A Survey on Heuristic Graph Coloring Algorithm

Akanksha Gupta[1] Harish Patidar[2]
[1]M.Tech Student [2]Assistant Professor
[1,2]Department of Computer Science and Engineering
[1,2]Lakshmi Narain College of Technology, Indore

Abstract— The problem of graph coloring is to assign color to all the vertices such that no neighbor vertices have same color. This is known as vertex coloring. Graph coloring problem belongs to the class of NP-hard problem. There are many real world applications in which the concept of graph coloring is used. There are certain approaches to implement the graph coloring algorithms. This paper gives certain heuristic graph coloring algorithms comparison in term of runtime and number of colors used.

Key words: Graph Coloring Problem, Approximate Algorithm, Heuristic Algorithm

I. INTRODUCTION

The graph coloring is very well-known problem with many real world applications such as Frequency Assignment [1], Register Allocation [2], Air Traffic Flow Management [3] and Examination Timetable Scheduling [4]. The graph coloring problem is defined as follow. Let G = (V, E) be the large undirected graph where V is set of vertices and E is set of edges. The edges are of the form (i, j) where i,j ∈ E. The problem of graph coloring is to assign color to each such that no two vertexes get same color. Finding optimum solution for graph coloring problem is well known NP-Hard class problem [5].

Algorithms of graph coloring problem can be divided into two classes: exact algorithms and approximate algorithms. Approximate algorithms are algorithms used to find approximate solutions to optimization problems. Heuristic parallel algorithms are the approximate algorithms which color the vertices of the graph in parallel to reduce the coloring speed and alto reduce the number of colors used to color the graph. In a parallel application, graph coloring is performed by partition the work associated with the vertices into independent subtasks such that the subtasks can be performed parallel. Depending on the quantity of work associated with each vertex, there are basically two coloring strategies one can follow. In the first strategy the focus is on minimizing the number of colors, whereas in the second the focus is on speed.

II. BACKGROUND

The problem of sequentially coloring an arbitrary graph has been studied extensively [6]. A 4-coloring is known to exist for any planar graph, but non-planar graphs require a larger amount of colors. Finding a coloring of a graph using the minimal number of colors is known to be an NP Hard problem [5]. A simpler problem is to color the graph using a small number of colors, not necessarily the minimum. A number of sequential polynomial-time algorithms exist for this problem, which use relatively few colors and have good bounds on computational complexity.

The Graph Coloring problem is extensively studied for solution using parallel algorithms. Luby's MIS algorithm finds independent set of vertices in parallel [7]. Luby's MIS algorithm give random number to all uncolored vertex in every iteration of algorithm. Jones-Plassman improved this approach and instead of calculating random number for each vertex they proposed to calculate random numbers in beginning only and use it throughout the program [8].

Research shows that some well-known sequential graph coloring algorithms, the Largest- Degree-First algorithm [9] and the Smallest-Degree-Last algorithm [10], can be readily parallelized. By comparing their performances with Luby's Maximal Independent Set algorithm and the Jones-Plassmann algorithm certain results shows that heuristic algorithms can give better performance. Other good sequential algorithms, notably the Saturation-Degree-Ordering [6] and Incidence-Degree-Ordering [11] algorithms, are not well-suited to parallelization.

E G Boman et al. proposed Parallel Graph Coloring Algorithm for Distributed Memory Computers [12]. In his work, author partition and distribute graph among processors and every processor takes responsibility of coloring the received partition. The coloring task is done in parallel with multiple iterations and each iteration processors color the graph, communicate for conflict and resolve it. This approach do fast coloring but does not gives optimal solution.
This paper considers only algorithms that will work for any (possibly non-planar) graph.

Buhua Chen, Bo Chen, Hongwei Liu, Xuefeng Zhang proposed A Fast Parallel Genetic Algorithm for Graph Coloring Problem Based on CUDA [13]. The algorithm is based on Compute Unified Device Architecture (CUDA). The initialization, crossover, mutation and selection operators are compute parallel in threads.

