Applications of Graph Coloring

Kausthub Thekke Madathil - 191IT125
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: kausthubtm.191it125@nitk.edu.in

Puttaraja - 191IT139
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: puttaraja.191it139@nitk.edu.in

Sohanraj R - 191IT149
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: sohanrajr.191it149@nitk.edu.in

Neeraj Mirji - 191IT232
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: neerajmirji.191it232@nitk.edu.in

I. Introduction

1) What is a graph ?: Graph is basically a collection of vertices and edges which connects these vertices. In graph we refer the these vertices as nodes. Graph is denoted by G, its vertices are denoted by V and its edges are denoted by E. An example of a graph is seen in fig.2 . The fig.2 can be considered as a graph since it consists of nodes which are connected by edges.
The set of vertices : V = { A , D , B , E , F , C } The set of edges : E = { AD , AE , AB , BC , CF , CE , DE , EF }

G = (V , E)

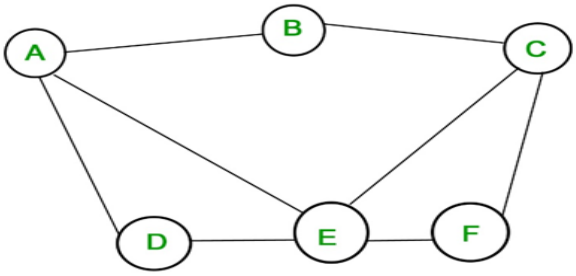


Fig. 1: An example of a graph

2) What is graph coloring?: A graph coloring is an assignment of labels, called colors, to the vertices of a graph such that no two adjacent vertices share the same color or subjected to certain constraints. The smallest number of colors required to color a graph G is called its chromatic number of that graph. Graph coloring problem is a NP Complete problem. A k-coloring of a graph G is a vertex coloring that is an assignment of one of k possible colors to each vertex of G (i.e., a vertex coloring) such that no two adjacent vertices receive the same color.

3) What is Backtracking Algorithm ?: 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 . Therefore if the current solution is not suitable, then backtrack and try other solutions. A backtracking algorithm is a problem-solving algorithm that uses a brute force approach for finding the desired output. The Fig.3 shows the standard algorithm

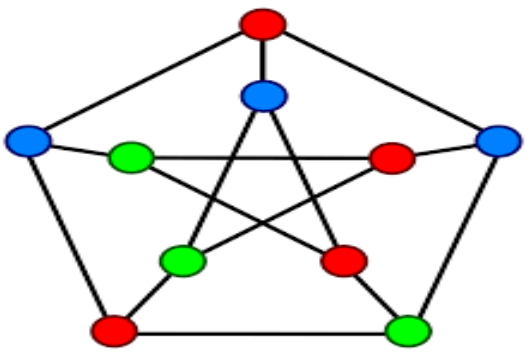


Fig. 2: Graph coloring

```
Backtrack(x)
    if x is not a solution
        return false
    if x is a new solution
        add to list of solutions
    backtrack(expand x)
```

Fig. 3: Backtracking Algorithm

A. Wedding seating problem

The Wedding Seating Problem (WSP) involves taking a set of wedding guests and assigning them to tables so that the following constraints are met:

- Guests belonging to groups, such as couples and families, should be sat at the same tables.
- The number of guests per table should be equal
- If there is any perceived animosity between different guests, these should be sat on different tables; and similarly
- If guests are known to like one another, they should be sat at the same table.

The WSP can be formally stated as type of graph partitioning problem. Specifically, we are given a graph G = (V, E) comprising a vertex set V and an edge set E. Each vertex v belongs to V is used to represent a group of guests who are required to sit together (couples, families, etc.).

B. Sudoku using graph coloring

Sudoku is a logic-based, combinatorial number-placement puzzle. In classic sudoku, the objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called ”boxes”, ”blocks”, or ”regions”) contain all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. The main of aim is to place all the numbers from 1 – 9 into the vacant cells in such a way that , that in every row , every column and every region (i.e a 3 x 3 block) each number appears only once. Initially the player is given with some of the numbers filled in the sudoku. He needs to fill the rest of the vacant places in the sudoku following the above rule.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(a) Unsolved sudoku

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(b) Solved sudoku

Fig. 4: Sudoku

II. Methodology

A. Wedding seating problem

The WSP can be formally stated as type of graph partitioning problem. Specifically, we are given a graph G = (V, E) comprising a vertex set V and an edge set E. Each vertex v belongs to V is used to represent a group of guests who are required to sit together (couples, families, etc.). The vertices can be seen as chairs or the guests. The vertices can be partitioned into ’k’ subsets U = {U1, . . . , Uk} where ’k’ is the requested number of tables which is defined by the user, with each subset Ui defining the guests assigned to a particular table i. Each table can be assigned one color and all the chairs or ’guests’ (vertices) should be of the same color in a table. The aim is to achieve a k coloring of G0 = (V, E0) for which no pair of adjacent vertices is assigned the same color.

