

Harmony by Order: Constructive Heuristics for Harmonious Graph Coloring with a Color Availability Dictionary

Tasso A. T. Pimenta

Departamento de Engenharia de Produção - Escola de Engenharia
Universidade Federal de Minas Gerais (UFMG)
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
TassoossaT842@gmail.com

Marcio Costa Santos

Departamento de Ciência da Computação - Instituto de Ciências Exatas
Universidade Federal de Minas Gerais (UFMG)
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901
marciocs@dcc.ufmg.br

ABSTRACT

The Harmonious Graph Coloring Problem (HCP) is an NP-hard graph coloring variant where each edge must be assigned a unique color pair. Despite its theoretical complexity and potential applications, there is a gap in the literature regarding effective heuristic approaches for this problem. This paper addresses this by proposing two greedy-like heuristics based on vertex ordering and a specialized color availability dictionary: the Vertex and Neighbors Coloring heuristic and the Vertex-for-Vertex Coloring heuristic. We conducted extensive experiments to evaluate their performance on various graph types, including random, bipartite, galaxy, and benchmark instances, using different vertex ordering strategies (degree-based, DFS, BFS, and random). Experimental results show that while heuristic performance converges for dense graphs, vertex ordering significantly impacts the number of colors used for sparse graphs, particularly galaxy graphs, where Lowest Degree and Random orderings often yield better solutions. Benchmark instances exhibited mixed sensitivity to ordering. The proposed heuristics successfully construct valid colorings and provide valuable insights into the impact of vertex ordering on performance for the challenging Harmonious Coloring Problem.

KEYWORDS. Coloring, harmonious Coloring, heuristic

Paper topics. Constructive heuristics, computation experiments, harmonious Coloring

1. Introduction

Vertex coloring is a fundamental concept in graph theory and combinatorial optimization, where the goal is to assign colors to vertices of a given graph, respecting a pre-defined property or constraint. When the constraint is to assign a color to each vertex such that no two adjacent vertices share the same color, the coloring obtained is said to be *proper*. The smallest number of colors required to achieve this for a given graph is called its *chromatic number*. The Vertex Coloring Problem (VCP) consists in determining its chromatic number.

The Vertex Coloring Problem is known to be NP-Hard Garey and Johnson [1979], and has been extensively studied in the literature, being also used to solve many real applications. One of the first applications, allocation of registers in compilers, was introduced by [Chaitin et al., 1981]. As other examples, we can cite timetabling [Werra, 1985], communication networks [Woo et al., 1991], train platforming [Caprara et al., 2007] and image segmentation [Gómez et al., 2007].

The Vertex Coloring Problem can have many variants, in an effort to embed or incorporate restrictions and characteristics of practical problems. In this paper we tackle the *harmonious graph coloring proper* or simply the *harmonious coloring problem* (HCP) which is a particular case of *line distinguishing colorings*. In a line distinguishing coloring every edge $uv \in E(G)$ has a unique pair of colors, more precisely, let $\{f(u), f(v)\}$ be the set of colors associated to the endpoints of an edge uv . In a line distinguishing coloring, given two distinct edges uv and zw we have $\{f(u), f(v)\} \neq \{f(w), f(z)\}$. We say that a coloring is a *harmonious coloring* Hopcroft and Krishnamoorthy [1983] if it is line distinguishing and proper. Notice that, due to the restriction that the pair of labels in each must be unique, one cannot provide a harmonious color for every number of colors Harary and Plantholt [1983], Salvi [1993]. To the smallest integer k such a given graph $G = (V, E)$ admits a harmonious coloring name its *harmonious chromatic number* and denote it by $\chi_h(G)$. The *harmonious graph coloring problem* consists in given a graph G , one must determine $\chi_h(G)$ Mitchem [1989].

