A Novel Heuristic for Estimating Two-Colorability for Conflict-Free Coloring in Simple Graphs

Rishabh Singhal, Ayushi Sengar, V. Prem Prakash, and Mayank Agrawal

1 Introduction

Graph coloring is a special case of graph labeling, wherein labels conventionally referred to as 'colors' are assigned to graph elements, subject to some constraints. Basically, graph coloring aims to assign colors to graph vertices in such a manner that adjacent vertices are not assigned the same color. In this context, the chromatic number refers to the smallest number of colors that can be used to cover all the vertices of the graph.

This process of simple graph coloring has a serious drawback, especially when the problem of assigning different frequencies to the n base stations shows up, involving a number of colors to be assigned to each vertex. The traditional approach to graph coloring needs large values of chromatic number, which is not desirable due to the high costs associated [1]. Hence, in order to further bring down the number of frequency assignments such that there is no interference of frequencies, conflict-free coloring comes into use.

Conflict-free coloring is the process of coloring vertices of a graph such that every vertex has a unique color when compared to other vertices in its neighborhood. The main purpose is to reduce the chromatic number of a graph. We consider two cases:

R. Singhal (✉) · A. Sengar · V. Prem Prakash · M. Agrawal
Dayalbagh Educational Institute, Agra, India
e-mail: r.singhal.com@gmail.com

A. Sengar
e-mail: ayushisengar48@gmail.com

V. Prem Prakash
e-mail: vpremprakash@dei.ac.in

M. Agrawal
e-mail: dei.mayank.agrawal@gmail.com

V. H. Saran and R. K. Misra (eds.), *Advances in Systems Engineering*,
Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-15-8025-3_33

1. Open Neighborhood, in which the concerned vertex does not belong to its own neighborhood, so each vertex is colored such that exactly one vertex in its neighborhood is uniquely colored.
2. Closed Neighborhood, in which the concerned vertex also belongs to its own neighborhood, thus requiring that each vertex is colored such that there is a vertex with a unique color including itself and its neighborhood.

Conflict-free closed neighborhood two-coloring was shown to be NP-complete by Gargano and Rescigno [2]. The recent work in this field has focused on coloring special graphs generated by specific geometric objects such as intervals [3], rectangles [4] and unit disks [5]. A heuristic for conflict-free closed neighborhood coloring has also been proposed for an interval graph with at most four colors [6].

This paper proposes a novel polynomial-time heuristic to determine, for any given graph $G = (V, E)$, if it has a conflict-free closed neighborhood chromatic number of 2. The rest of the paper is organized as follows. The proposed heuristic is described in Sect. 2, and experimental results are given in Sect. 3, and concluding remarks made in Sect. 4.

2 Proposed Heuristic

2.1 Overview

The novel heuristic proposed in this work is described in this section. In this context, some relevant definitions and terminology are given as follows.

Given a $G = (V, E)$ be a graph, for a vertex $v \in V(G)$, $N(v)$ denotes the set consisting of all vertices which are adjacent to v, called open neighborhood of v [6].

- The set $N[v] = N(v) \cup \{v\}$ is called the closed neighborhood of v.
- A coloring C_G of a graph G is called conflict-free closed neighborhood (CF-CN) coloring if for every vertex $v \in V(G)$, the set $N[v]$ has a unique color.
- A coloring C_G of a graph G is called conflict-free open neighborhood (CF-ON) coloring if for every vertex $v \in V(G)$, the set $N(v)$ has a unique color.
- The minimum value k for which there is a CF-ON (respective CF-CN) coloring of G with k colors is called the CF-ON (resp. CF-CN) chromatic number of G and is denoted as $\chi_{\text{cf}}(G)$ (respective $\chi_{\text{cf}}[G]$).

The heuristic makes use of the following ideas. Consider a closed neighborhood for a vertex, say 'X'. For 'X', we can have three possible coloring conditions:

- There are some vertices which are colored in a specific color, say color 2
- There are some vertices which are colored in a specific color, say color 3
- The remaining vertices are not assigned any color as yet.