III. APPROXIMATE ALGORITHM

Approximation algorithms are algorithms used to find approximate solutions to optimization problems. Approximation algorithms are often NP -hard problems. This technique does not guarantee the best solution. The aim of an approximation algorithm is to come as close as possible to the optimum solution in a polynomial time. Examples of approximate algorithm are knapsack problem and TSP shortest path problem. Approximate algorithms are classified into 4 classes' constructive heuristics [14], local search heuristics [15], and metaheuristics and other heuristic algorithms.

IV. HEURISTIC ALGORITHM

The word heuristic is used for algorithms which find solutions among all possible solutions, but the solution found will be best is not sure, therefore they are considered as approximately and not exact algorithms. These algorithms, usually find a solution near to the best one and they find it fast and easily. Heuristic algorithms are simple and easy to implement but they do not find the optimal solution. These algorithms are used when a feasible solution is required

rapidly. Well known examples of heuristic algorithm are Traveling Salesman Problem and Knapsack Problem. This paper, compares the parallel heuristic graph coloring algorithms such as: Maximal Independent Set [7], Jones-Plassmann [8], Largest Degree First [9], Smallest Degree Last [10], Local Minima-Maxima First [16], Local Largest Degree First [16] and Local Smallest Largest Degree First[16].

A. Maximal Independent Set:

The Maximal Independent Set (MIS) algorithm [7] colors the graph by repeatedly finding the largest possible independent set of vertices in the graph. All vertices in the first such set are given the same color and these vertices are removed from the graph. The algorithm then finds a new MIS and gives these a second color, and continues finding maximal independent sets and colors them until all vertices have been colored.

1) Assign all the uncolored vertex value to a variable U
2) Execute algorithm in parallel till U>0
3) Choose an independent set I from U
4) Color all the vertices of set I
5) Remove the vertices of set I from U

U=U-I

End

Luby has described a parallel version of the MIS algorithm [7]. Luby proposed a Monte Carlo method for making the independent set in parallel. The maximal independent sets are found by giving a weight to each vertex. The weights chosen by Luby are any integer value. An independent set can be constructed in parallel by choosing all vertices whose weights are larger than any of their neighbors in the subgraph.

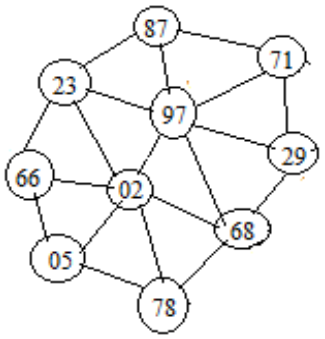
The parallel MIS algorithm for graph coloring uses the basic method shown in Fig. 1, with a maximal independent set being constructed in parallel at each step using Luby's method. The coloring is done by giving each MIS a different color.

B. Jones-Plassmann:

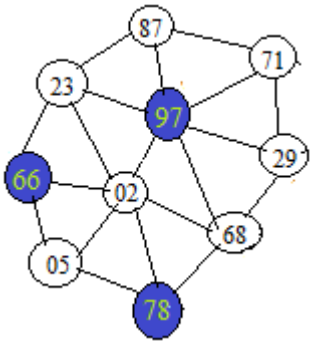
Jones and Plassmann (J-P) recently described a parallel coloring algorithm which is an improvement over the parallel MIS algorithm [8]. They pointed out that it is not necessary to create a new random value of the vertices every time when an independent set is calculated. A set of unique random weights can be constructed at the starting and used all over the algorithm. This can be done by assigning random numbers to each of the vertices and using the unique vertex number to resolve a conflict in the unlikely event of neighboring vertices getting the same random number.

