Coloring Inductive Graphs Online

Project Report

COMP 691: Online Algorithms and Competitive Analysis

Author: *Yan Ren (40212201)*

Department:	Computer Science & Software Engineering
Date:	April 6, 2023

Contents

1 Introduction

This project report presents the results of a study based on the paper ”Coloring Inductive Graphs On-Line” by Sandy Irani [1], which examines the First Fit online graph coloring algorithm in the context of d-inductive graphs. Online graph coloring is a process whereby a graph is presented one node at a time, along with the edges connecting it to previously presented nodes. Each node is assigned a color different from its neighbors before the next node is presented, with the objective of minimizing the number of colors used. Formally, the online graph coloring problem discussed by Sandy Irani can be defined as follows:

Online Graph Coloring

Input:	$G = (V, E, \prec)$ where G is an d-inductive graphs.
	$\prec$ is a total order on V.
	$(v_1, N_1), (v_2, N_2), \ldots, (v_n, N_n)$
	input sequence, where $v_1 \prec v_2 \prec \cdots \prec v_n$
	and $N_i = N(v_i) \cap \{v_j : j < i\}$.
Output:	$c : V \to [k]$, c is the color of vertices V,
	k is the number of colors used.
Objective:	to find c so as to minimize k subject to c being a valid coloring
Chromatic number:	χ , smallest number of need for coloring

Online graph coloring can be applied to a range of optimization problems in computer science, including processor assignment and register allocation. These problems involve assigning tasks or variables to available resources such as processors or registers to minimize resource usage and maximize performance [2]. Online graph coloring algorithms can be used to solve these problems in real-time as tasks or variables arrive incrementally. By assigning colors to them as they arrive, these algorithms can optimize resource usage and improve system efficiency. For instance, in processor assignment, online graph coloring algorithms can schedule incoming tasks to available processors while minimizing the number of processors needed [2]. Similarly, in register allocation, these algorithms can assign variables to available registers to minimize the number of registers needed and maximize code performance. Online algorithms are generally compared to the optimal offline algorithms to measure their efficiency. In Sandy Irani’s study, she examines online graph coloring for the class of inductive graphs using the First Fit. A d-inductive graph is defined as one where nodes can be assigned distinct numbers in such a way that each node is adjacent to at most d higher-numbered nodes. First Fit assigns to each node the lowest-numbered color possible, i.e., the lowest-numbered color such that the node is not already adjacent to any nodes of that color. The author shows that First Fit will use $\Omega(d \log n)$ colors on any d-inductive graph with n nodes. Since $d + 1$ is an upper bound on the chromatic number of any d-inductive graph, this yields a bound of $\Omega(\log n)$ on the performance ratio of First Fit for d-inductive graphs. The author also discusses the effect of lookahead on the model, showing that with lookahead $n/\log n$, an online algorithm still requires $\Omega(d \log n)$ colors to color a d-inductive graph, while for lookahead $l > n/\log n$, the graph can be colored online in $\Theta(min\{d \log n, dn/l\})$ colors.

2 Background

Online graph coloring has been extensively studied in computer science. Lovasz et al. proposed an algorithm with a performance ratio of $\mathcal{O}(n/\log^* n)$ on all graphs [3], which is close to the best any online algorithm can do for general graphs. Lower bounds $\Omega(n/(\log n)^2)$ [1] for performance ratios have also been established for online algorithms on general graphs. Researchers have explored online coloring of more restricted classes of graphs such as bipartite graphs, which can be colored using $\mathcal{O}(\log n)$ colors. Optimal performance ratios have been achieved for interval graphs [4] and split graphs, complements of bipartite graphs, and complements of chordal graphs [5]. In general, First Fit performs poorly in these restricted classes. There exist bipartite graphs that require k colors, where $2k$ is the number of nodes, indicating poor performance for First Fit. Vishwanathan has presented a randomized online algorithm [6] for coloring graphs with a constant chromatic number but where the size of the graph can grow, achieving a competitive ratio of $\mathcal{O}(n/\sqrt{\log n})$ colors. Online graph coloring algorithms have practical applications in areas such as scheduling and register allocation in compilers, and further research in this area could lead to improved performance in various computer systems. The provided algorithms could also be useful in solving a range of other combinatorial optimization problems.

