Improving the Performance of Graph Coloring Algorithms through Backtracking

Sanjukta Bhowmick¹ and Paul D. Hovland²

¹ Department of Computer Science and Engineering, The Pennsylvania State University,
University Park, PA 16802

bhowmick@cse.psu.edu

² Mathematics and Computer Science Division, Argonne National Laboratory,
9700 South Cass Avenue, Argonne, IL 60439-4844

hovland@mcs.anl.gov

Abstract. Graph coloring is used to identify independent objects in a set and has applications in a wide variety of scientific and engineering problems. Optimal coloring of graphs is an NP-complete problem. Therefore there exist many heuristics that attempt to obtain a near-optimal number of colors. In this paper we introduce a backtracking correction algorithm which dynamically rearranges the colors assigned by a top level heuristic to a more favorable permutation thereby improving the performance of the coloring algorithm. Our results obtained by applying the backtracking heuristic on graphs from molecular dynamics and DNA-electrophoresis show that the backtracking algorithm succeeds in lowering the number of colors by as much as 23%. Variations of backtracking algorithm can be as much as 66% faster than standard correction algorithms, like Culberson's Iterated Greedy method, while producing a comparable number of colors.

Keywords: Graph Coloring, Backtracking.

1 Introduction

Graph coloring is used for partitioning a collection of objects into "independent" sets. Objects belonging to the same set are identified by having the same color. Objects with the same color are non-conflicting, that is, certain operations can be performed simultaneously on them.

Coloring is used in many computational and engineering applications that require identification of concurrent tasks. Some examples include register scheduling, frequency assignments for mobile networking, the evaluation of sparse Jacobian matrices, etc. Optimal coloring strategies improve parallelism. The fewer colors required to classify the objects, the more the inherent parallelism of the problem can be exploited. The holy grail of graph coloring is achieving the *chromatic number*, the smallest number of colors required to color the graph so that no two adjacent vertices have the same color. Algorithms for determining the chromatic number are NP-complete [1] and designing polynomial time heuristics to obtain quasi-optimal solutions is an active area of research.

It has been observed that for many heuristics the order in which vertices are colored significantly affects the number of colors obtained [2]. Based on this observation, we present a *backtracking* correction heuristic that mitigates the effects of a “bad” vertex ordering. The backtracking algorithm dynamically rearranges the colors assigned by the top level coloring algorithm thereby, changing the vertex ordering to a more favorable permutation. We study the performance of backtracking algorithm in conjunction with several popular coloring heuristics and compare it with another correction technique, Culberson’s iterated greedy scheme [3]. Results from our experiments on graphs obtained from molecular dynamics [4] and DNA electrophoresis [5] show that backtracking improves upon the performance of the top level heuristic as well as the iterated greedy approach. The average reduction of colors is as much as 23% (16%), compared to the original (iterated greedy) method. Most correction algorithms necessarily take more time to determine the near-optimal number of colors. We have designed a variation of backtracking that is as much as 66% faster than the iterated greedy method, while giving the number of colors within 1% of that obtained by the correction method.

The rest of the paper is arranged as follows. In Section 2 we present the mathematical description of relevant terms from graph theory and define graph coloring. We provide a brief review of some of the standard coloring heuristics in Section 3. In Section 4 we describe the backtracking algorithm. We discuss experimental results in Section 5 and present improved variations to the heuristic such as *Multilevel* and *Reverse* backtracking. Section 6 contains conclusions and discussion of our future research plans.

2 Mathematical Definitions

In this section we define some terms used in graph theory. Unless mentioned otherwise, the terms used here are as they are defined in [6].

A graph $G = (V, E)$ is defined as a set of vertices V and a set of edges E . An edge $e \in E$ is associated with two vertices u, v which are called its *endpoints*. If a vertex v is an endpoint of an edge e , then e is incident on v . A vertex u is a *neighbor* of v if they are joined by an edge. The *degree* of a vertex u is the number of its neighbors. A *walk*, of length l , in a graph G is an alternating sequence of $v_0, e_1, v_1, e_2, \dots, e_l, v_l$ vertices and edges, such that for $j = 1, \dots, l$; v_{j-1} and v_j are the endpoints of edge e_j . An *internal vertex* is a vertex that is neither the initial nor final vertex in this sequence. A *path* is a walk with no edges or internal vertices repeated. Two vertices are said to be *distance- k neighbors* if the shortest path connecting them has length at most k [7].

