

**Graph Coloring Problem Using**

**Backtracking Algorithm**

Submitted by:

Abeda Sultana Khanam

I.D: 2016-1-60-011

Shakil Mahmud

I.D: 2016-1-60-007

Course code: CSE365

Course title: Artificial Intelligence

Section: 03

Date: 12 December 2018

**Abstract**

Backtracking algorithm is capable of deciding whether a decision problem has a solution or not through a sequence of trials and errors. Here, we present a demonstration of backtracking algorithm by coloring some random graphs. Some current limitations and perspective are also present in here.

**Introduction**

Graph Coloring is an assignment of colors to the vertices of a graph such that no two adjacent vertices have the same color that too by using the minimum number of colors. The smallest number of colors needed to color a graph G is called its chromatic number, and is often denoted χ(G). Unfortunately, there is no efficient algorithm available for coloring a graph with minimum number of colors but there are approximate algorithms to solve the problem such as the Backtracking Algorithm.

**Problem Description**

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time. The idea is to assign colors one by one to different vertices, starting from the vertex 0. Before assigning a color, we check for safety by considering already assigned colors to the adjacent vertices. If we find a color assignment which is safe, we mark the color assignment as part of solution. If we do not a find color due to clashes then we backtrack and return false.

**Algorithm**

**function** BACKTRACKING-SEARCH(*csp*) **returns** a solution, or failure

**return** BACKTRACK({}, *csp*)

**function** BACKTRACK (*assignment, csp*) **returns** a solution, or failture

**if** *assignment* is complete **then return** *assignment*

*var* := SELECT-UNASSIGNED-VARIABLE(*csp*)

**for each** *value* **in** ORDER-DOMAIN-VALUES(*var, assignment, csp*) **do**

**if** *value* is consistent with *assignment* **then**

add { *var = value* } to *assignment*

*inferences* := INFERENCE(*csp, var, value*)

**if** *inferences ≠ failure* **then**

add *inferences* to *assignment*

*result* := BACKTRACK(*assignment, csp*)

**if** *result* ≠ *failure* **then**

**return** *result*

remove { *var* = *value* } and *inferences* from *assignment*

**return** *failure*

Implementation Details

Data structure used: Hashmap and Hashset used. In hashmap vertexes node number is key and value of hasmap is a hashset which contains all adjacent nodes of that node. ArrayList, Array(fixed memory size) also used in some cases.

Graphical User Interface: GUI is built by using java swing framework. Some classes, sub-classes of Java AWT library is used for drawing the graph.

**Conclusion and future work:**

This is just a one way to solve graph coloring problem by backtracking algorithm,  
 and it has advantages in terms of search space with respect to some other  
algorithm like bfs, dfs, A\*.

.  
This project is a small version. In future, it can be modified. Some others algorithm  
 can be added to this and it is possible to simulate them graphically through this  
 project and adding animation is also possible.