3 Online Algorithm

For this project, the chosen online algorithm is First Fit, which is being implemented in the class of inductive graphs. The algorithm assigns the lowest possible color to each node that is not adjacent to any nodes of that color. After the adversary presents node j, the algorithm examines the graph and determines the neighborhood $N(j)$ of node j. Suppose node j is assigned color c, then for every color in the range $1...c-1$, there is a node in $N(j)$ that has been assigned that color. The algorithm then fixes a set of nodes S_j that satisfies certain conditions, such that having exactly one node in S_j that has been assigned a unique color in $1, ..., c-1$. The size of S_j is $c-1$, and for every i in S_j, $i < j$. The algorithm assigns the lowest-numbered color to each node to ensure that no adjacent nodes have the same color.

First Fit for Online Graph Coloring

```
1: function FIRSTFIT($G$)
2:     $colors \leftarrow \varnothing$                                  ▷ Initialize an empty set of colors
3:     for all $j = 1, 2, \ldots$ do                          ▷ For each node presented by the adversary
4:         $N_j \leftarrow$ set of colors of neighbors of node $j$ in $G_{j-1}$ ▷ Get the neighborhood of node $j$ in the current graph
5:         $c \leftarrow 1$                                              ▷ Start with the first color
6:         while $c \in N_j$ do                        ▷ Find the lowest-numbered color not in $N_j$
7:             $c \leftarrow c + 1$
8:         end while
9:         Assign color $c$ to node $j$ in $G_j$        ▷ Assign color $c$ to node $j$ in the updated graph
10:        $colors[j] \leftarrow c$                            ▷ Record the color assignment for node $j$
11:    end for
12:    return $colors$                             ▷ Return the color assignments for each node
13: end function
```

The algorithm's approach is beneficial for obtaining a good competitive ratio because its simple greedy nature minimizes the number of colors used to color the graph. It performs as well as any online algorithm for d-inductive graphs. In fact, it has been shown that for any d and any online graph coloring algorithm A, there exists a d-inductive graph that forces A to use $\Omega(d \log n)$ colors to color G [1]. For any given d, the competitive ratio of First Fit for coloring inductive graphs is at least $\Omega(\log n)$ [1]. Therefore, using First Fit, which assigns colors to nodes based on a simple and intuitive heuristic, provides a competitive ratio that is at least as good as any other online algorithm for coloring inductive graphs. If the algorithm were to make different decisions, such as assigning colors randomly or according to a different heuristic, it could result in a larger number of colors being used to color the graph, which would lead to a worse competitive ratio.

4 Implementation details

The aim of this implementation is to simulate the First Fit algorithm for online d-inductive graph coloring. The first part of the implementation involves constructing a d-inductive graph that simulates an adversary generating a d-induction graph using the VAM-PH input model. In each step, a new node is presented along with its neighbors from the previous nodes, and the edges between the new node and the existing nodes are randomly selected. The major challenge in this step is to maintain the d-inductive graph property, which requires that each node has at most d edges to higher-numbered nodes. The graph simulation function takes the total number of nodes (n) and inductive parameter (d) as inputs. To maintain the d-inductive property, each newly added node is labeled in reverse order, and its neighbors are determined by running random selection d times among the nodes that appeared before. Using reverse order to number the node makes sure that there is always a number sequence for each node that satisfies the d-inductive graph property. For example, to generate a 2-inductive 10-node graph, the first node is labeled as 10, the second as 9, and so on. When determining the neighbors of the newly added node, random selection is run twice among all previously appeared nodes. The selected nodes are then connected to the newly added node.

The second part of the implementation is the First Fit coloring. For each node added to the graph, all of its neighbors are obtained, and the lowest-numbered color that is not used by its neighbors is found. To help with color numbering, a pre-generated color_dict.json file containing 100,000 colors is used, with labels from 0 to 99,999 and hexadecimal color codes. This file is loaded at the beginning of the program, and the color label is used for node coloring and comparison of the lowest-numbered color with the nodes' neighbors. The hexadecimal color code is used for visualization and to verify the correctness of the implementation.