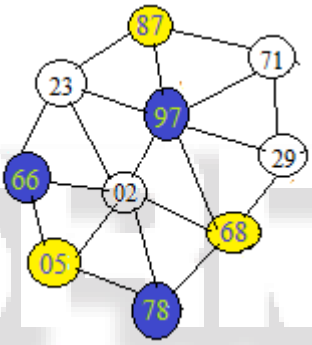
It just finds an independent set in parallel using Luby's method of choosing vertices whose weights are maxima among all its neighbors. In this the vertices are colored individually using the smallest color that has not already been assigned to a neighboring vertex. This procedure is repeated using the standard method shown in Figure 1 until the entire graph is successfully colored.

An example of Jones-Plassmann coloring can be seen in Fig. 2.

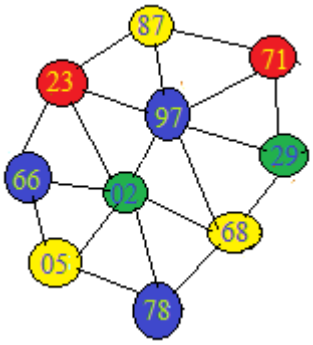


Assigned Random Number





Each uncolored vertex looks at its uncolored neighbors and if it has the highest number gets colored with the lowest available color, and so on.



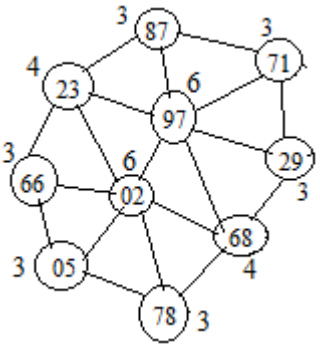
Final coloring

Fig. 2: Coloring of the vertices during the Jones-Plassmann Algorithm.

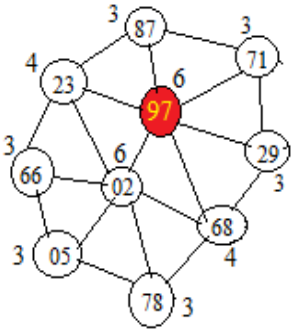
C. Largest-Degree-First:

The Largest-Degree-First algorithm (LDF) [9] can be parallelized using a very similar method to the Jones-Plassmann algorithm. The only difference is that instead of using weights to create the independent sets, the degree of the vertex is used in the induced subgraph. Random numbers are used to resolve conflicts between neighboring vertices having the same degree. In this method, vertices are not colored in

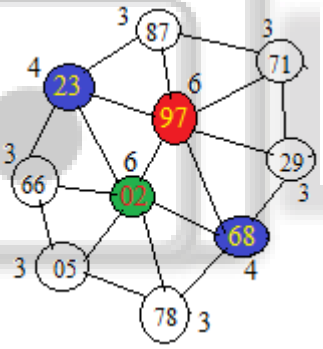
random order, but rather in order of decreasing degree, with those of largest degree being colored first.



Randomly assigned numbers with degree of each vertex



Each vertex looks at its neighbors and if it has the highest degree is then assigned a color, ties are broken by the highest random number



Algorithm continues in this way

Fig. 3: Coloring of The Vertices During the Largest-Degree-First Algorithm

This approach uses fewer colors than the Jones-Plassmann algorithm. A vertex with i colored neighbors will require at most color i+1. The Largest-Degree-First algorithm aims to keep the maximum value of i as small as possible throughout the computation, so that there is a better chance of using only a small number of colors.

An example of the Largest-Degree-First coloring algorithm can be seen in Fig. 3.

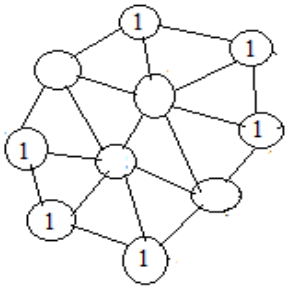
D. Smallest-Degree-Last:

The Smallest-Degree-Last algorithm (SDL) [10] tries to improve upon the Largest-Degree-First algorithm by using a more sophisticated system of weights. In order to achieve this, the algorithm operates in two phases, a weighting phase and a coloring phase.