Our objective is to now color these uncolored vertices on the same principles using just two colors—color 2 and color 3 in this case. Assume the count of color 2 be 'α'

and count of color 3 be 'β'. The relationship between 'α' and 'β' can physically be interpreted in different ways, which eventually will help us to color those uncolored vertices.

- 'α' = 'β' = 0: No vertex has been colored yet in the closed neighborhood of 'X'. Our strategy here would be to color the vertex 'X' with color 2 and remaining vertices in its neighborhood with color 3. In this way, color 2 will be unique in the closed neighborhood of 'X'. Now, if 'X' has exactly one neighbor, then it cannot be colored using only two colors. Hence, $\chi_{\text{cf}}[G] \neq 2$.
- 'α' = 1 and 'β' = 0: Color 2 is already unique, and no vertex has been assigned color 3. Thus, our task here is to assign the color 3 to all the remaining vertices in the neighborhood of 'X'. Again, if 'X' has exactly one neighbor, then it cannot be colored using only two colors. Hence, $\chi_{\text{cf}}[G] \neq 2$.
- 'α' = 0 and 'β' = 1: Color 3 is already unique, and no vertex has been assigned color 2. Thus, our task is now to assign the color 2 to all the remaining vertices in the neighborhood of 'X'. Again, if 'X' has exactly one neighbor, then it cannot be colored using only two colors. Hence, $\chi_{\text{cf}}[G] \neq 2$.
- 'α' = 0 and 'β' > 1: If we have at least one uncolored vertex, then assign one vertex color 2 and remaining vertex with color 3. If we have no uncolored vertex with this condition, then the given graph cannot be colored using two colors. Hence, $\chi_{\text{cf}}[G] \neq 2$.
- 'α' > 1 and 'β' = 0: If we have at least one uncolored vertex, then assign one vertex color 3 and remaining vertex with color 2. If we have no uncolored vertex with this condition, then the given graph cannot be colored using two colors. Hence, $\chi_{\text{cf}}[G] \neq 2$.
- 'α' = 1 and 'β' > 1: This condition is already satisfying the definition. Our job is to color the remaining vertices with color 3.
- 'α' > 1 and 'β' = 1: This case also satisfies the definition. Our job is to color the remaining vertices with color 2.
- 'α' = 1 and 'β' = 1: If there is no uncolored vertex left, then $\chi_{\text{cf}}[G] \neq 2$. But if there are some uncolored vertex left, choice has to be made between color 2 and color 3. Assign color 2 and proceed, if the graph cannot be colored in this fashion, go back to the last occurring case and then assign color 3 to validate the result.
- 'α' > 1 and 'β' > 1: In this case, graph cannot be colored using just two colors. Hence, $\chi_{\text{cf}}[G] \neq 2$.

The proposed heuristic operates using the above simple rules. The closed neighborhood is accessed by the rows of the adjacency matrix of the generated graph, taken one at a time. Before actually accessing the adjacency matrix, replace all the diagonal elements with '1' so as to include that vertex also in the conflict-free closed neighborhood coloring. The neighborhood of 'X' is simply defined as the non-zero columns of the matrix for the row 'X'. For each row taken under operation, assign two counters to count color 2 and color 3 assigned with nodes in row 'X' and an array to hold the nodes position, which is simply the column number, that are not yet been assigned any color and are neighbor of 'X'. Whenever we assign any node a color, replace '1' in the columns of the adjacency matrix for the corresponding row

with that color—'2' or '3', in our case. In next step, replace all the row elements with entry '1' for that specific column whose color has been changed, with the color assigned. It will help us to calculate number of color 2 and color 3 assigned to nodes when we change the neighborhood. Repeat the above steps 'N' times, where 'N' is the number of nodes. Pseudo-code for the proposed heuristic is given in listing 1.

