Chapter V - Constraint Satisfaction Problems (CSPs)

5.1 Defining Constraint Satisfaction Problems

A **Constraint Satisfaction Problem (CSP)** is a type of problem in which a solution must satisfy a set of constraints. CSPs are widely used in AI for solving scheduling, planning, and optimization problems.

Components of a CSP:

1. **Variables**: A set of variables X={X1,X2,...,Xn}X = {X_1, X_2, ..., X_n}.
2. **Domains**: Each variable XiX_i has a domain DiD_i of possible values.
3. **Constraints**: A set of rules that specify allowed combinations of values for subsets of variables.

Examples of CSPs:

- **Map Coloring**: Assign colors to regions on a map such that no adjacent regions have the same color.
- **Sudoku**: Assign numbers to a 9×9 grid while satisfying row, column, and box constraints.
- **Scheduling**: Assign time slots to tasks while avoiding conflicts.

Types of Constraints:

- **Unary Constraints**: Apply to a single variable (e.g., "X cannot be 3").
- **Binary Constraints**: Apply to pairs of variables (e.g., "X ≠ Y").
- **Higher-Order Constraints**: Involve more than two variables (e.g., "X, Y, Z must be all different").

Solution to a CSP:

- A solution is an assignment of values to variables such that **all constraints are satisfied**.
- A **partial assignment** is an assignment where only some variables are assigned values.

5.2 Constraint Propagation: Inference in CSPs

Constraint propagation is a technique used to **reduce the search space** by enforcing constraints before searching for a solution.

Common Inference Techniques:

1. **Arc Consistency (AC-3 Algorithm)**
   - Ensures that every value in a variable's domain has at least one valid assignment in related variables.
   - Removes values that cannot satisfy constraints.
   - Example: In Sudoku, if a cell can only contain one number, other cells in the row, column, and box must exclude that number.
2. **Forward Checking**
   - Whenever a variable is assigned a value, forward checking removes inconsistent values from other variables' domains.
   - Improves efficiency by preventing future conflicts early.
3. **Constraint Propagation in Sudoku**
   - If a number is placed in a cell, other cells in the same row, column, and box update their possible values.

5.3 Backtracking Search for CSPs

Backtracking Search is a depth-first search approach where we:

1. Assign a value to a variable.
2. Check if it violates any constraints.
3. If a conflict arises, backtrack and try a different assignment.

Algorithm:

1. Choose an **unassigned variable**.
2. Select a value from its **domain**.
3. Check if it **satisfies constraints**.
4. If valid, assign the value and continue. Otherwise, **backtrack**.

Techniques to Improve Backtracking:

- **Variable Ordering (MRV - Minimum Remaining Values)**: Choose the variable with the fewest legal values first.

- **Value Ordering (Least Constraining Value)**: Choose the value that rules out the fewest options for other variables.
- **Forward Checking**: After assigning a value, remove inconsistent values from other variables' domains.

Example:

- Solving a **map-coloring problem** using backtracking.

5.4 Local Search for CSPs

Unlike backtracking, **local search** methods do not systematically explore all possibilities but rather improve a single candidate solution iteratively.

Common Local Search Techniques:

1. **Min-Conflicts Algorithm**
    - Start with a random assignment.
    - Iteratively change the value of a variable that is causing conflicts.
    - Works well for large problems like scheduling.
2. **Simulated Annealing**
    - Randomly explores different solutions but gradually reduces randomness to converge on an optimal solution.
3. **Genetic Algorithms**
    - Uses evolution-inspired techniques like mutation and crossover to improve solutions over generations.

Example:

- **Solving N-Queens with Min-Conflicts**
    - Start with random queen placements.
    - Move queens to reduce conflicts until a solution is found.

5.5 Problem Structure

The structure of a CSP can greatly affect how efficiently it can be solved.

Types of CSP Structures:

1. **Tree-Structured CSPs**
   - If the constraint graph is a tree, CSPs can be solved in **linear time** using dynamic programming.
2. **Graph-Based CSP Decomposition**
   - Large CSPs can be divided into smaller subproblems, reducing complexity.
3. **Constraint Graph Representation**
   - CSPs can be visualized as a graph where nodes are variables and edges represent constraints.

Example:

- In **timetabling**, some constraints only affect a subset of variables (e.g., students in the same class), which can be exploited to simplify the problem.

Exercises

1. **CSP Representation:** Represent a Sudoku puzzle as a CSP. Define variables, domains, and constraints.
2. **Backtracking:** Implement a backtracking algorithm to solve a simple CSP (e.g., Map Coloring).
3. **Constraint Propagation:** Apply the AC-3 algorithm to simplify a CSP before using backtracking.
4. **Local Search:** Implement the Min-Conflicts algorithm for solving the N-Queens problem.
5. **Problem Structure:** Given a CSP graph, identify independent subproblems and solve them separately.