Below, a 10-node graph with 2-inductive is presented, and at each step, First Fit colors the node. A plot of the graph is then displayed, where the node number demonstrates the correctness of the 2-inductive graph, and node color show that no two nodes with the same color have an edge between them. By the end of graph generation, First Fit uses 3 colors for the entire graph. Figure 1 shows the resulting plots.

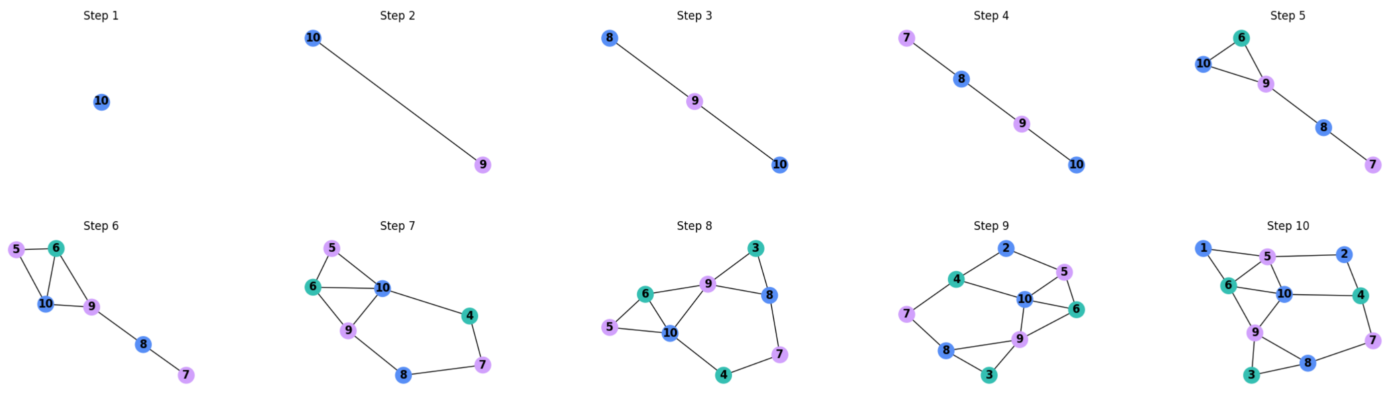


Figure 1: 2-inductive graph with 10 nodes

5 Experimental Setup

The experiment was implemented in Python 3.9 and conducted on a machine with a 14-core 2.3GHz CPU and 16GB RAM. The two controlled variables in the graph simulation were the total number of nodes (n) in the graph and the inductive parameter (d). The experiments were designed based on various numbers of graph nodes and inductive parameters. The number of colors used under different experiment setups and runtime was collected to generate line plots to show the trends for result analysis.

Three specific experiments were designed as follows:

5.1 Experiment 1

Experiment 1 investigates the relationship between the number of nodes and the number of colors used by the First Fit, with a fixed inductive parameter of 5, resulting in a planar graph in each graph simulation step. The range of nodes is from 10 to 10,000, with a step of 10. In the beginning, the graph is initialized with 0 nodes, and 10 nodes are added step by step following the VAM-PH input mode. Then, the graph is reinitialized to 0 nodes, and 20 nodes are added using the same method. The process is repeated until the graph reaches 10,000 nodes. The number of colors used by First Fit to color each graph is recorded along with the runtime.

5.2 Experiment 2

Experiment 2 investigated the relationship between the inductive parameter and the number of colors used by First Fit. The experiment followed the same method as Experiment 1 but with a fixed total of 1000 nodes and an inductive parameter ranging from 2 to 500 with a step of 1. The number of colors used by First Fit to color each graph was recorded along with the runtime.

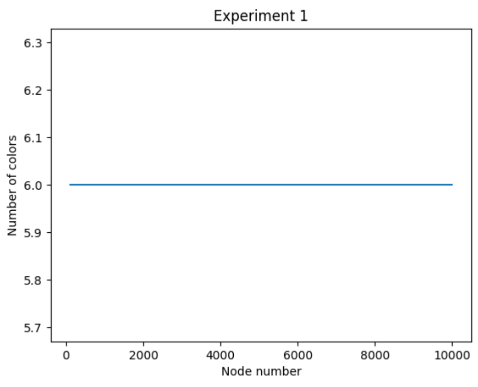
5.3 Experiment 3

Experiment 3 consists of two groups. Group 1 includes graphs with 500, 1000, and 2000 nodes, while group 2 includes graphs with 100, 1000, and 10000 nodes. All graphs in each group were run on an inductive parameter ranging from 1 to 80. The graphs in group 1 have a total number of nodes differing by a factor of 2, while the graphs in group 2 have a total number of nodes differing by a factor of 10. The purpose of this experiment was to investigate how the number of nodes affects the increase in the number of colors when the inductive parameter is increased. The number of colors used by First Fit for each graph under different inductive parameters was recorded.