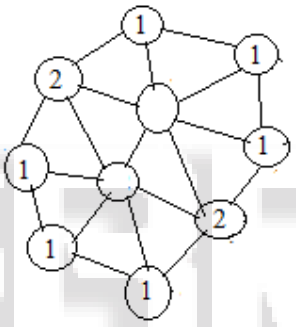
The weighting phase starts by finding all vertices with degree equal to the smallest degree d present in the graph. These are assigned the current weight and removed from the graph, which changes the degree of their neighbors. The algorithm repeatedly removes vertices of degree d, assigning successively larger weights at each iteration. When there are no vertices of degree d left, the algorithm looks for vertices of degree d+1. This continues until all vertices have been assigned a weight.

After the weight are assigned, the coloring starts from the highest value of weight and working backwards. In the coloring phase each vertex look at its uncolored neighbors and when it finds that it has the highest weight (conflicts are resolved by a random number); it colors itself using the lowest available color in its neighborhood.

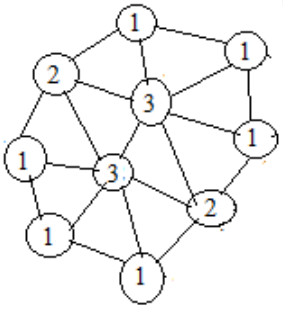
A simple example of the weighting algorithm is shown in Fig. 4, with the corresponding coloring shown in Fig. 5.



Node of degree 3 are given weight 1

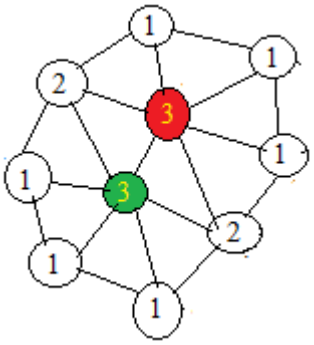


Nodes of degree 2 (ignoring 1's) are assigned weight 2.

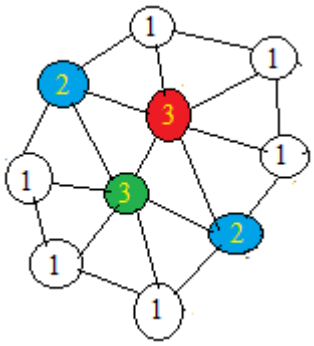


Final weight

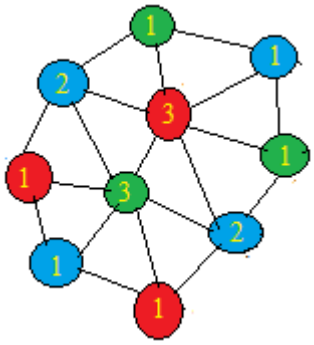
Fig. 4: Weighting of the vertices during the Smallest-Degree-Last Algorithm.



Nodes of weight 3 are colored



Nodes of weight 2 are then colored



Final coloring

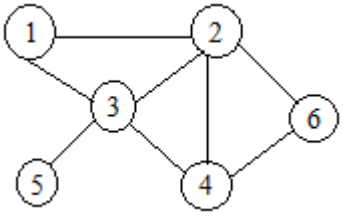
Fig. 5: Coloring of the vertices during the Smallest-Degree-Last Algorithm.

E. Local Smallest-Largest Degree First:

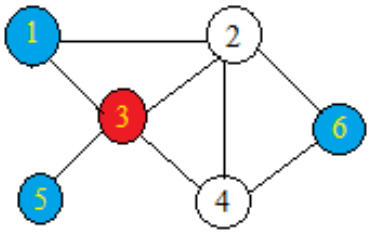
According to the Nishant M Gandhi and Rajiv Misra [14] Local Minima-Maxima First algorithm (LMMF) follow the greedy approach and based on the maximum and minimum values of vertexid between neighbors. In this algorithm vertices with minimal vertexid and maximal vertexid are selected at the same time and colored with two different colors.

