Constraint Satisfaction Problem (CSP)

Shervin Emrani

1 Introduction

This project is a Python-based solution for solving a *Constraint Satisfaction Problem* (CSP) in the context of assigning groups to halls based on certain constraints and preferences. The problem is solved using different constraint satisfaction techniques such as **Backtracking** and **AC-3 (Arc Consistency Algorithm)**. The project also implements various heuristic techniques like **Minimum Remaining Values (MRV)** and **Least Constraining Value (LCV)** to optimize the search process.

2 Problem Description

The problem involves:

- A set of **groups**, each having preferences for certain halls.
- A set of **halls**, each of which can host a group with specific constraints.
- A list of **constraints** that define which groups should not be assigned to specific halls.

The goal is to assign groups to halls in a way that satisfies all given constraints, using CSP techniques to explore valid configurations efficiently.

3 Input and Output

3.1 Input

The input consists of:

- **halls** (number of available halls)
- **groups** (number of groups to assign)
- A list of **group preferences**, where each group specifies the halls it prefers.
- A list of **constraints**, where each constraint is a tuple indicating that a specific group cannot be assigned to a specific hall.

Example input format:

```
6 3
1 4 6
1 2 3 5 6
3 4 5
5
1 2
2 3
3 4
3 5
3 6
```

3.2 Output

The output consists of the assignment of groups to halls, based on the chosen algorithm (Backtracking or AC-3). The output also includes the time taken to find the solution.

4 Implementation Details

The code is structured into several modules:

- `Hall` - A class representing each hall and its associated constraints.
- `Groups` - A class representing each group and its preferences.
- `Algo` - A class that implements the CSP algorithms and heuristics, including Backtracking, AC-3, MRV, and LCV.

4.1 Main Algorithm Flow

The main script handles the input, sets up the problem, and then solves it using the following steps:

1. Parse the input to get the number of halls, groups, preferences, and constraints.
2. Create hall objects and group objects. Each group is initialized with its preferences, and each hall is initialized with the groups that are constrained.
3. The most constrained hall is selected using the **MRV (Minimum Remaining Values)** heuristic.
4. The least constraining values for that hall are determined using the **LCV (Least Constraining Value)** heuristic.
5. The user is prompted to select between **Backtracking** and **AC-3** algorithms.
6. Based on the user's choice:
    - If Backtracking is selected, the algorithm runs with forward checking to prune inconsistent values.
    - If AC-3 is selected, arc consistency is established first, and then backtracking is performed to find the solution.
7. Finally, the solution is printed along with the time taken.