Solving a Sudoku by Graph Coloring Algorithm

by

Sai Dev Prakash Janapareddi (11507630) , Akshay Kumar Reddy Gillella(11527403)

*[] – Represent the references

Objective: -

The main objective of this project is to solve sudoku. There are many ways to solve sudoku and one of them is to use Graph Coloring using the backtracking method algorithm. As we know there is only one solution to a given sudoku problem but to achieve it we may have to try out many ways. So to minimize the number of ways and to achieve the optimal solution with less run time we are using this algorithm.

Applications of the Algorithm: -

The algorithm we are using can be seen as two, one is the "Graph Coloring" algorithm and the other is the "Backtracking" algorithm.

Graph Coloring: -

When coming to their applications first with the Graph Coloring algorithm there are many methods of coloring like 'Acyclic vertex coloring', 'Circular vertex coloring', 'Star vertex coloring', and a few special colorings like 'Circular edge coloring'. These are important, interesting and most importantly are the developing branch of graph theory. Because we can construct algorithms with new properties, proofs, and conjectures and can be explored by mathematics by formulating formulas by all the computer scientists all over the world. More about its methods, applications and types can be seen here **[1]**.

When coming to our project I am using a circular vertex coloring method (see Fig 9). So it assigns colors to vertices (See Fig 1) and this algorithm makes sure that no two adjacent nodes have the same color and we have made a sure number of ways graph can be colored, checking if the graph can be colored with the given set of colors which is called m – Coloring Decision (for Fig 1 the chromatic number is '3') and finding out the minimum colors (for Fig 1 it is '3') required to color a graph i.e. m – Coloring Optimization. More about it can be seen in [2]

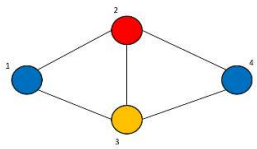

Fig 1: Colored Graph

Back Tracking: -

The backtracking algorithm uses a method to find a solution incrementally, i.e. by trying out recursively many ways to solve a problem. It helps to remove all the failure solutions that do

not satisfy the given constraints of a particular problem. It helps to solve some of the famous problems like the “N Queen Problem”. More about it can be seen in [3].

When coming to our project we fill the positions that are empty in the suduko one by one and whenever we find out that a particular current digit does not take us anywhere or does not satisfy the given constraints of a problem. Then we backtrack or remove the digit and try out with any new digit, so this mainly helps us to decrease the permutations and hence the run time. A perfect visualization is shown in Fig 2(Please click on it).

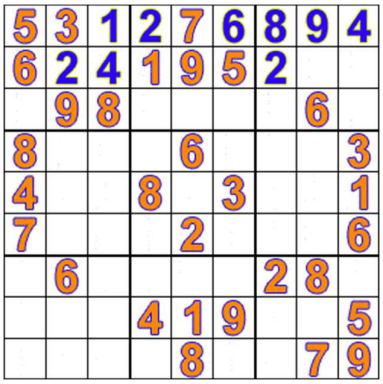

Fig 2: Back Tracking in Sudoku

Related Works: -

1) One of the Research papers **[4]** shows to solve sudoku using a minigrid based backtracing method. Here it divides the sudoku puzzle from (9 x 9) to (3 x 3) mini-grids and each of those grids are labeled from 1 to 9.

After that based on the clues, it tries to find out the valid permutations and the number of clues given depicts it difficulty. Then it uses the backtracking method to solve the mini-grid and finally find the solution to the sudoku puzzle, i.e. if it has a solution.

2) As per the research paper **[5]**, it also divides the (9 x 9) into (3 x 3) subgrids. Each cell that is in the grid is depicted as the node and we link that node to each of the other nodes in the same grid and as well as in its respective cell grids, i.e, in its same rows and columns(see Fig:). Now since it has few of the cells filled it uses the Douglas–Rachford algorithm to solve the sudoku. It solves the predicting the colors of the unfilled numbers using the prefilled numbers by considering them as colors.

Experiments and Output Results: -

Exp1: -

In the first experiment, we did understand the graph coloring algorithm by using pen and paper by assigning colors to nodes. So we made the dataset as below diagram (See Fig 3)

Here we have 5 Nodes and 7 edges {(0 1), (0 2), (1 2), (1 3), (1 4), (2 3), (3 4)}

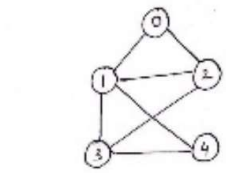

Fig 3: Pen and Paper Graph Coloring

Exp1 Result: -

We can color them as below and understood how to implement graph coloring algorithm.

Red → Node 0,3

Green → Node 1

Blue →Node 2,4

Exp2: -

As we understood the above coloring, we implemented it in a code that used greedy algorithm and generates a LaTex(TikZ) of the colored graph. Below is the data set.