Local Largest Degree First (LLDF) also follows the greedy approach but it is based on the degree of vertex. In this the vertex with the highest degree is selected and colored. In case of the same degree the tie is broken with the highest vertexid.

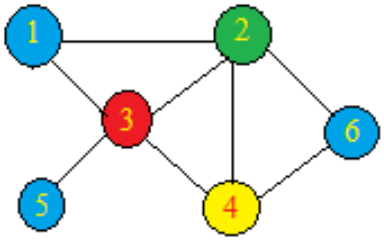
Local Smallest-Largest Degree First algorithm (LSLDF) [14] is modified and improved version of previous algorithm. In this algorithm to create two independent set at a time using degree of vertex and color them with two different colors. For remaining uncolored vertices, we select two sets of vertices one with smallest degree and one with largest degree at a same time. In the conflict of same vertex degree of neighbors, the vertexid is used to break the tie. In case of tie, vertex with larger vertexid is considered for set of largest degree vertices and vertex with smallest vertexid is considered for set of smallest degree vertices. This algorithm reduces the number of steps and gives better runtime performance than the previous one.



Step=0



Step=1



Step=2

Fig. 6: Example: Local Smallest-Largest Degree First

Example: Fig. 6 shows the examples of graph coloring using Local Smallest-Largest Degree First algorithm. The graph is initialized in Step 0. The Step 1, gives vertex set {1,5,6} and {3} as local smallest degree and local largest degree vertices respectively among their neighbors, the ties are broken with the value of vertexid. Those two sets are colored differently. In the next Step, vertex set {2} and {4} is colored.

V. RESULTS

The algorithms were tested on meshes with from 256 to 16384 vertices. The Jones-Plassmann (J-P) and Largest Degree First (LDF) algorithms are significantly faster than the MIS and SDL algorithms. The MIS algorithm was generally slower than J-P and LDF, with the Smallest-Degree-Last (SDL) being slowest of all.

In terms of the number of colors used the SDL was the most powerful. The LDF algorithm was slightly worse and the J-P and MIS algorithm were one color worse than LDF. The LDF, J-P and MIS algorithms all used increasing numbers of colors as the graph size increased. When these algorithms are tested on WILL199 of order 199 than J-P and LDF take 0.21 seconds and 0.26 seconds respectively. Whereas MIS take 0.28 seconds which is slower then J-P and LDF, SDL take 0.48 seconds being slowest from all. In terms of number of color used SDL use 7.0 average number of color which was the most powerful among all algorithms. The LDF use 8.0 whereas J-P and MIS uses 8.4 and 8.3 colors respectively.

The table 1 shows how the algorithm Local Minima-Maxima First, Local Largest Degree First and Local Smallest-Largest Degree First compared in terms of number of colors used and also runtime of each algorithm for different datasets given in [14] . The Local Minima-Maxima First performs almost equal in this parameter while Local Largest Degree First and Local Smallest-Largest Degree First perform significantly better than rest two algorithms in major datasets. On the parameter of runtime Local Smallest-Largest Degree First and Local Minima-Maxima First performance better than rest two algorithms.

Dataset	LMMF		LLDF		LSLDF	
	No. of colors	Runtime (in Seconds)	No. of colors	Runtime (in Seconds)	No. of colors	Runtime (in Seconds)
DBLP	232	97	116	198	116	82
Texas Road Network	344	129	123	119	124	92
Internet Topology	1586	1330	484	4423	484	681
Eron Email	394	83	156	103	156	70
Youtube	704	420	261	881	262	193
Amazon	32	50	23	62	24	57

Table 1: Comparisons of Lmmf, Lldf and Lsldf

VI. Conclusion

The LDF algorithm performs well among all the algorithms. The coloring time required remains lower compared to many of the other algorithms even as the size of the problem increases. In particular the SDL and MIS algorithms require more communication in each pass, which is responsible for their poor performance at large graph sizes and with more parallel processors. Even noting that the communication required by the LDF algorithm is equal to that required by the J-P algorithm, it consistently performs better. The LDF algorithm is easy to implement and to understand, which makes it relatively easy to incorporate into many parallel architecture that may be written at some point in the future.