Regarding the theoretical point of view, the harmonious chromatic number has been studied from different perspectives. Selvi [2015] related $\chi_H(G)$ with the chromatic index $\chi'(G)$, i.e., the smallest number of colors needed to color the edges of a graph, such way that no adjacent edges share a color. On the same direction, Harary and Plantholt [1983] showed properties that ensure a particular number to $\chi_H(G)$ and Hopcroft and Krishnamoorthy [1983] proved that the harmonious chromatic problem is NP-Hard, even for trees, the problem remains NP-Hard. Mitchem [1989] showed that the harmonious chromatic number is polynomial for some graph classes and Akbari et al. [2012] studied the problem for trees to a large degree. We may find a scarce literature of direct applications of the harmonious coloring problem. In [Selvi, 2015], the authors formulated a radio navigation problem throughout a harmonious coloring.

In the last years, some works from the point of view in operational research have arisen in this topic, specially focused in exact methods and integer formulations. Nevertheless, there is still a gap in heuristic methods in the literature, one that motivates this paper. Therefore, we present greedy-like heuristics to the problem, aiming to provide good quality solutions to instances for the problem presented in the literature.

2. The Proposed Heuristics

The heuristics proposed in this work are based on assigning colors to the vertices of the graph from a set of possible *feasible* colors. These subset of possible colors are stored in a dictionary structure that, for each color c , keeps the subset of compatible colors if it, namely the colors such that there is no pair of adjacent vertices colored with it in the current partial coloring. This allows the algorithm to efficiently manage the available colors. Therefore, the algorithm constructs a harmonious coloring by coloring each vertex respecting a pre-defined order. In what follows we describe in details the proposed method.

The **Color Availability Dictionary** is a data structure that maps each color to a list of compatible colors, allowing dynamic control over allowed color combinations during the graph coloring process. This structure supports the following operations:

- **AddColor(Dict, Color):** Adds a new color to the dictionary. The new color is made compatible with all existing colors. It is appended to the compatibility list of all current entries, and its own compatibility list is initialized with all other defined colors.
- **RemoveCombination(Dict, c_1 , c_2):** Removes the compatibility between two colors c_1 and c_2 . This function deletes each color from the other's compatibility list, ensuring they are no longer used together in future assignments.
- **CommonAvailableColors(Dict, S):** Computes the intersection of compatible colors across a set S of already assigned neighbor colors. The function returns the list of colors that remain compatible with all colors in S , or the empty set if no such colors exist.

This dictionary serves as the backbone of the coloring heuristics, enabling local and second-order constraints to be imposed in a flexible way.

The **Coloring a Single Vertex** (Algorithm 1) is a function assigns a valid color to a vertex $v \in G$ based on the current state of color assignments and the dictionary of allowed color combinations. It considers whether v has any colored neighbors. If v has no colored neighbors, use the `CommonAvailableColors`, if empty, use the `AddColor`. If v has colored neighbors, the function uses `CommonAvailableColors` to retrieve compatible options and if none are available, a new color is inserted, and this color is necessarily allowed because it is new and initially compatible with all existing colors. The selected color must not be used by any second-order neighbor of v ; this ensures that the color assigned to v does not duplicate a color pair already formed between a neighbor and its adjacent vertices and finally, the used color combinations are removed from the availability dictionary using `RemoveCombination`, in order to prevent their future use.

Algorithm 1 Coloring a Single Vertex

```

1: function COLORVERTEX( $G, v, D, vertexColor, Color$ )
2:    $N \leftarrow$  list of neighbors of  $v$  in  $G$ 
3:    $Colored_{neighborhood} \leftarrow$  list of all colored neighborhood in  $vertexColor$ 
4:   if  $Colored_{neighborhood}$  is empty then
5:     if  $D$  is empty then
6:        $Color \leftarrow Color + 1$ 
7:       ADDCOLOR( $D, Color$ ) ▷ defined in Algorithm ??
8:        $available \leftarrow$  keys( $D$ )
9:     else
10:       $available \leftarrow$  COMMONAVAILABLECOLORS( $D, C_{colored}$ ) ▷ defined in Algorithm ??
11:      if  $available$  is empty then
12:        ADDCOLOR( $D, Color$ )
13:         $Color \leftarrow Color + 1$ 
14:         $available \leftarrow$  COMMONAVAILABLECOLORS( $D, C_{colored}$ )
15:      choose any  $c \in available$  such that no second-order neighbor of  $v$  has color  $c$ 
16:       $vertexColor[v] \leftarrow c$ 
17:      for all neighbor  $n$  of  $v$  with  $n \in$  keys( $vertexColor$ ) do
18:        REMOVECOMBINATION( $D, c, vertexColor[n]$ ) ▷ defined in Algorithm ??
19:      return  $Color$ 