The main idea is that if two guests have issues sitting with each other, then there is an edge between those two nodes. Thus while assigning colors to the nodes, the connected nodes recieve different colors which in this case relates to guests with conflicts not getting the same table.

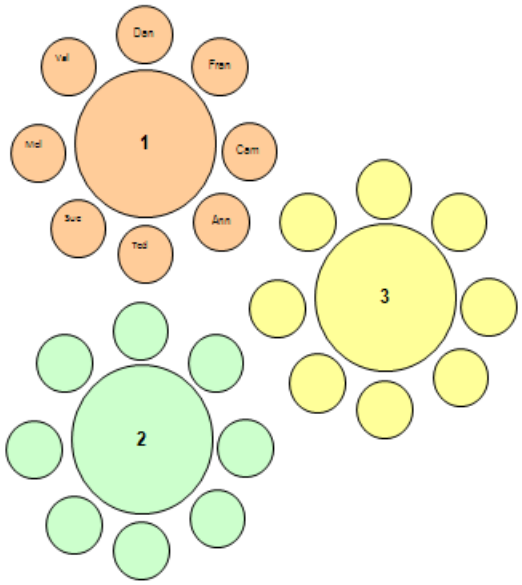


Fig. 5: Graph coloring on WSP

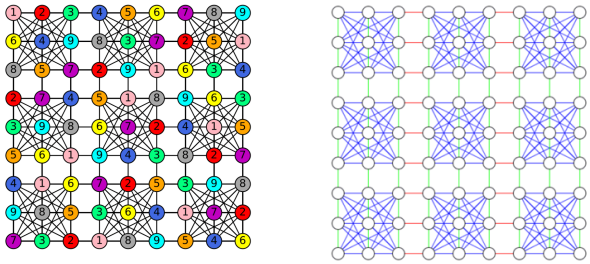
1) Algorithm:

- Input the no of vertices (guests), no of table (k) and no of chairs per table (m).
- Input the no of vertices (guests), no of table (k) and no of chairs per table (m).
- Creating the edges of the graph by user inputs on likes and dislikes of the user.
- Create a recursive function to assign colors to the graph which can be done using a couple of helper functions to check whether it is safe to color and checks whether it follows all requirements or not
- If the function returns ’True’ then assign the color and if it returns ’False’ backtrack.
- call the recursive function continuously till all the vertices are filled.
- Try assigning the vertex with the next possible color and move forward in the same way.
- The recursive function should return true if there is a possible way to color all the vertices else it should return false.

B. Sudoku

The main reason behind sloving a sudoku using graph coloring is the resemblance of cells present in the sudoku to the nodes present in the graph. So here we consider all the cells as the nodes in the graph and all the cells in the given row , column and the 3 x 3 grid as the neighbours of that particular cell i.e the neighbouring cells are directly connected to the considered cell using a edge. The main idea is to assign colors to all the different vertices one by one starting from the vertex 0 and before we assign a given color to a vertex we must make sure that none of the nodes that are directly connected to the given node have been assigned the same color. If there is any color assignment that does not violate the rule, mark the color as the part of the solution. If at any point if time the assignment of color is not possible we should backtrack and try with different colors that are available to us.

Therefore Sudoku can be considered as a graph with 81 vertices. Each node (or cell) has an edge to every other node (cell) in its respective column, row, and 3 x 3 grid. Fig. 4 depicts the same.



(a) Unsolved sudoku (b) Solved sudoku

Fig. 6: Graph coloring for sudoku

So we can say that Sudoku can be viewed as Graph and thus can be solved by Graph Coloring with a Chromatic Number, G = 9.It is no different to using 9 different colors to color the vertices in a way that no two adjacent vertices have the same color.