2.2 Erdos–Renyi Graph Model

The test graphs used in the experimental work in this paper were generated using the Erdos–Renyi graph model [7]. Specifically, the model used is the $G(n, p)$ model, wherein a graph is constructed by connecting nodes randomly. Each edge is included in the graph with probability p independent from every other edge. Equivalently, all graphs with n nodes and M edges have equal probability of

$$p^M(1-p)^{\binom{n}{2}_C - M} \tag{1}$$

The parameter p in this model can be thought of as a weighting function; as p increases from 0 to 1, the model becomes more likely to include graphs with more edges and relatively less likely to include graphs with fewer edges. In particular, the case $p = 0.5$ corresponds to the case where all $2^{\binom{n}{2}_C}$ graphs on n vertices are chosen with equal probability.

Listing 1

```
Function conflictfree(Adjacency Matrix amat, Nodes)
{
Array: nodecolor[nodes] = {0}
amat[i][i] = 1
Repeat from 0 to nodes-1
{
Array: nocol [nodes] = {0} // position ofuncolored
nodes
int col2 = 0, col3 = 0 // count of nodes with color
2 & 3
int flag = 0 // avoid repetitive coloring of same
node
IF (col2 = 1), u2 = 1
ELSE IF (col3 = 1 and u2 ≠ 1), u3 = 1.

  Repeat from 0 to length(nocol) -1
  {
```

```
{
IF (u2 ≠ 1 and flag = 0); color of first uncolored
node = 2, update nodecolor, change all the values of
corresponding rows to 2 of that column and set flag
to 1.
IF (col2 > 1 and col3 ≠ 0), u3 = 1.
IF (col2 = 1 or (col2 ≠ 1 and u3 ≠ 1)), u2 = 1
}
{
IF (u3 ≠ 1 and flag = 0); color of first uncolored
node = 3, update nodecolor, change all the values of
corresponding rows to 3 of that column and set flag
to 1.
IF (col3 > 1 and col2 ≠ 0), u2 = 1.
IF (col2 = 1 or (col2 ≠ 1 and u2 ≠ 1)), u3 = 1
}
Flag = 0
}

  Col2 = 0, col3 = 0

}
Check the conditions that could violate conflict
free coloring. If found return "Not possible", else
return the array of colors with elements at diagonal
of adjacency matrix.
}
```

3 Experimental Work

The proposed heuristic was implemented in Python 3.6 environment: idle (Python 3.6 64-bit), OS: Windows 10 Home, Processor: Intel (R) Core (TM) i5-7200U CPU @ 2.5 GHz, RAM: 8.00 GB, System Type: 64-bit OS.

For $n = 5, p = 0.7$ the resultant colored graph is shown in Fig. 1.

Likewise, different set of probabilities, number of nodes and the computation time, several graphs were plotted.

The result of graphs generated using Erdos–Renyi is summarized in Table 1. The first column indicates the number of nodes in given graph and the second indicates probability of generating edge between two pair of nodes. Column three represents the time taken (seconds) to compute the result and the last column indicates the computational result: 'positive' indicates that the graph can be colored and 'negative' means that graph cannot be colored with two colors.

The heuristic took 0.00157 s to determine that a graph with 10 nodes and edge probability of 0.33 can be two-colored (Fig. 2). Next, it took 0.01562 s to evaluate a

graph of 50 nodes with edge probability of 0.45. Similarly, various other instances were also evaluated likewise. Lastly, a graph of 1000 nodes with 2,499,500 edges was tested in 0.46362 s.

Some cyclic graphs were generated and their computational result is summarized in Table 2. The first column indicates number of nodes followed by computational

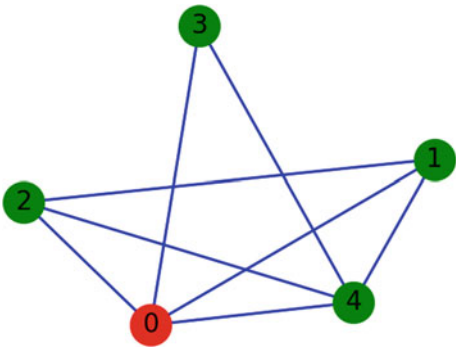

Fig. 1 Conflict-free closed neighborhood-colored five-node graph

Table 1 Test cases for Erdos–Renyi graphs

Nodes	Probability	Computation time (s)	Result
10	$p = 0.33$	0.0015784	Positive
50	$p = 0.45$	0.01562	Negative
500	$p = 0.7$	0.11591	Negative
846	$p = 0.8$	0.2853	Negative
500	$p = 1$	0.11269	Positive
1000	$p = 1$	0.46362	Positive