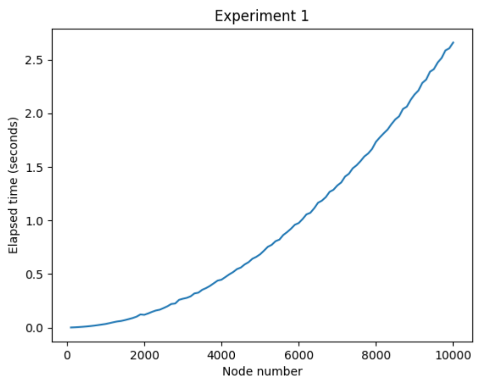
6 Results

6.1 Experiment 1

Experiment 1 investigated the relationship between the number of nodes and the number of colors used by First Fit on a fixed 5-inductive graph. The range of nodes was from 10 to 10,000, with a step of 10. The result showed a constant number of colors used by First Fit, even as the number of nodes increased, Figure 2a. This is consistent with Sandy Irani's proof that First Fit uses $O(d \log n)$ colors on d-inductive graphs [1]. Since the total number of nodes in this experiment is relatively small compared to the logarithmic growth, the number of colors remained constant and was lower than the upper bound of $O(d \log n)$. However, as the number of nodes increased, the runtime for the graph also increased, Figure 2b, as each node was presented one at a time followed by First Fit coloring.



(a) Number of colors plot



(b) Runtime plot

Figure 2: Results for Experiment 1

6.2 Experiment 2

Experiment 2 examined a 1000-node graph under inductive parameters ranging from 2 to 500. The number of colors used by First Fit increased approximately linearly with the inductive parameter, confirming the linear relationship between inductivity and color numbers, Figure 3. When the graph reached a 500-inductive, the number of colors used was 128.

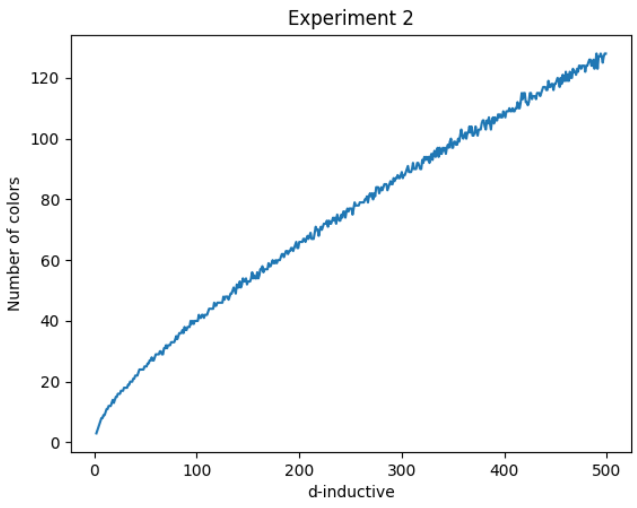
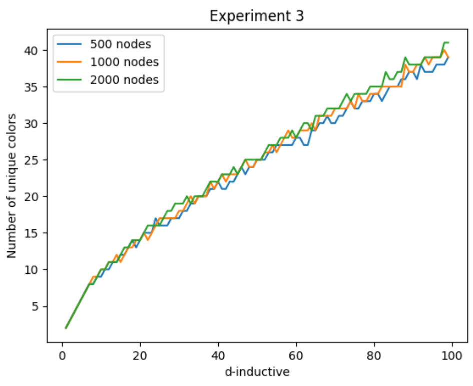