1) Algorithm used :

- Create a recursive function that considers the current vertex index.
- If the current vertex is colored return true..
- Else pick a color from the set of colors in the given order.
- Check whether is it safe to assign the color to the considered vertex.
- If yes assign the color and call the recursive function continuously till all the vertices are filled.
- If at any point of coloring assignment of color is not possible return false and backtrack.
- Try assigning the vertex with the next possible color and move forward in the same way.
- The recursive function should return true if there is a possible way to color all the vertices else it should return false.

III. Results and Analysis

All runs and testing was done on our machine which is has i7 processor (4.4GHz) Windows XP machine with 3.18 GB RAM. All the computation required and the results obtained are from the above device.

A. Wedding seating Problem

The problem and algorithm described above has been implemented in C++ which uses the concepts of graph coloring and backtracking. The time taken to produce the outputs where quite low and all the solutions where right.

1) Testing:

- Fig. 6 shows the test case where the solution exists and where guests '1' and '2' does not want to seat next to each other. Fig. 6(a) corresponds to the output where the number of guests are two and Fig. 6(b) corresponds to the output where the number of guests are four.
- Fig. 7 shows a test condition where the solution does not exist. The test case is when there is only one table and two guest does not want to sit next to each other.

2) Complexity Analysis:

- Time complexity: O(k$\hat{}$(nn))
- Space Complexity: O(nn) To store the output array a matrix is needed.

B. Sudoku

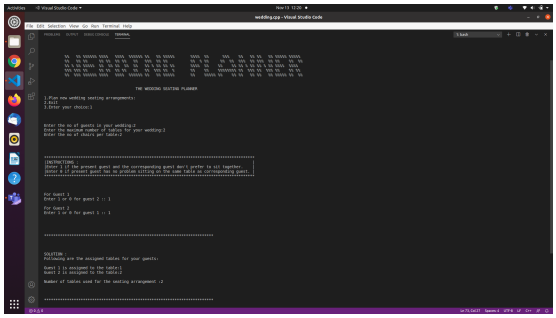
So to check whether graph coloring could be really used for solving sudoku , we developed a c++ script that actually uses concepts of graph coloring and backtracing to solve the sudoku. All results where obtained under two seconds (2 secs). The results and analysis are as follows :

1) Testing:

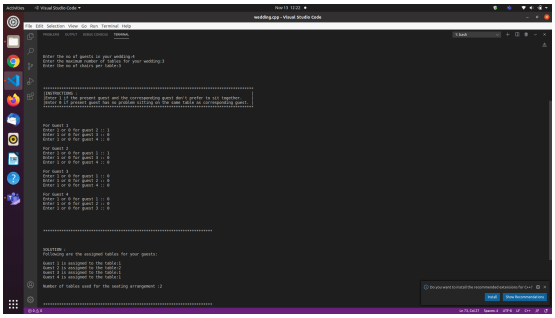
- Fig. 8 shows the test case where the solution exists and we can see that the output produced is correct with 1-9 numbers in each column, row and in smaller box, which satisfies all conditions of a sudoku.
- Fig. 9 shows a test condition where the solution does not exist and the corresponding output is displayed.

2) Complexity Analysis:

- Time complexity: O(9$\hat{}$(nn)):
  For every unassigned index, there are 9 possible options so the time complexity is O(9$\hat{}$(nn)). The time complexity remains the same but there will be some early pruning so the time taken will be much less than the naive algorithm but the upper bound time complexity remains the same.
- Space Complexity: O(nn)
  To store the output array a matrix is needed.