```

Finally we are able to present the proposed heuristics, the first of each is the **Vertex and Neighbors Coloring Heuristic** (Algorithm 2) is an heuristic that colors each vertex and all of its uncolored neighbors in a single iteration. It follows a pre-defined vertex order V_{order} . For each vertex, the vertex is colored using ColorVertex (Algorithm 1), then all the vertex neighborhood is colored using ColorVertex and isolated vertices (those with no edges) are assigned an arbitrary color (This does not change the result and does not violate the restrictions). This heuristic is aggressive in expanding the coloring frontier and promotes early use of compatible colors among neighbors.

Algorithm 2 Vertex and Neighbors Coloring Heuristic

```

1: function COLORGRAPHVANDN( $G, V_{order}$ )
2:    $D \leftarrow$  INITIALIZEDICTIONARY
3:    $vertexColor \leftarrow$  empty map
4:    $nextColor \leftarrow 0$ 
5:   for all vertex  $v$  in  $V_{order}$  do
6:     if  $v$  has at least one neighbor in  $G$  then
7:       if  $v \notin vertexColor$  then
8:          $nextColor \leftarrow$  COLORVERTEX( $G, v, D, vertexColor, nextColor$ )
9:       for all neighbor  $u$  of  $v$  do
10:        if  $u \notin vertexColor$  then
11:           $nextColor \leftarrow$  COLORVERTEX( $G, u, D, vertexColor, nextColor$ )
12:       else
13:          $vertexColor[v] \leftarrow 1$  ▷ Isolated vertices get color 1
14:   return ( $vertexColor$ )

```

The **Vertex-for-Vertex Coloring Heuristic** (Algorithm 3) is a heuristic that applies the coloring procedure only to the vertex currently under consideration in the given order, in contrast to Algorithm 2 where neighbors are also colored immediately. Neighbors are colored later when they appear in the sequence. More specifically, for each vertex it is colored using `ColorVertex` (Algorithm 1) and then, isolated vertices (those with no edges) are assigned an arbitrary color.

Algorithm 3 Vertex for Vertex Coloring Heuristic

```

1: function COLORGRAPHVANDN( $G, V_{\text{order}}$ )
2:    $D \leftarrow \text{INITIALIZEDICTIONARY}$ 
3:    $\text{vertexColor} \leftarrow \text{empty map}$ 
4:    $\text{nextColor} \leftarrow 0$ 
5:   for all vertex  $v$  in  $V_{\text{order}}$  do
6:     if  $v$  has at least one neighbor in  $G$  then
7:       if  $v \notin \text{vertexColor}$  then
8:          $\text{nextColor} \leftarrow \text{COLORVERTEX}(G, v, D, \text{vertexColor}, \text{nextColor})$ 
9:       else
10:         $\text{vertexColor}[v] \leftarrow 1$  ▷ Isolated vertices get color 1
11:   return ( $\text{vertexColor}$ )

```

3. Heuristics Evaluation and Validation

To validate and evaluate the proposed heuristics, experiments were conducted using two heuristic configurations across four types of graphs: bipartite graphs, galaxy graphs, random graphs, and benchmark graphs from the literature.