Dataset: -

Vertices → {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19}

Edges →

{(0,3),(1,2),(1,5),(3,7),(1,6),(2,4),(4,5),(10,12),(19,3),(14,9),(7,8),(15,16),(13,14),(18,9),

(11,3),(10,19),(8,11),(0,9),(13,17),(18,6),(16,17),(15,8) }

Exp2 Result: -

Red → 0, 1, 4, 7, 10, 11, 13, 15, 18

Yellow → 2, 3, 5, 6, 8, 9, 12, 16

Blue → 14, 17, 19


graphColored.tex.txt

Output document →

But we were note able to use the Latex(TikZ) code to solve the sudoku. So we retained the code which gives the number of unique colors needed for a given graph having vertices and edges.

Exp3: -

To find out the minimum number of colors required for any sudoku and to verify does our algorithm gives the minimum color for any sudoku puzzle. Basically, we have to color the below Fig 4.

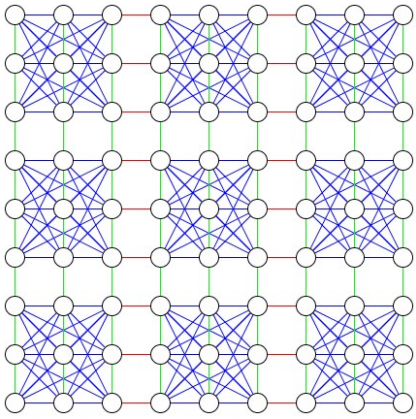

Fig 4: Uncolored Sudoku

Dataset: -

We used below two sudoku puzzles, one is easy (Fig 5) and the other is a moderate puzzle(Fig 6)

```
Input un-solved sudoku
1 . . 4 . . 7 . .
. 4 . 8 . . 2 . .
. . 7 2 . 1 6 . .
. 7 4 . . 8 . . 3
3 . . . 7 . 4 . .
5 . 1 . . 3 8 . 7
. . 6 . 2 . . . 8
. . . 3 6 4 . . 2
7 . . 1 . . 5 . .
```

Fig 5: Easy Sudoku Puzzle

```
Input un-solved sudoku
. . . 4 8 . 2 . 9
. . . . . 7 . . 5 1
. 8 3 . 2 . . . .
. . 4 . . . . . .
7 6 . . . . . . 2
. 5 . 7 . 9 . . .
. . 7 . . 5 9 . 4
. . . . . . 5 . .
4 . . 8 . . 6 7 .
```

Fig 6: Moderate Sudoku Puzzle

Exp3 Result: -

For both the above puzzles we got the number of colors required is **9** (Fig 7)**,** hence we achieved the algorithm which does colors with only 9 colors(which is the minimum number).

```
Number of colors: 9
```

Fig 7: Number of colors used in Sudoku

Analysis: -

To visualize the colored graph of Fig 5(Easy Sudoku) as shown in below Fig 8.

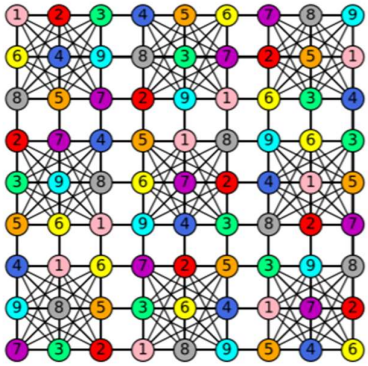

Fig 8: Graph colored solution for Easy Sudoku Puzzle

So over here as discussed above(Point 2 of Related Works) or from **[5]** research paper. Each node is connected to all the nodes in its (3 x 3) grid and also to all the nodes in the same rows and columns, so it should not be colored the same as these mentioned nodes. So the minimum number of colors required is 9 which we achieved through our algorithm.It can be easily viewed from below video or from [6s]

image4.mp4

Fig 9: Video Visualization of nodes connecting to other nodes

Exp 4:-

Solving an easy sudoku puzzle

1. Verify the output solved puzzle
2. Number of backtracks happened
3. Number of colors used.
4. Run Time.

Dataset:-

Easy Suduko puzzle (Fig 9)

```
Input un-solved sudoku
. 7 . . 2 . . 4 6
. 6 . . . . 8 9 .
2 . . 8 . . 7 1 5
. 8 4 . 9 7 . . .
7 1 . . . . . 5 9
. . . 1 3 . 4 8 .
6 9 7 . . 2 . . 8
. 5 8 . . . . 6 .
4 3 . . 8 . . 7 .
```

S

Fig 9: Easy Sudoku Puzzle

Exp4 Result:

All 4 questions are answered from our code output(Fig 10).

```
Solved Sudoku:
8 7 5 9 2 1 3 4 6
3 6 1 7 5 4 8 9 2
2 4 9 8 6 3 7 1 5
5 8 4 6 9 7 1 2 3
7 1 3 2 4 8 6 5 9
9 2 6 1 3 5 4 8 7
6 9 7 4 1 2 5 3 8
1 5 8 3 7 9 2 6 4
4 3 2 5 8 6 9 7 1

Number of backtracks: 1113

Number of colors: 9

Run time:- 0.024 sec
```