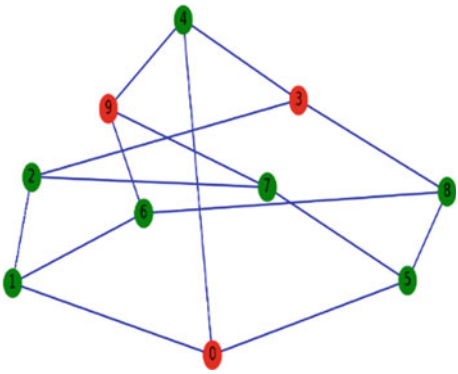

Fig. 2 Conflict-free closed neighborhood-colored 10-node graph

Table 2 Test cases for cyclic graphs

Nodes	Computation time (s)	Result
20	0.00565	Positive
400	0.10025	Positive
555	0.15371	Positive
1009	0.4168	Positive
10,000	46.5499	Positive

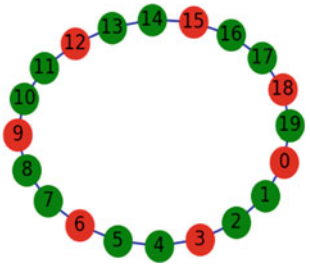

Fig. 3 Cyclic graph with 20 nodes

time (seconds) and the last column gives the result.

- Cyclic graph of 20 nodes was generated, and heuristic gave positive result in 0.00565 secs. Coloring pattern is shown in Fig. 3.
- Similarly, odd cycles of 555 nodes and 1009 nodes and even cycles of 400 and 10,000 nodes were generated and were tested.

Further, cyclic graphs with nodes ranging from 1000 to 15,000 were plotted against computational time in seconds (Fig. 4).

Graphs with nodes ranging from 100 to 5000 were generated using Erdos-Renyi graph with $p = 1$ (Fig. 5) and $p = 0.5$ (Fig. 6). Working of heuristic on those nodes were plotted against computational time.

4 Conclusions

Conflict-free coloring in closed neighborhood as presented in this paper is one possible generalization of the traditional approach to graph coloring. Conflict-free coloring of graphs finds practical applications in the cellular network domain for problems ranging from radio frequency identification to frequency assignment. The

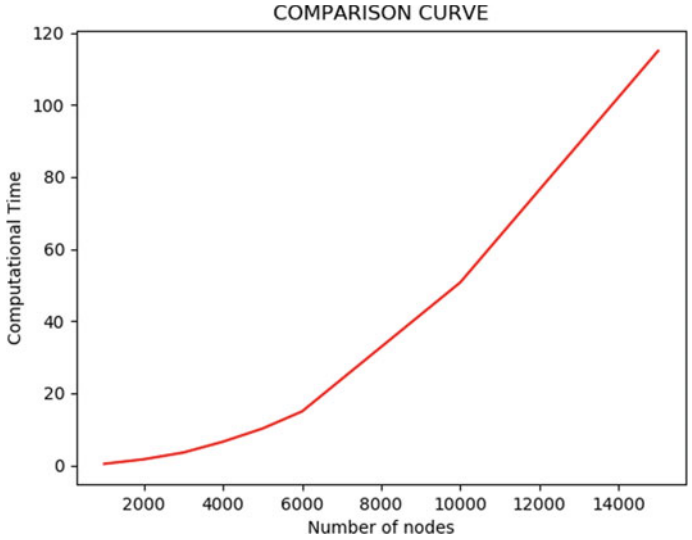

Fig. 4 Compare size of graph against computational time (seconds) for cyclic graph