The vertex ordering strategies used as input lists for the heuristics were designed to assess their impact on performance. The following ordering criteria were tested:

- **Highest Degree:** vertices are sorted in descending order of degree;
- **Depth-First Search (DFS):** vertices are ordered according to a DFS traversal;
- **Breadth-First Search (BFS):** vertices are ordered based on a BFS traversal;
- **Lowest Degree:** vertices are sorted in ascending order of degree.
- **Random:** Random ordering

3.1. Random Graphs

Initially, the heuristics were evaluated on randomly generated graphs with vertex counts ranging from 10 to 100, increasing in steps of 10. Each set was tested under different levels of edge density (connectivity): 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90%. The random ordering was tested by generating 100 different vertex permutations for each instance, recording both the average and the best result among the 100 runs.

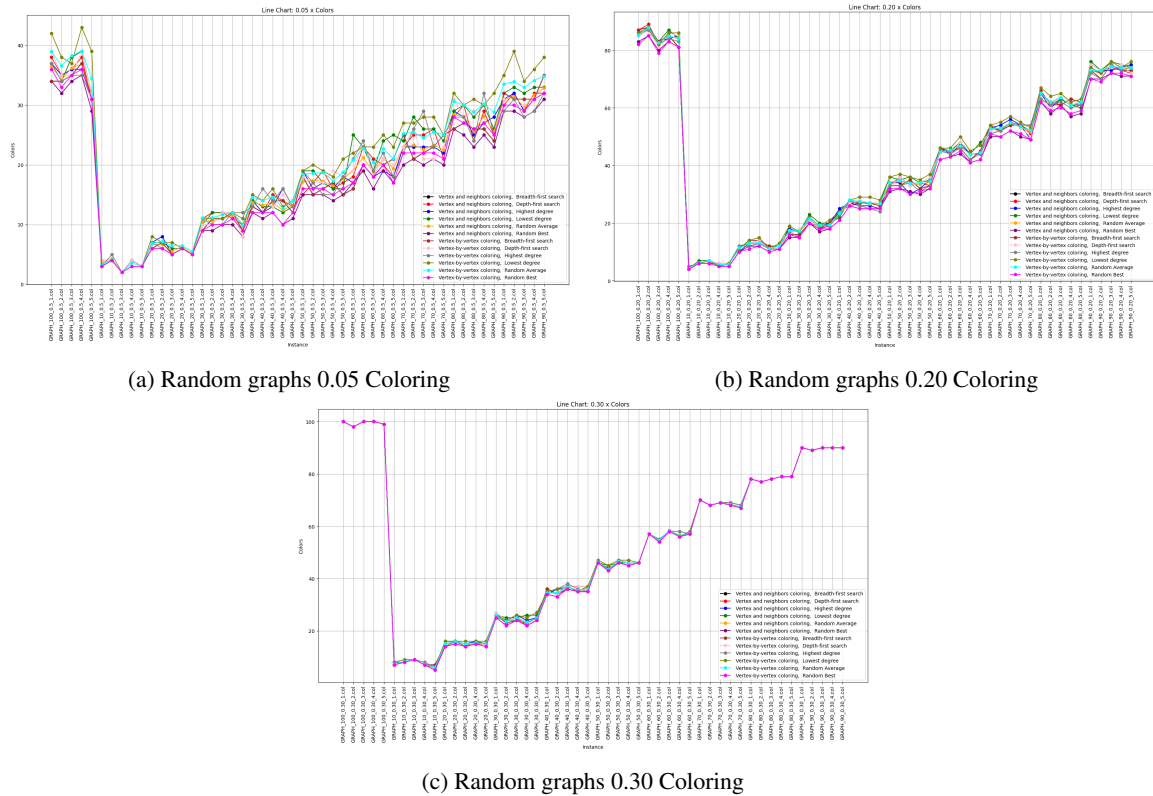


Figure 1: Comparison of heuristic performances for random graphs with 0.05, 0.20 and 0.30 of connectivity

The table below presents the average number of colors used and the variance for each connectivity level:

Connectivity	Mean Colors Used	Variance
5%	18.83	1.16
10%	27.53	1.28
20%	41.25	0.83
30%	52.06	0.04
40%	54.55	0.00
50%	54.86	0.00
60%	54.98	0.00
70%	55.00	0.00
80%	55.00	0.00
90%	55.00	0.00

Table 1: Coloring performance on random graphs with varying connectivity

A preliminary observation is that, at high levels of connectivity, the performance of different heuristics converges. This behavior is expected: as the graph approaches a complete graph (clique), the chromatic number increases.