Local Minima-Maxima First, performed better in terms of computation time but worst in terms of number of colors. The heuristic algorithm, Local Largest Degree First, performed better in terms of number of colors but it take more computation time than rest algorithms. The Local Smallest-Largest Degree First algorithm performed overall best among all the algorithms.

As Local Smallest-Largest Degree First algorithm is not optimal, there is lot of scope in improvement of distributed graph coloring algorithms. The given algorithms follow greedy approach. There is also scope to develop algorithms using different approach like heuristic parallel algorithms.

References

[1] J. Riihijarvi, M. Petrova, and P. Mahonen, "Frequency allocation for wlans using graph coloring techniques.," in WONS, vol. 5, pp. 216–222, 2005.
[2] Werra DD, Eisenbeis C, Lelait S, Marmol B. " On a graph-theoretical model for cyclic register allocation." iscrete Applied Mathematics 1999; 93(2- 3):191-203.
[3] Nicolas Barnier and Pascal Brisset "Graph Coloring for Air Traffic Flow Management" in École Nationale de l'Aviation Civile, 7, avenue Édouard Belin, BP 4055, 31055 Toulouse Cedex 4, France, Annals of Operations Research 130, 163–178, 2004
[4] Nasser R. Sabar ,Masri Ayob ,Rong Qu , Graham Kendall " A graph coloring constructive hyper-heuristic for examination timetabling problems" in Appl Intell (2012) 37:1–11 DOI 10.1007/s10489-011-0309-9.
[5] M. R. Garey and D. S. Johnson, "Computers and intractability," Freeman, 1979.
[6] D. Br_elaz, "New Methods to Color the Vertices of a Graph", Communications of the ACM 22 (1979) 251.
[7] M. Luby, "A simple parallel algorithm for the maximal independent set problem," SIAM journal on computing, vol. 15, no. 4, pp. 1036–1053, 1986.
[8] M. T. Jones and P. E. Plassmann, "A parallel graph coloring heuristic," SIAM Journal on Scientific Computing, vol. 14, no. 3, pp. 654–669, 1993.
[9] D. J. A. Welsh and M. B. Powell, "An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems", Computing Journal 10 (1967) 85.
[10] D. W. Matula, G. Marble, and J. D. Isaacson, "Graph Coloring Algorithms", Academic Press, New York, 1972.
[11] T. F. Coleman and J. J. More,"Estimation of Sparse Jacobian Matrices and Graph Coloring Problems", SIAM Journal of Numerical Analysis 20 (1983) 187.
[12] E. G. Boman, D. Bozdaˇg, U. Catalyurek, and Gebremedhin, "A scalable parallel graph coloring algorithm for distributed memory computers," in Euro-Par 2005 Parallel Processing, pp. 241–251, Springer.
[13] Buhua Chen, Bo Chen, Hongwei Liu, Xuefeng Zhang "A Fast Parallel Genetic Algorithm for Graph Coloring Problem Based on CUDA" in 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery.
[14] Werra DD, Eisenbeis C, Lelait S, Marmol B. "On a graph-theoretical model for cyclic register allocation". iscrete Applied Mathematics 1999; 93(2- 3):191-203.
[15] Werra DD. "An introduction to timetabling". European Journal of Operational Research 1985; 19:151-162.
[16] Nishant M Gandhi, Rajiv Misra, "Performance Comparison of Parallel Graph Coloring Algorithms on BSP Model using Hadoop" in 2015 International Conference on Computing, Networking and Communications, Cloud Computing and Big Data.

