ITCS 6150 – Intelligent Systems
Fall 2021
A Project report
On
"Constraint satisfaction problems (CSP) - Map Coloring"

Team details

Mohith Raju
mraju2@uncc.edu
801254279

Nehal Kathale
nkathale@uncc.edu
801205316

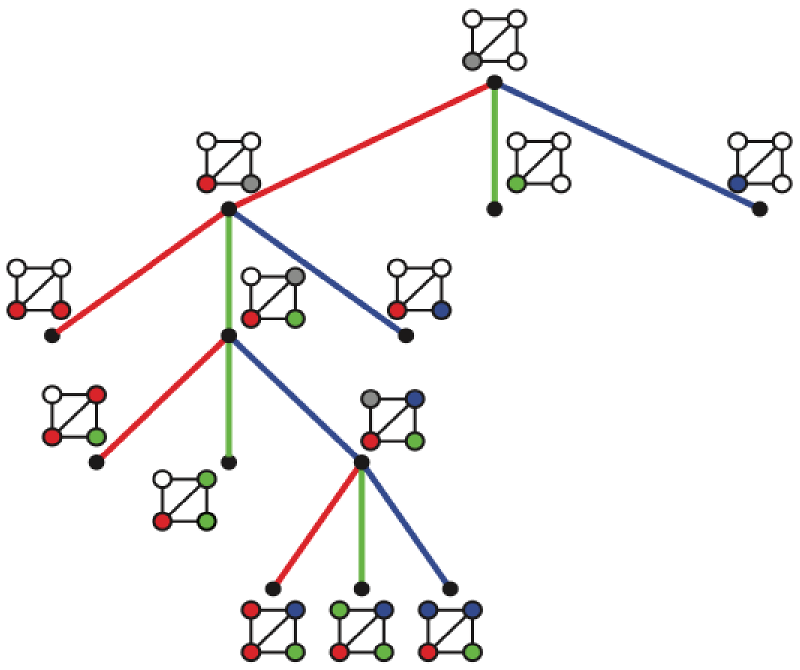
Renu Karule
rkarule@uncc.edu
801261337

Constraints Satisfaction Problems:

A Constraint Satisfaction Problem consists of a collection of variables, a domain of values and a set of constraints. The objective of the problem is to assign values to each variable such that all constraints are satisfied. In other words, CSPs represent a homogeneous set of constraints over a collection of variables which is solved by constraint satisfaction methods. More formally, a constraint satisfaction problem can be defined as by the set {V, D, C}, where

- { V1, V2, V3,... } is a set of variables,
- { D1, D2, D3,... } is a set of their respective domains of values, and
- { C1, C2, C3,... } is a set of constraints.

Example:

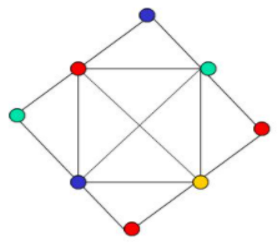


Citation: https://quantum-journal.org/wp-content/uploads/2019/07/AltBackTrack-1.png

In the above example, The set of variables are the vertices to color, the domain values are the set of possible solutions that are generated at each level of the tree and the constraints for this problem are that no two adjacent vertices must have the same color. The best possible solution for this example is at depth 4, after branching out the green edge in which no two vertices have the same color.

Chromatic Number:

Chromatic number in a CSP graph is the minimum number of colors required to color the vertices of the graph such that no two adjacent vertices have the same color.
For example,



This graph has a chromatic number = 4 as we required 4 colors (Red, Green, Blue, Yellow) to color all the vertices.

Backtracking:

Backtracking is a technique that uses an algorithmic way to solve the problem at hand using a recursive approach. It uses recursive calling to find the solution by building the solution step by step in an incremental manner and then removing the solutions if they do not follow the constraints of the problem.

For CSPs, the problem is solved by expanding from the initial node and creating new solutions at each iteration based on the previous nodes. Once we reach a point in the tree which does not lead to the solution of the problem, we backtrack and expand from the previous node looking for the solution. This way, we do not perform redundant searches while looking for a solution.

Strategies to solve the CSP - Map coloring:

- Depth First Search.
- Depth First Search + Forward Checking.
- Depth First Search + Forward Checking + Propagation through singleton domains.

Depth First Search:

The DFS is similar to the depth first search of a tree. Since we are dealing with graphs, We might run into infinite loops if we encounter an already discovered node. To avoid this, we attach a flag value to each node so that we do not branch to an already discovered node.

Depth First Search + Forward Checking:

DFS with forward checking is an extension of the backtracking search that uses the look-ahead approach. At each iteration, we keep track of the remaining legal values for unassigned variables and terminate search if no legal values exist for the variables. A variable is considered illegal if it does not follow the constraints specified.