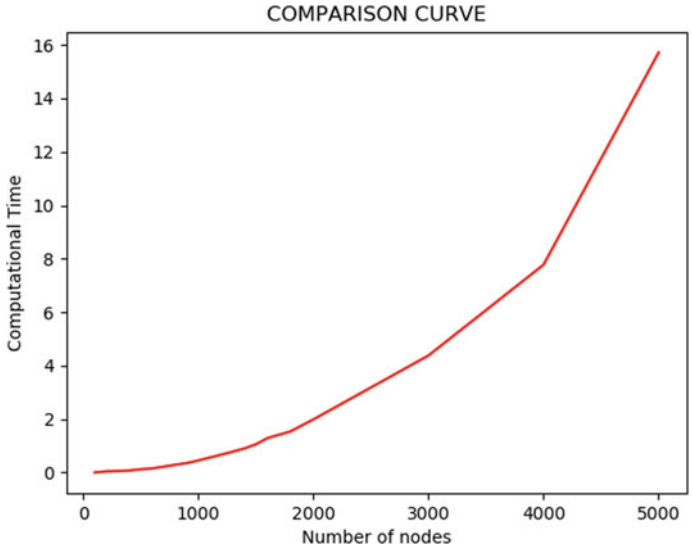

Fig. 5 Compare size of graph against computational time (seconds) for Erdos–Renyi graph with edge probability 1

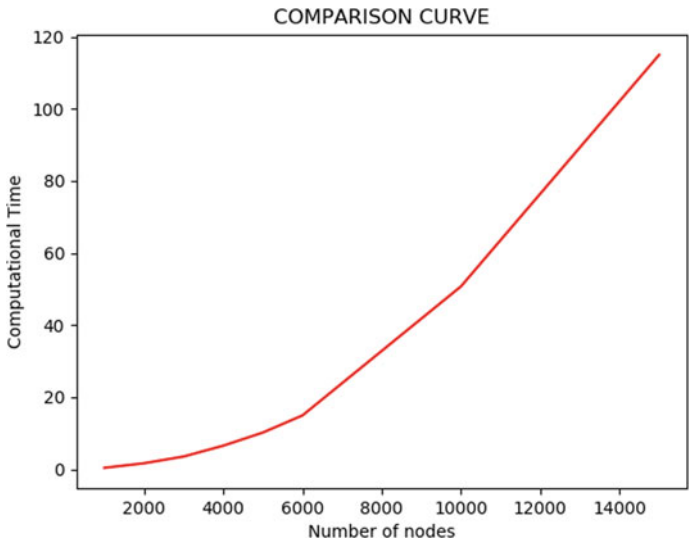

Fig. 6 Compare size of graph against computational time (seconds) for Erdos–Renyi graph with edge probability 0.5

problem of two-coloring a conflict-free closed neighborhood is known to be NP-complete. The novel heuristic presented in this work solves this problem in polynomial time with big O complexity of $O(n^3)$, and the results are given on a wide range of random graphs instances.

References

1. Cheilaris P (2009) Conflict-free coloring. Dissertation, City University of New York, p 5
2. Gargano L, Rescigno AA (2015) Complexity of conflict-free colorings of graphs. Theoret Comput Sci 566:39–49
3. Bar-Noy A, Cheilaris P, Smorodinsky S (2008) Deterministic conflict-free coloring for intervals: from offline to online. ACM Trans Algorithms (TALG) 4(4):44
4. Ajwani D, Elbassion K, Govindarajan S, Ray S (2007) Conflict-free coloring for rectangle ranges using $O(n^{0.382})$ colors. In: Proceedings of the nineteenth annual ACM symposium on parallel algorithms and architectures. ACM, pp 181–187
5. Lev-Tov N, Peleg D (2009) Conflict-free coloring of unit disks. Discrete Appl Math 157(7):1521–1532
6. Reddy V (2017) Parameterized algorithms for conflict-free colorings of graphs. arXiv:1710.00223v1 [cs.DS], 30 Sept 2017
7. Erdős P, Rényi A (1959) On random graphs. I (PDF). Publ Math 6:290–297