3.2. Bipartite Graphs

Following the same analysis, bipartite graphs presented similar trends to random graphs, with relatively low variance across different connectivity levels. Table 2 summarizes the results.

Connectivity (%)	Average	Variance
20	48.05	3.65
30	63.17	2.57
40	75.70	1.65
50	83.97	1.39
60	89.31	1.10
70	92.66	0.71
80	95.00	0.45
90	96.82	0.25

Table 2: Average and variance for bipartite graphs at various connectivity levels.

3.3. Galaxy Graphs

Given their extremely sparse connectivity, galaxy graphs represent the worst-case scenarios for graph coloring heuristics. Their sparse structure makes them a true stress test for evaluating algorithm performance. Table 3 displays the average and variance results for different connectivity levels.

Connectivity (%)	Average	Variance
20	169.26	17449.78
30	226.15	33638.87
40	262.15	46788.90
50	299.59	62606.55
60	329.08	77779.98
70	362.61	97941.59
80	393.49	121468.76
90	423.17	142002.39

Table 3: Average and variance for galaxy graphs at various connectivity levels.

The results show a wide range of performance values among different heuristics, indicating that for galaxy graphs, both very good and very poor outcomes are possible depending on the heuristic and input quality.

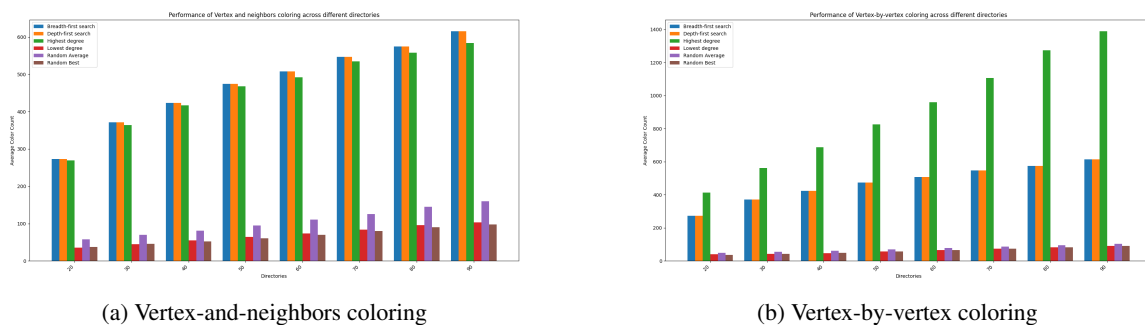


Figure 2: Comparison of heuristic performances for galaxy graphs under two different coloring strategies.: mean heuristic performance across different connectivity levels (20% to 90%).

Table 4 compare the performance of each heuristic for two coloring strategies: vertex-and-neighbors coloring and vertex-by-vertex coloring. The results for the BFS and DFS ordering

strategies yielded the same result for both heuristics.

Conn.	BFS		DFS		High. Deg.		Low. Deg.		Rng. Avg.		Rng. Best	
	(VN	VV)	(VN	VV)	(VN	VV)	(VN	VV)	(VN	VV)	(VN	VV)
20	273.29		273.29		269.12	413.21	35.42	39.67	57.52	48.31	37.75	36.96
30	371.83		371.83		364.00	562.38	44.96	42.54	69.62	55.11	45.42	42.42
40	423.58		423.58		416.71	688.50	54.88	47.00	80.92	62.30	52.04	49.12
50	474.29		474.29		468.25	825.21	64.58	58.25	95.24	69.62	60.42	56.38
60	508.12		508.12		491.88	959.50	73.92	66.42	111.18	78.27	70.33	64.96
70	546.58		546.58		534.71	1106.62	83.92	74.38	125.76	86.48	79.79	73.38
80	574.33		574.33		558.21	1274.12	95.42	82.62	145.17	95.66	90.33	83.00
90	614.92		614.92		584.12	1387.88	103.38	91.58	159.83	103.38	97.50	90.75