A *vertex-coloring* of a graph $G = (V, E)$ is a function $\phi : V \rightarrow C$ from the set of vertices to a set $C = \{1, 2, \dots, n\}$ of “colors”. A *distance- k coloring* of a graph $G = (V, E)$ is a mapping $\phi : V \rightarrow \{1, 2, \dots, n\}$ such that $\phi(u) \neq \phi(v)$, whenever u and v are distance- k neighbors. The least possible number of colors required for a distance- k coloring of a graph G is called its *k -chromatic number* [7].

3 Review of Some Coloring Algorithms

In this section we provide an overview of some standard coloring algorithms. It has been observed that the order in which vertices are colored is an important parameter in

lowering the number of colors. Consequently, many coloring heuristics focus on finding an efficient vertex ordering. Some apply well known graph traversal methods like the *depth-first* search [6] while others focus on orderings based on the degree of the vertices or the number of colored neighbors. Some examples of the later category include the *largest first* [8] ordering where the vertices are arranged in non-increasing order of their degrees and the *smallest last* [9] ordering which dynamically orders the vertices such that the last vertex in the sequence is one with the minimum degree in the subgraph induced by the yet uncolored vertices. In the *incidence degree* [10] ordering, the vertex with the maximum number of colored neighbors is the next one to be colored. The effectiveness of these heuristics depend on the underlying graph structure. The results can be improved by using correction algorithms such as *Culberson's Iterated Greedy* method [3]. In this approach, once the initial algorithm has been applied, the iterated greedy method rearranges the vertices in decreasing order of color, and re-colors them. Culberson's method guarantees that the reordering does not increase the number of colors.

4 The Backtracking Correction Heuristic

The *backtracking* correction heuristic is based on dynamically reassigning colors amongst already colored vertices in order to restrict the number of colors within an user specified minimum. The heuristic is implemented as follows; the user specifies a coloring threshold set to a lower bound on the chromatic number. The backtracking heuristic is invoked whenever this threshold is exceeded. It is easy to see that when backtracking is called there is only one vertex, designated as the *last-vertex*, that is assigned a color higher than the threshold. Evidently, the rest of the colors up to the threshold would have been used to color the neighbors of the last vertex. These colors form the *acceptable set* of colors. The last-vertex is temporarily assigned a *pseudo-color* from the acceptable color set. The backtracking algorithm tries to determine whether there is an alternate assignment of colors from the acceptable set to the neighboring vertices that would allow the last-vertex to retain the pseudo-color and prevent conflicts. If such an assignment is found, then we have a coloring within the limits of the threshold. If no such arrangement can be obtained for any color from the acceptable set, the last vertex is assigned its original color and the threshold is increased by one.

Pseudocode for Backtracking Heuristic

```

Set threshold to T
For all vertices v
    Color vertex v with initial coloring algorithm
    pseudocolor[v]=color[v]
    If color[v]>T
        For all colors c;  $1 \leq c \leq T$ 
            Set pseudocolor[v] to c
            Set fail to FALSE
            For all neighbors n of v,
                If color[n]=c

```

```

        Reassign pseudocolor[n] to avoid conflicts;
        If pseudocolor[n]>T; Set fail to TRUE;
        Break;
    If fail is FALSE
        Re-coloring is successful
        Break
    Else continue for next color
If fail is FALSE (Alternative coloring assignment
found)
    For all vertices v; set color[v]=pseudocolor[v]
Else
    For all vertices v; set pseudocolor[v]=color[v]
    Increase T by 1

```

5 Performance of Backtracking Heuristic

We report on the performance of the backtracking algorithm on two test suites each containing six matrices. We applied the coloring algorithms discussed in Section 3 to the adjacency graphs corresponding to these matrices. The first set obtained from molecular dynamics [4], consists of a group of graphs with fixed vertices (11414) and gradually increasing number of edges. The second set obtained from the Florida Sparse Matrix Collection [5], representing DNA electrophoresis, consists of graphs whose size increases with both vertices and edges.

