tools to check for image manipulation or deepfake characteristics.
Threat Prediction from Logs	You receive server logs from a compromised system. Train a basic AI model to flag log entries likely associated with threats or policy violations.
Anomaly Detection in User Behaviour	An unauthorized login occurred at 02:00 AM. Train/test an anomaly detection model to flag unusual user behavior based on historical access data.