Table 4: Comparison of Vertex-and-Neighbors (VN) and Vertex-by-Vertex (VV) Coloring Heuristics
Performance for Galaxy Graphs

3.4. Graphs of Literature

For the benchmark graphs from the literature, there was little variation in the results across the different vertex orderings. The heuristics exhibited very similar behavior regardless of the input order. The benchmark instances used in this study were obtained from the DIMACS Graph Coloring Challenge repository DIMACS [1992].

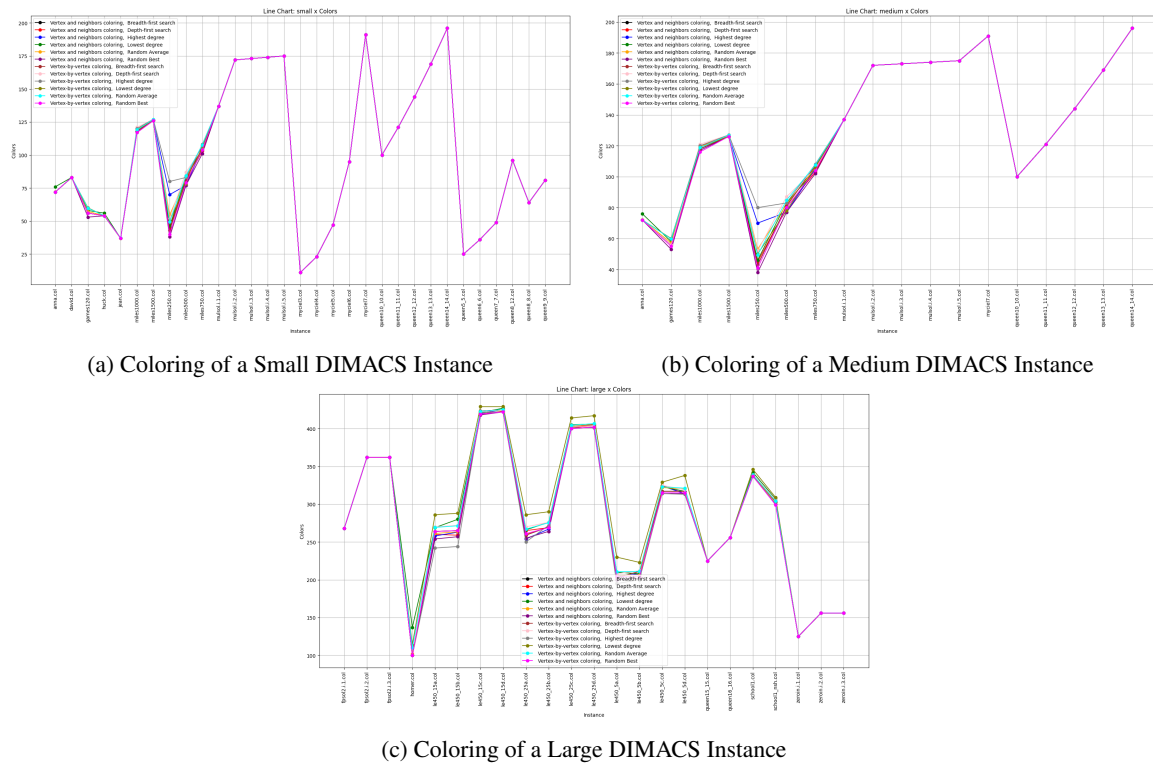


Figure 3: Coloring Results on DIMACS Graphs

The results for each graph where some variation was observed are presented in Table 5, which reports the number of colors used by two heuristics under different vertex orderings. Heuristic

VN refers to the *Vertex-by-Neighbor* approach, while Heuristic VV refers to the *Vertex-by-Vertex* approach.