We used the following ordering heuristics; Natural Ordering (N), Smallest Last (S), Largest First (L), Incidence Degree (I), and Depth First Ordering (D). For each heuristic we conducted three sets of experiments using: i) only the heuristic, ii) Culberson's Iterated Greedy method [3], with the current heuristic in the first iteration and iii) the heuristic with the Backtracking algorithm. Our experiments include results for both distance-1 and distance-2 coloring objectives. The threshold for distance-1 coloring was set to 3 and the threshold for distance-2 coloring was set to the minimum degree of the graph.

5.1 Reduction of Colors

The results summarized in Tables 1 and 2 demonstrate that backtracking can significantly reduce the number of colors. The number of colors obtained is lower than that given by the iterated greedy algorithm. The reduction is higher for distance-1 coloring (maximum reduction of 23%) than for distance-2 coloring (maximum reduction of 18%). This is to be expected, since coloring vertices based on distance-2 neighbors requires fulfilling more constraints, thus reducing the possibility of color reassignments.

5.2 Running Time

The time taken to color a graph is proportional to its size. Figure 1 plots the time taken to distance-1 color the two sets of graphs. The results show that though the execution time of backtracking is competitive with the iterated greedy heuristic for smaller

Fig. 1. Comparison of the time taken between Depth First Search (DF), DF using Culberson's Iterated Greedy Algorithm and DF using Backtracking for distance-1 coloring. The left-hand side figure represents graphs from molecular dynamics and the right-hand side figure represents graphs from DNA-electrophoresis. Time is given in seconds.

graphs, it gets much larger as the size of the graph increases. The running time is worse for distance-2 coloring and with particular bad combinations of vertex ordering and threshold for dense matrices the execution time can go up to as much as 77 times the time taken by the top-level heuristic. The time required to backtrack can be reduced by increasing the level of the threshold. Since backtracking does not start until the threshold is reached, a judicious selection can significantly decrease the time, as shown in Figure 2, without compromising in the number of colors.

Fig. 2. Comparison of time taken to color a molecular dynamics graph with Natural (N) ordering, Iterated Greedy Method (G) and Backtracking (B) with different thresholds, given in parenthesis. The values on top of the bars give the number of colors obtained. Time is given in seconds.

5.3 Improving Backtracking

Improvements to backtracking can include i) reduction of colors and ii) reduction of execution time. We will explore the first option with respect to distance-1 coloring and the second option with respect to distance-2 coloring.

Multilevel Backtracking: Multilevel backtracking reduces the number of colors by recursively invoking the correction heuristic. For example, if in course of backtracking a neighbor of the last-vertex is colored higher than the threshold, we use multilevel backtracking to explore further reassignments to the vertices adjacent to the neighbor in order to obtain limit the colors within the acceptable set. Figure 3 and Table 1 summarizes the number of colors for distance-1 coloring of the representative graphs. The results show that 2-level backtracking (recursion used once) can reduce the number of colors by 10% compared to non-recursive backtracking.

Fig. 3. Number of colors required by the graphs for distance-1 coloring. The groups from left to right represent different top level heuristics, Natural Ordering (N), Smallest Last (S), Largest First (L), Incidence Degree (I), and Depth First Ordering (D). Suffix (G) represents the coloring obtained by the iterated greedy algorithm, suffix (B) represents the coloring obtained by backtracking and suffix (2) represents 2-level backtracking. The left-hand side figure represents graphs from molecular dynamics and the right-hand side figure represents graphs from DNA-electrophoresis.

Reverse Backtracking: Reverse backtracking is used to reduce the execution time of the backtracking algorithm. In this heuristic the vertices are first colored using a top level algorithm and then backtracking is applied to see if it is possible to re-color *all* the vertices of the the highest color to a lower color. The process is continued until a lower color cannot be assigned. This algorithm is similar to the iterated greedy algorithm in that the vertices are grouped according to color after the initial coloring is complete. However instead of re-coloring all the vertices again as in the iterated greedy algorithm, reverse-backtracking re-colors only the vertices with the highest colors (and their neighbors as required). Consequently reverse backtracking has a lower running time than the iterated greedy heuristic. Figure 4 compares the execution time for distance-2 coloring with respect the most expensive algorithm (depth-first search). The results show that reverse backtracking is faster by as much as 66% compared to the iterative greedy algorithm. The performance of these algorithms with respect to the number of colors is summarized is Figure 5 and Table 2. For most algorithms, the number of colors obtained by reverse backtracking is competitive to that obtained by the iterated greedy method.