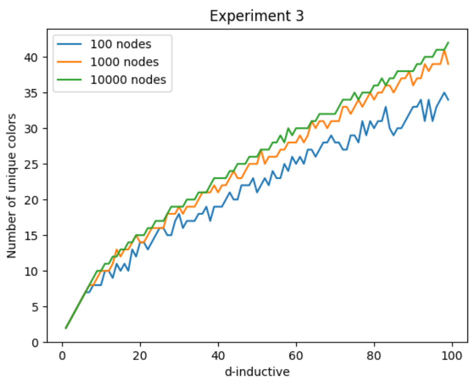
Figure 3: Number of colors plot for Experiment 2

6.3 Experiment 3

Experiment 3 aimed to investigate how the number of nodes affects the increase in the number of colors as the inductive parameter increases. Two groups were used: Group 1 included graphs with 500, 1000, and 2000 nodes, while Group 2 included graphs with 100, 1000, and 10000 nodes. All graphs in each group were run on an inductive parameter ranging from 1 to 80. Group 1's graphs had a total number of nodes differing by a factor of 2, while Group 2's graphs differed by a factor of 10. The results showed that when the graphs differed only by a small magnitude, the number of colors did not differ significantly as the inductive parameter increased, Figure 4a. The three plots for Group 1 showed similar increasing curves and had approximately the same number of colors used. In contrast, Group 2's graphs showed different slopes, Figure 4b. The 10000-node graph had a steeper slope than the 1000-node graph, and the 1000-node graph had a steeper slope than the 100-node graph. The graphs also differed in the number of colors used when they were all under 80-inductive. The 10000-node graph used 44 colors, the 1000-node graph used 37 colors, and the 100-node graph used 32 colors.



(a) Number of colors plot for Group 1



(b) Number of colors plot for Group 2

Figure 4: Results for Experiment 3

7 Future directions

One possible direction for future research is to investigate the behavior of First Fit and other graph coloring algorithms on more complex graphs, such as hypergraphs or graphs with weights on the nodes and edges [7]. This could shed light on the practical performance of graph coloring algorithms in real-world applications. Additionally, it would be worthwhile to compare the performance of First Fit with other graph coloring algorithms, such as Greedy, DSatur, or Backtracking [8], in terms of their theoretical time and space complexity, as well as their empirical performance on various types of graphs. Finally, another interesting area to explore would be the use of parallel computing to speed up graph coloring algorithms [9, 10]. This could involve developing parallel versions of First Fit or other graph coloring algorithms and testing their performance on large-scale graphs. With more time, these avenues of research could lead to a deeper understanding of the limitations and strengths of graph coloring algorithms and could contribute to the development of more efficient and effective algorithms for solving graph coloring problems.

References

[1] S. Irani, “Coloring inductive graphs on-line,” *Algorithmica*, vol. 11, pp. 53–72, 1994.

[2] B. Bhattacharya, S. Chakraborty, and P. Dasgupta, “Online graph coloring with applications to processor assignment and register allocation,” *SIAM Journal on Computing*, vol. 43, no. 6, pp. 2099–2118, 2014.

[3] L. Lov’asz, M. E. Saks, and W. T. Trotter, “An on-line graph coloring algorithm with sublinear performance ratio,” *Discrete Mathematics*, vol. 75, no. 1-3, pp. 319–326, 1988.

[4] H. A. Kierstead, “The linearity of first-fit coloring of interval graphs,” *SIAM Journal on Discrete Mathematics*, vol. 1, no. 4, pp. 526–530, 1988.

[5] A. Gy’arf’as and J. Lehel, “On-line and first fit colorings of graphs,” *Journal of Graph Theory*, vol. 12, no. 2, pp. 217–227, 1988.

[6] S. Vishwanathan, “Randomized online coloring of graphs,” in *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science*. IEEE, 1990, pp. 252–259.

[7] Y. Sakai and H. Tamaki, “Efficient graph coloring algorithms using optimization-based strategies,” in *International Conference on Learning and Optimization Algorithms: Theory and Applications*. Springer, 2018, pp. 243–254.

[8] D. Brelaz, “New methods to color the vertices of a graph,” *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[9] T. Akiba and S. Iwata, “Parallelizing coloring on massive graphs,” in *International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2010, pp. 1–12.

[10] M. Kubale, E. Bechet, F. Copty, Y. Etsion, S. Kahan, A. Lumsdaine, and M. Mundhenk, “Parallel graph coloring for many-core architectures,” in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. IEEE, 2013, pp. 1191–1202.