The graphs for which no variation was observed across any vertex ordering or heuristic are listed in Table 6.

Instance	BFS		DFS		High Deg.		Low Deg.		Rng. Avg.		Rng. Best	
	(VN	VV)	(VN	VV)	(VN	VV)	(VN	VV)	(VN	VV)	(VN	VV)
games120	56	56	56	56	56	60	58	60	57	59	53	56
homer	100	100	100	100	100	100	137	114	102	110	100	100
huck	54	54	54	54	54	54	56	54	54	54	54	54
le450_15a	264	264	260	270	258	242	269	286	261	269	254	264
le450_15b	265	265	259	272	263	244	280	288	263	271	257	265
le450_15c	423	423	419	422	420	421	421	429	421	422	418	419
le450_15d	425	425	423	425	425	422	427	429	424	425	422	422
le450_25a	259	259	265	270	254	250	266	286	260	267	255	261
le450_25b	270	270	269	276	267	272	276	290	269	275	264	269
le450_25c	402	402	401	402	403	401	405	414	402	404	400	400
le450_25d	405	405	404	404	405	401	406	417	405	406	402	402
le450_5a	203	203	206	208	205	195	210	230	205	210	200	204
le450_5b	211	211	207	212	208	200	208	223	206	211	202	204
le450_5c	317	317	325	325	324	314	323	329	322	322	315	315
le450_5d	317	317	314	319	314	313	317	338	319	321	314	315
miles1000	118	118	120	121	118	120	118	119	119	119	117	117
miles250	43	43	45	52	70	80	46	50	54	49	38	40
miles500	81	81	80	87	77	83	83	78	82	84	77	79
miles750	107	107	103	105	105	108	104	106	106	107	101	104
school1	339	339	338	339	337	337	342	346	337	338	337	337
school1_nsh	303	303	304	306	304	300	307	309	302	304	300	299

Table 5: Coloring results for each input ordering using two heuristics

Instance	Colors	Instance	Colors	Instance	Colors
anna	72	david	83	jean	37
fpsol2.i.1	268	fpsol2.i.2	362	fpsol2.i.3	362
miles1500	126	mulsol.i.1	137	mulsol.i.2	172
mulsol.i.3	173	mulsol.i.4	174	mulsol.i.5	175
myciel3	11	myciel4	23	myciel5	47
myciel6	95	myciel7	191	queen5_5	25
queen6_6	36	queen7_7	49	queen8_8	64
queen8_12	96	queen9_9	81	queen10_10	100
queen11_11	121	queen12_12	144	queen13_13	169
queen14_14	196	queen15_15	225	queen16_16	256
zeroin.i.1	125	zeroin.i.2	156	zeroin.i.3	156

Table 6: Instances with no variation

4. Conclusion

This paper addressed the Harmonious Graph Coloring Problem (HCP), an NP-hard variant of graph coloring characterized by the constraint that each edge must be assigned a unique color pair.

Despite its theoretical complexity and potential applications, the literature presents a gap regarding heuristic approaches capable of finding good quality solutions for this problem.

To tackle this challenge, we proposed two greedy-like heuristics based on vertex ordering and the management of compatible color combinations through a specialized dictionary structure: the Vertex and Neighbors Coloring heuristic (Algorithm 2) and the Vertex-for-Vertex Coloring heuristic (Algorithm 3). These heuristics employ the `ColorVertex` function (Algorithm 1) to assign valid colors iteratively, respecting the harmonious coloring constraints by considering local and second-order neighborhood information.

We conducted extensive experiments to evaluate the proposed heuristics' performance across various graph types, including random, bipartite, galaxy, and benchmark graphs from the DIMACS challenge, utilizing different vertex ordering strategies (Highest Degree, Lowest Degree, DFS, BFS, and Random).

The experimental results demonstrated that for dense graphs, such as random and bipartite graphs with high connectivity, the choice of heuristic and vertex ordering has a diminished impact, with the number of colors used tending to converge. This suggests that the inherent structure of dense graphs imposes strong constraints that limit the variability of greedy approaches.