Table 1. Number of colors required by the graphs for distance-1 coloring. The groups from left to right represent different top level heuristics, Smallest Last (S), Largest First (L), Incidence Degree (I), and Depth First Ordering (D). Suffix (G) represents the coloring obtained by the iterated greedy algorithm, suffix (B) represents the coloring obtained by backtracking and suffix (2) represents 2-level backtracking.

Graphs	S	S-G	S-B	S-2	L	L-G	L-B	L-2	I	I-G	I-B	I-2	D	D-G	D-B	D-2
V=11K E=15K	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
V=11K E=64K	8	8	8	8	9	9	9	8	9	9	9	8	11	9	9	8
V=11K E=130K	14	14	14	13	16	16	15	14	15	14	14	13	18	16	15	14
V=11K E=412K	34	34	32	32	42	40	37	35	34	34	33	32	45	42	39	37
V=11K E=1.6M	115	115	110	109	137	135	124	117	117	117	112	109	164	156	142	130
V=11K E=2.6M	183	183	176	171	218	216	187	181	183	182	177	173	266	241	225	201
V=37 E=196	4	5	4	4	5	5	4	4	5	5	4	4	5	5	5	4
V=93 E=692	6	5	5	5	6	7	5	5	7	6	5	5	6	6	5	5
V=1K E=9K	8	7	7	7	8	8	7	6	8	7	7	7	8	7	8	7
V=3K E=38K	9	8	8	8	9	9	8	8	10	9	9	8	11	9	9	8
V=39K E=520K	10	10	10	9	13	11	10	9	12	11	10	10	14	12	11	10
V=130K E=1.9M	12	11	10	9	13	12	10	10	13	11	10	10	14	12	12	11

Fig. 4. Number of colors required by the graphs distance-2 coloring. The groups from left to right represent different top level heuristics, Natural Ordering (N), Smallest Last (S), Largest First (L), Incidence Degree (I), and Depth First Ordering (D). Suffix (G) represents the iterated greedy algorithm, suffix (B) represents backtracking and suffix (R) represents reverse backtracking. The left-hand side figure represents graphs from molecular dynamics and the right-hand side figure represents graphs from DNA-electrophoresis.

Reverse Backtracking Heuristic

```
Color all vertices
Set C to maximum number of colors
While TRUE
  For all vertices colored with C;
    Apply backtracking with threshold C-1
  If all vertices can be recolored; set C to C-1
  Else break
```

Table 2. Number of colors required by the 12 graphs for distance-2 coloring. The groups from left to right represent different top level heuristics, Smallest Last (S), Largest First (L), Incidence Degree (I), and Depth First Ordering (D). Suffix (G) represents the iterated greedy algorithm, suffix (B) represents backtracking and suffix (R) represents reverse backtracking. Due to space constraints the colors for the graph with vertex=11k and edges=2.6M are given in the order of 10^3 .

Graphs (V:E)	S	S-G	S-B	S-R	L	L-G	L-B	L-R	I	I-G	I-B	I-R	D	D-G	D-B	D-R
11K:15K	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
11K :64K	22	22	22	22	26	24	24	24	24	23	22	23	29	26	24	26
11K:130K	47	47	45	47	57	56	52	55	50	49	45	48	65	59	55	58
11K:412K	152	150	144	150	183	178	157	170	152	152	148	151	217	198	184	200
11K:1.6M	617	609	585	608	654	651	617	645	615	615	596	610	862	771	710	789
11K:2.6M	1.02	1.01	.97	1.01	1.07	1.07	1.01	1.05	1.01	1.00	.98	1.00	1.39	1.22	1.14	1.24
37:196	11	11	10	11	10	10	10	10	11	11	11	11	12	12	11	11
93:692	19	18	17	18	17	17	17	17	17	17	17	16	19	18	17	18
1K:9K	30	28	27	29	30	29	27	28	31	30	28	29	31	31	28	29
3K:38K	39	38	36	39	42	41	38	38	44	42	37	41	48	43	40	43
39K:520K	62	62	56	59	67	67	61	64	66	63	58	62	81	82	68	70
130K :1.9M	68	66	60	64	73	65	72	68	72	71	64	68	87	77	73	78

Fig. 5. Comparison of the time taken between Depth First Search (DF), DF using Culberson’s Iterated Greedy Algorithm and DF using Reverse Backtracking for distance-2 coloring. The left-hand side figure represents graphs from molecular dynamics and the right-hand side figure represents graphs from DNA-electrophoresis. Time is given in seconds.

