AI Project 2: Constraint Satisfaction Problem for Graph Coloring using Backtracking Search with MRV and LCV Heuristics

Introduction:

The graph coloring problem is a classical combinatorial problem, where the objective is to assign colors to the vertices of a graph so that no two adjacent vertices share the same color. This Java program is designed to solve the graph coloring problem using a backtracking search with the Minimum Remaining Values (MRV) and Least Constraining Value (LCV) heuristics. The program aims to find a good graph coloring using the number of colors the user gives.

The backtracking search algorithm is a depth-first search algorithm that constructs partial solutions step-by-step and abandons a partial explanation when it determines that the partial solution cannot be extended to a complete solution.

It is a widely used approach for solving constraint satisfaction problems (CSPs), including the graph coloring problem.

The program utilizes two heuristics to improve the efficiency of the backtracking search:

Minimum Remaining Values (MRV) heuristic: This heuristic selects the next variable (vertex) to be assigned a color based on the smallest remaining domain size (the number of available colors). By choosing the vertex with the fewest remaining colors, the algorithm focuses on the most constrained vertices first, which helps to detect failures earlier and prune the search space more effectively.

Least Constraining Value (LCV) heuristic orders the domain values (colors) of the selected vertex based on the number of constraints they impose on the neighboring vertices. The algorithm increases the likelihood of finding a valid coloring without backtracking by choosing the color that sets the fewest constraints.

The program reads the graph information and the number of colors from an input file, constructs the adjacency list representation of the graph, and initializes the domain list for all vertices. Then, it performs the backtracking search with MRV and LCV heuristics to find a valid coloring of the graph. Finally, it prints the vertex-color assignments if a solution is found or "No solution found!" is displayed if no solution exists.

Program Structure:

The program is composed of the following classes:

1. AIProjectCSP:

This is the main class responsible for executing the program. It performs the following tasks:

- Checks if the input file is provided and valid.
- Reads the input file using the ReadFile.read() method, which returns a GraphInput object containing the adjacency list and the color count.
- Initializes a BTSearch object with the GraphInput object.
- Executes the search algorithm to find a solution.
- Prints the vertex-color assignments if a solution is found or "No solution found!" if no solution exists.

2. ConsistencyCheck:

This class stores the results of a consistency check performed during the backtracking search. It has two main attributes:

- removedValues: A HashMap containing the values removed from the domains of vertices during the consistency check.
- isConsistent: A boolean flag indicating whether the current assignment is consistent with the constraints.

The class provides getter methods to access the removedValues and checkConsistency attributes.

3. BTSearch:

This class implements the backtracking search algorithm with MRV and LCV heuristics. It contains the following attributes:

- adjacencyList: A HashMap representing the graph's adjacency list.
- domainList: A HashMap representing the domains of vertices (possible colors for each vertex).
- colorCount: An integer representing the number of colors available for graph coloring.

The class contains the following methods:

- initializeDomain(): Initializes the domain list for all vertices with all available colors.
- executeSearch(): Starts the backtracking search process and returns a HashMap containing the vertex-color assignments if a solution is found or an empty HashMap if no solution exists.
- search(): The recursive backtracking search method that performs variable selection, value ordering, and consistency checking.
- selectVariable(): Implements the MRV heuristic to select the next variable (vertex) to be assigned a color.
- orderDomain(): Implements the LCV heuristic to order the domain values of a selected variable.

- checkConsistency(): Performs a consistency check to ensure the current assignment does not violate any constraints.
- update(): Updates the domains of adjacent vertices based on the selected variable's color assignment.

4. GraphInput:

This class represents the input graph and contains the following attributes:

- adjacencyList: A HashMap representing the graph's adjacency list.
- colorCount: An integer representing the number of colors available for graph coloring.

The class provides getter methods to access the adjacencyList and colorCount attributes.

5. ReadFile:

This class uses a static method to read the input file and parse the adjacency list and color count. The method performs the following tasks:

- Reads the input file line by line.
- Skips empty lines or lines starting with "#".
- Parses the color count from lines starting with "c" or "C".
- Parses the edge information (vertex pairs) from other lines.
- Constructs and returns a GraphInput object containing the adjacency list and color count.

6. StoreSet:

This is a utility class used to store a pair of integers. It has two attributes:

- first: An integer representing the first value in the pair.
- second: An integer representing the second value in the pair.

The class provides getter methods to access the first and second attributes.

By combining these classes, the program effectively solves the graph coloring problem using backtracking search and MRV and LCV heuristics, leading to an efficient and effective solution.

Working:

1. Initialization:
   The graph is an adjacency list, a HashMap with vertex identifiers as keys, and an ArrayList of adjacent vertices as values. The program begins by reading an input file containing the graph's adjacency list and the number of colors to color the graph. The ReadFile.read() method parses the input file and creates a GraphInput object.

2. Backtracking Search (BTSearch):
   The backtracking search algorithm is implemented in the BTSearch class. The search starts by initializing the domain list, a HashMap where the key is the vertex identifier

and the value is a HashSet of possible colors. Initially, the domain for each vertex contains all available colors.

3. Variable Selection (MRV):
   The program selects the next variable (vertex) to be assigned a color using the MRV heuristic. The MRV heuristic chooses the variable with the fewest legal values remaining in its domain. If multiple variables have the same minimum domain size, the algorithm selects the one with the highest degree (i.e., the largest number of adjacent vertices) as a tiebreaker.

4. Value Ordering (LCV):
   The program orders the domain values for the selected variable using the LCV heuristic. The LCV heuristic orders the values by the number of restrictions they impose on neighboring variables, preferring values that impose the fewest restrictions. In graph coloring, LCV counts the number of neighbors with the same color in their domain and orders colors by increasing counts.
5. Consistency Check:
   After selecting a variable and ordering its domain values, the program performs a consistency check to ensure the assignment does not violate any constraints. The consistency check involves updating the domains of adjacent vertices based on the selected variable's color assignment. If the assignment is consistent, the search proceeds to the next variable; otherwise, it backtracks to the previous variable.

6. Recursion:
   The backtracking search algorithm is recursive, assigning colors to vertices and checking consistency at each step. The search returns a solution if the assignment is consistent and all vertices have been given a color. If it fails, it backtracks to the previous vertex and tries a color.

7. Solution:
   Once the search is complete, the program prints the vertex-color assignments if a solution is found or prints "No solution found!" if no solution exists.

Instructions to Run:

To run the AIProjectCSP Java program, follow these steps:

1. Save the Java code in a file named ‘AIProjectCSP.java’
2. Open a terminal or command prompt and navigate to the directory where the AIProjectCSP.java file is saved.
3. Compile the Java program using the following command: ‘***javac AIProjectCSP.java***’
   This will generate different class files.
4. Run the compiled Java program with the input file as an argument: ‘***java AIProjectCSP input.txt***’

5. Replace input.txt with the name of your input file containing the graph information and the number of colors.
6. The program will execute and display the vertex-color assignments if a solution is found or "No solution found!" if no solution exists.