In contrast, for sparse graphs, particularly the galaxy graphs, the vertex ordering strategy played a crucial role, leading to significant variations in the number of colors used, as evidenced by the high variance observed (Table 3). Our findings indicate that orderings like Lowest Degree and Random often yield better results for these challenging instances compared to structural orderings like BFS or DFS (Tables 4). This highlights the importance of selecting an appropriate vertex processing order, especially when dealing with sparse graph structures.

For the benchmark instances from the literature (DIMACS), we observed that while many graphs showed little to no variation in results across different orderings and heuristics (Table 6), a notable subset of instances exhibited sensitivity to the input order (Table 5). For these instances, the relative performance of the two proposed heuristics and the effectiveness of different orderings varied, suggesting that graph-specific properties interact with the greedy strategies.

A key limitation of the proposed greedy heuristics is their dependence on the vertex ordering, which directly impacts the quality of the resulting coloring. Identifying an optimal or near-optimal ordering for any given graph remains a non-trivial challenge and is crucial for achieving the best possible performance with these methods.

As future work, we suggest exploring adaptive vertex ordering strategies that dynamically adjust the order during the coloring process based on the current state of the partial coloring. Further investigation into hybrid approaches combining these greedy heuristics with local search or other optimization techniques could also lead to improved solution quality. Expanding the evaluation to a broader range of graph types and larger benchmark instances, as well as comparing the heuristics against exact methods for smaller instances, would provide a more comprehensive understanding of their strengths and weaknesses.

In conclusion, the greedy-like heuristics presented offer a viable and efficient approach for the Harmonious Coloring Problem, successfully constructing valid colorings for diverse graph instances. While sensitive to vertex ordering, particularly in sparse graphs, the experimental evaluation provides valuable insights into their behavior and the impact of different ordering strategies, laying groundwork for further research in developing effective heuristic solutions for this challenging problem.

References

- Akbari, S., Kim, J., and Kostochka, A. (2012). Harmonious coloring of trees with large maximum degree. *Discrete Mathematics*, 312(10):1633–1637. ISSN 0012-365X. URL <https://www.sciencedirect.com/science/article/pii/S0012365X12000817>.
- Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P. (2007). Passenger railway optimization. *Handbooks in operations research and management science*, 14:129–187.
- Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E., and Markstein, P. W. (1981). Register allocation via coloring. *Computer languages*, 6(1):47–57.
- DIMACS (1992). Dimacs implementation challenges. <http://dimacs.rutgers.edu/programs/challenge/>. Accessed on: March 10, 2025.
- Garey, M. R. and Johnson, D. S. (1979). Computers and intractability. *A Guide to the*.
- Gómez, D., Montero, J., Yáñez, J., and Poidomani, C. (2007). A graph coloring approach for image segmentation. *Omega*, 35(2):173–183.
- Harary, F. and Plantholt, M. (1983). Graphs with the line distinguishing chromatic number equal to the usual one. *Utilitas Math*, 23:201–207.
- Hopcroft, J. E. and Krishnamoorthy, M. S. (1983). On the harmonious coloring of graphs. *SIAM Journal on Algebraic Discrete Methods*, 4(3):306–311.
- Mitchem, J. (1989). On the harmonious chromatic number of a graph. *Discrete mathematics*, 74(1-2):151–157.
- Salvi, N. Z. (1993). A note on the line-distinguishing chromatic number and the chromatic index of a graph. *Journal of graph theory*, 17(5):589–591.
- Selvi, M. F. T. (2015). Harmonious coloring of central graphs of certain snake graphs. *Applied Mathematical Sciences*, 9(12):569–578.
- Werra, D. (1985). An introduction to timetabling. *European journal of operational research*, 19(2):151–162.
- Woo, T.-K., Su, S. Y., and Newman-Wolfe, R. (1991). Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm. *IEEE Transactions on Communications*, 39(12):1794–1801.