5.4 Comparison with Results from Integer Programming

We can measure the effectiveness of backtracking by comparing the results with known optimal colorings. We used integer programming to find the optimal number of colors for some graphs of small size. The results provided in Table 5.4 demonstrate that we can also obtain optimal coloring within a few levels of backtracking. The threshold was set to the minimum nonzeros per column of the corresponding column intersection matrix.

Table 3. Number of colors for distance-1 coloring. The columns from left to right represent coloring using Natural ordering, Integer Programming, and Natural ordering with Backtracking. The number of levels of backtracking is given in parenthesis. The IP execution for graphs with E=1555 and E=3925 could not be completed within the time bounds.

Graph	Natural	IP	BT (Levels)
V=12 E=59	11	11	11 (1)
V=12 E=32	8	8	8 (1)
V=60 E=1555	50	between 50 and 45	50 (1)
V=20 E=115	10	10	10 (1)
V=10 E=45	10	10	10 (1)
V=14 E=77	8	7	7 (2)
V=72 E=472	8	6	6 (2)
V=68 E=2074	51	51	51 (1)
V=100 E=3925	49	unknown	40 (1)

6 Discussion and Future Work

We have described a backtracking algorithm and shown that it is indeed successful in reducing the number of colors. The price for achieving a smaller number of colors is an increase in the time required to compute the coloring. The execution time is generally higher for dense matrices or coloring problems with more constraints, such as distance-2 coloring. This is to be expected, since as the number of neighbors increase backtracking has more vertices to search. Backtracking has several provisions for lowering the execution time based on user specific requirements, such as by varying the threshold, or invoking backtracking only at specific intervals. Reverse backtracking is also effective in reducing the computing time and the number of colors given by this method is close to the original backtracking for distance-2 coloring. The trade-off between the reduction of colors and the execution time of the algorithm depends on the purpose of the coloring. If the underlying application uses the same partition multiple times then an upfront large cost to obtain a near-optimal partition is justified.

We have observed that the performance of backtracking is largely dependent on the top-level heuristic. One of our research goals, therefore, is to design more efficient variations of backtracking to match the top-level coloring techniques. Our other future plans include application of backtracking in parallel coloring algorithms.

Acknowledgments. This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational Technology Research, U.S. Department of Energy under Contract W-31-109-Eng-38. The idea for the backtracking algorithm was inspired by reading the excellent review on graph coloring algorithms by Assefaw Gebremedhin, Fredrik Manne and Alex Pothen [7]. We are also grateful to Assefaw Gebremedhin and Rahmi Aksu for letting us use their graph coloring software. Our implementation of the backtracking algorithm was built on top of this software. We would also like to thank Sven Leyffer for his assistance in using integer programming to find optimal colorings for several small matrices. We thank Gail Pieper for proofreading a draft of this paper.

References

1. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York (1979)
2. Siek, J.G., Lee, L., Lumsdaine, A.: *Boost Graph Library, The: User Guide and Reference Manual*. Addison Wesley Professional, Reading (2001)
3. Culberson, J.C.: *Iterated Greedy Graph Coloring and the Difficulty Landscape*. Technical Report (1992)
4. Carloni, P.: PDB Coordinates for HIV-1 Nef binding to Thioesterase ii, <http://www.sissa.it/sbp/bc/publications/publications.html>
5. Davis, T.: *University of Florida Sparse Matrix Collection* (1997), <http://www.cise.ufl.edu/research/sparse/matrices>
6. Gross, J.L., Yellen, J.: *Handbook of Graph Theory and Applications*. CRC Press, Boca Raton (2004)
7. Gebremdhin, A., Manne, F., Pothén, A.: What color is your Jacobian? Graph coloring for Computing Derivatives. *SIAM Review* 47, 629–705 (2005)
8. Welsh, D.J.A., Powell, M.B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. *Computer J.*, 85–86 (1967)
9. Matula, D.W.: A min-max theorem for graphs with application to graph coloring. *SIAM Review*, 481–482 (1968)
10. Coleman, T., Moré, J.J.: Estimation of Sparse Jacobian Matrices and Graph Coloring Problems. *SIAM Journal of Numerical Analysis* 20(1), 187–209 (1983)