Fig 10: Solved Easy Sudoku

Exp 5:-

Solving an Moderate sudoku puzzle

1. Verify the output solved puzzle
2. Number of backtracks happened
3. Number of colors used.
4. Run Time.

Dataset:-

Easy Suduko puzzle (Fig 11)

```
Input un-solved sudoku
1 3 . 5 . 6 4 . 9
. . . . . . . . 2
. . 4 . 1 2 5 . .
. . 7 8 . . . 1 .
9 . 5 . 4 3 . 8 .
. 8 . 7 . . 2 . .
2 . . 3 . . . 9 .
. . . . 5 8 . . 6
. . . . 2 . . . 5
```

Fig 11: Moderate Sudoku Puzzle

Exp5 Result:

All 4 questions are answered from our code output (Fig 12).

```
Solved Sudoku:
1 3 2 5 8 6 4 7 9
5 9 8 4 3 7 1 6 2
7 6 4 9 1 2 5 3 8
4 2 7 8 6 5 9 1 3
9 1 5 2 4 3 6 8 7
6 8 3 7 9 1 2 5 4
2 5 6 3 7 4 8 9 1
3 4 9 1 5 8 7 2 6
8 7 1 6 2 9 3 4 5

Number of backtracks: 283

Number of colors: 9

Run time:- 0.024 sec
```

Fig 12: Solved Moderate Sudoku

Exp 6:-

Solving an Difficult sudoku puzzle

1. Verify the output solved puzzle
2. Number of backtracks happened
3. Number of colors used.
4. Run Time.

Dataset:-

Easy Suduko puzzle (Fig 13)

```
Input un-solved sudoku
. . . 2 . 9 . . .
1 . 4 . . . . . 5
. . 2 . . 5 . . 8
3 . 8 . . . . 9 .
. 2 . 7 . 8 . 4 .
. 6 . . . . 8 . 3
4 . . 5 . . 6 . .
8 . . . . . 3 . 9
. . . 1 . 3 . . .
```

Fig 13: Difficult Sudoku Puzzle

Exp6 Result:

All 4 questions are answered from our code output (Fig 14).

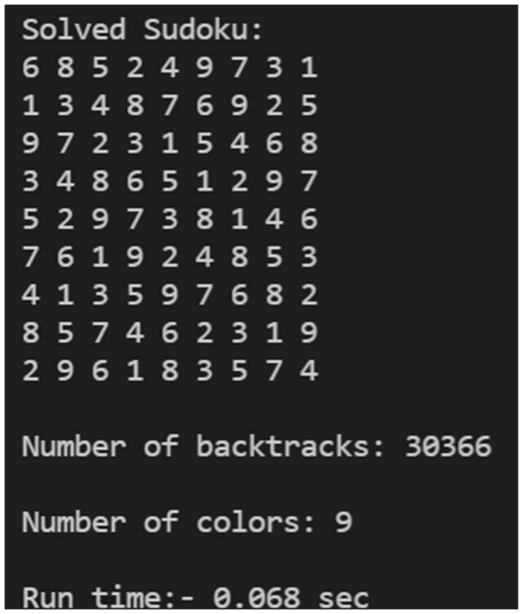

Fig 14: Solved Difficult Sudoku

Conclusion: -

We are able to solve any sudoku puzzle (Easy, Moderate, Hard) and we achieved all the solutions using only 9 colors. We were are show the number of backtracks for a given puzzle respective to their difficulty.

We observed that the run time for easy and moderate puzzles is same **(~ 0.024 sec)** but the difficult puzzle took a slightly higher time **(~ 0.068 sec)** and has the highest backtracking's in it.s

References: -

[1] https://cyberleninka.org/article/n/5s00179.pdf

[2] https://medium.com/code-science/sudoku-solver-graph-coloring-8f1b4df47072

[3] https://www.researchgate.net/profile/Martin-Ward-6/publication/257253260_A_Multipurpose_Backtracking_Algorithm/links/59f08b56aca272cdc7ca80bd/A-Multipurpose-Backtracking-Algorithm.pdf

[4] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6779291

[5] https://www.researchgate.net/profile/Ruben-Campoy/publication/311668725_Solving_Graph_Coloring_Problems_with_the_Douglas-Rachford_Algorithm/links/5853aac708ae95fd8e1defe0/Solving-Graph-Coloring-Problems-with-the-Douglas-Rachford-Algorithm.pdf

[6] https://preview.redd.it/tqfs8zpxlzfz.gif?format=mp4&s=4dd707b937cfc61f20864f89f7a6e7caa4e6c0f0s