(a) No. of guests = 2



(b) No. of guests = 4

Fig. 7: Test case where solution exists

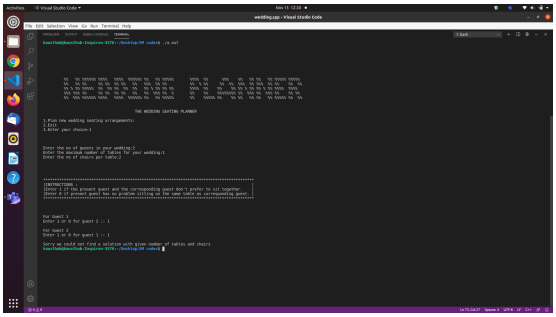


Fig. 8: Test case where solution does not exists

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(a) input

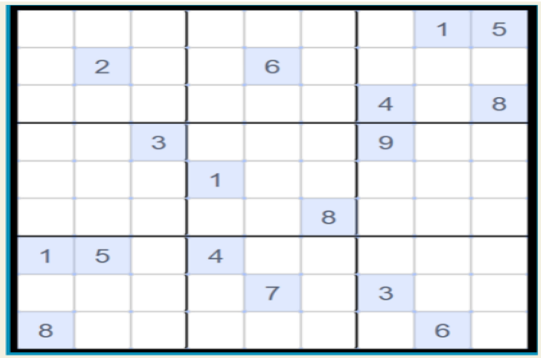
```
Sloving the sudoku.....

After sloving the sudoku :

5   3   4   |   6   7   8   |   9   1   2
6   7   2   |   1   9   5   |   3   4   8
1   9   8   |   3   2   4   |   5   6   7
-----------------------------------------
8   1   9   |   7   6   2   |   4   5   3
4   2   6   |   8   5   3   |   7   9   1
7   5   3   |   9   4   1   |   8   2   6
-----------------------------------------
9   6   1   |   5   3   7   |   2   8   4
2   8   7   |   4   1   9   |   6   3   5
3   4   5   |   2   8   6   |   1   7   9
```

(b) Output

Fig. 9: Test case where solution exists



(a) Input

```
Sloving the sudoku.....

After sloving the sudoku :

No solution exists
```

(b) Output

Fig. 10: Test case where solution does not exists

Task ID	Task Description	Task Duration	Start Date	End date	8/29/2020	9/3/2020	9/4/2020	9/7/2020	9/8/2020	9/27/2020	9/28/2020	10/1/2020	10/2/2020	10/6/2020	10/7/2020	10/17/2020	10/18/2020	10/20/2020	10/21/2020	10/24/2020	11/5/2020	11/10/2020
1	Getting familarize with graph and graph coloring	5	8/29/2020	9/3/2020																		
2	Finding all the necessary docs and papers	4	9/4/2020	9/8/2020																		
3	Developing script for wedding seating problem	20	9/7/2020	9/27/2020																		
4	Analysis of results of wedding seating problem	4	9/28/2020	10/2/2020																		
5	Optimised the script to give minimum table	5	10/1/2020	10/6/2020																		
6	Developing script for Sudoku slover	10	10/7/2020	10/17/2020																		
7	Analysis for results of sudoku solver	3	10/18/2020	10/21/2020																		
8	Updating the sudoku slover script to handle all the cases	4	10/20/2020	10/24/2020																		
9	Report Prepartion and complilation	5	11/5/2020	11/10/2020																		

Fig. 11: Gantt chart

IV. Conclusion

To solve the Wedding Seating Problem and Sudoku , graph coloring - a concept of discrete mathematics - comes into handy. Using graph coloring the imaging and finding solutions to the above mentioned problems can be easily done. Similarly, many real life applications such as time scheduling and register allocation is done using this wide concept of graph coloring **Future work** would be to include more criteria for the WSP like equal number of guests per table, families and close relations together etc.

V. Base paper

"Creating Seating Plans: A Practical Application" by Rhyd Lewis and Fiona Carroll, Cardiff School of Mathematics, Cardiff University, WALES was taken as our base paper for the wedding seating problem (WSP).

VI. Acknowledgment

We would like to thank Mr. Manjunath K and Mr. Dinesh Naik for providing us with the necessary inputs and information for completing this project. Moreover special mention to National Institute of Technology Karnataka, Information Department for providing this wonderful opportunity.

VII. Individual Contributions

- **Kausthub Thekke Madathil (191IT125)** : Coding, code documentation, PPT, co-ordinated the peoject
- **Neeraj Mirji (191IT232)** : Coding, documentation, PPT presentation
- **Puttaraja (191IT139)** : Coding, documentation, Report presentation
- **Sohanraj R (191IT149)** : Coding, documentation, Report presentation

The Fig.11 shows the gantt chart.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] "Creating Seating Plans: A Practical Application" ,Rhyd Lewis and Fiona Carroll, Cardiff School of Mathematics, Cardiff University, WALES

[3] https://en.wikipedia.org/wiki/Sudoku_graph

[4] https://en.wikipedia.org/wiki/Graph_coloring

[5] http://www.cs.kent.edu/ dragan/ST-Spring2016/SudokuGC.pdf

[6] https://www.geeksforgeeks.org/