Depth First Search + Forward Checking + Propagation through singleton domains:

In this approach, in addition to DFS with Forward Checking, we use Propagation through singleton domains in which we wipe out alternatives from a neighbour in the previous step such that it has only one remaining choice. Now, this neighbour is added to the rundown of factors to expand and expand everyone of the variables in the list.

To expand a singleton variable, observe its outstanding choice and think about everyone of its neighbours and check off the values that are contradictory to the one residual choice. After checking off choices to a neighbouring variable and that variable has just a single choice left, add that neighbour to the single choice of factors to expand.

Heuristics Used:

Minimum remaining values (MRV):
The MRV heuristic always selects the variable with the lowest number of legal values.

Least Constraining Values Heuristic(LCV):
The Least Constraining Values Heuristic chooses a variable with the value that rules out the fewest number of values in the remaining variables.

Degree Heuristic:
The Degree heuristic selects the variable with the highest degree. i.e. a variable that is involved with the largest number of constraints on other unassigned variables.

Program Design and Implementation:

ExecteConstrainSatisfactionProblem.java:
This is the main function of the program. i.e. it controls the program flow. It executes the map coloring operations based on the given requirements.

OutputSummary.java:
This function performs the backtracking method in milliseconds. It also stores and provides information about the backtracking performed and also provides the average time required to calculate the output.

MapOfAustralia.java:
This method is used to initialize the different states that are present in the map of Australia. There are three possible colors that can be assigned to these states. The method addNewRule is used to determine which two neighbouring states cannot have the same color.

MapOfAmerica.java:
This method is used to initialize the different states that are present in the map of America. There are four possible colors that can be assigned to these states. The method addNewRule is used to determine which two neighbouring states cannot have the same color.

Backtracking.java:
This method is used to perform backtracking with the heuristic function specified. It is a recursive function that ensures that no two adjacent states have the same color.

BacktrackingWithHeuristic.java:
The methods used in this file are,

1. useDEGStrategy: It performs map colouring using the degree heuristic evaluation method.
2. useLCVStrategy: It performs map colouring using the LCV heuristic evaluation method.
3. userMRVStrategy: It performs map colouring using the MRV heuristic evaluation method.
4. checkForwardAndProceed: It performs map colouring using forward checking strategy.

Calculation.java:
This class has setters and getters for literal list values and formatting for output summary.

CSP.java:
This class has setter and getter methods for rule, scope and network constraints.

CSPIterator.java:
This is a custom iterator class.

Literal.java:
This class represents literal properties and it’s setters and getters.

Rule.java:

Scope.java:
It’s a utility class for iterating object arrays.

ScopeData.java:
This class handles modification of literal values.

Tuple.java:
It is an utility class for handling tuples.

UnequalityRule.java:
This class handles constraints checks.

Source:
https://drive.google.com/drive/folders/1FXYU4yXQd7zTRKS4bvN2EOTskSlaz3Qy?usp=sharing

Result :

1. Australia

   Chromatic Number : 3

   Total number of iterations: 4

Without Heuristics :

	Number of Backtracks	**Execution Time (ms)**
DFS	20	1.0
DFS + Forward Checking	20	0.0
DFS + Forward Checking + Singleton domains	12	0.0

With Heuristics :

LCV	20	0.0
LCV + Forward Checking	0	0.0
LCV + Forward Checking + Singleton domains	0	0.0
Degree	8	0.0
Degree + Forward Checking	0	0.0
Degree + Forward Checking + Singleton domains	0	0.6
MRV	1.2328615E7	15756.0
MRV + Forward Checking	0	1.0
MRV + Forward Checking + Singleton domains	0	0.0

2. America

   Chromatic Number: 4

   Total number of iterations: 4

Without Heuristics:

	Number of Backtracks	**Execution Time (ms)**
DFS	1.2328615E7	9036.0
DFS + Forward Checking	1.2328615E7	9802.0
DFS + Forward Checking + Singleton domains	7397169.0	5677.8

With Heuristics:

LCV	12328615.00	12273.00
LCV + Forward Checking	0	1.0
LCV + Forward Checking + Singleton domains	0	0.0
Degree	66	2.0
Degree + Forward Checking	0	1.0
Degree + Forward Checking + Singleton domains	0	0.60
MRV	12328615	13977
MRV + Forward Checking	0	0.0
MRV + Forward Checking + Singleton domains	0	0.0

Summary:

The Constraint Satisfaction Problem problem was successfully implemented using

1. Depth-first search only
2. Depth-first search + forward checking
3. Depth-first search + forward checking + propagation through singleton domains

The number of backtracks and the execution times were also illustrated for each of the heuristics used (LCV, MRV, Degree) which helps us to better understand the operation of map coloring performed.