JCOL: A Java package for solving the graph coloring problem

Shalin Shah[1]

1 Johns Hopkins University

Summary

The graph coloring problem aims at assigning colors to the nodes of a graph such that no two connected nodes have the same color. The graph coloring problem is NP-complete and one of the harder problems to solve. Here we present a heuristic to solve this problem using three cascaded algorithms. The graph coloring problem was one of Karp's 21 NP-complete problems, and is also know as the problem of finding the chromatic number of a graph. Several other problems reduce to graph coloring including solving generalized Sudoku puzzles.
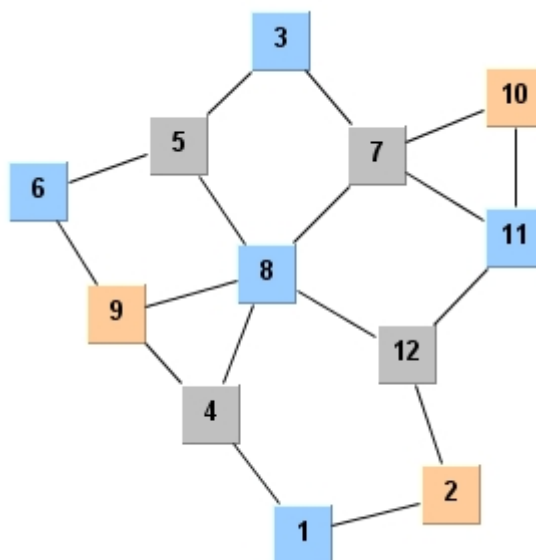
Figure 1: Three Coloring.

The problem of graph coloring can be solved using exact approaches like branch and cut and using heuristics like this work. Exact algorithms are not suitable for very large instances with more than 500 vertices. Malaguti & Toth (2010) surveys some of the algorithms that are exact and heuristic and find that the % gap for exact algorithms, even when allowed to run for 7200 seconds is quite large. This algorithm is able to solve several large instances, and finds the optimum chromatic number or at least a very close to optimum solution in a short amount of computational duration. When an exact algorithm is available and suitable, it is of course the preferred way as the solution can be provably optimal. But for very large graphs, this is often unsuitable because of the exponential complexity of NP-hard problems. Our heuristic can help in these cases. Results of our algorithm can be found on the software page.

The ColPack solver (Gebremedhin, Nguyen, Patwary, & Pothen, 2010) was run on some of the benchmark instances and the results are presented in the software README and in the table below. To summarize, this method is as good or better on all the randomly chosen instances. On the queen9_9 and le450_5d instances, this method out-performs ColPack. Also, this method is randomized and iterative so it can be left running for a few hours to see if the coloring improves further (for large graphs with unknown chromatic numbers). The following table shows the results of the coloring. The run time for JCOL for 100 iterations is shown in the last column (milliseconds). This could be improved by decreasing the number of iterations and by disabling local search.

DataSet	LARGEST FIRST	SMALLEST LAST	INCIDENCE DEGREE	This Algorithm	ColPack(ms)	JCOL(ms)
—	—	—	—	—	—	—
fpsol2.i.3	30	30	30	30	32	430
inithx.i.3	31	31	32	31	36	457
le450_5d	14	12	14	7	33	361
mulsol.i.5	31	31	31	31	38	342
zeroin.i.3	30	30	30	30	30	332
games120	9	9	9	9	30	229
miles750	32	31	32	31	31	276
queen9_9	15	15	15	11	30	218
myciel7	8	8	9	8	30	569
—	—	—	—	—	—	—
GM*	19.55	19.15	19.88	17.43		

*GM=Geometric Mean

This software uses the DSATUR (Brélaz, 1979) heuristics along with iterated greedy heuristics (Culberson, 1992) to color a graph. The DSATUR heuristic orders the nodes of a graph in non-increasing order of the degree of saturation. The degree of saturation is the number of colors found in the adjacency list of a node. The iterated greedy heuristics perform randomized ordering of the vertices to color them in that order. By randomizing this process, improvement can be found in successive iterations. The algorithm then uses min-conflicts local search to improve the coloring. The method is quite successful in finding good colorings of the majority of the publicly available data sets. Results can be found on the software repository page.

The method uses the following steps:

1) Compute a clique (maximum is good)
2) Color the clique
3) Sort the rest of the vertices in non-increasing order of the degree of saturation
4) Color the vertices in the order given by 3. Also, when a vertex is colored, change the degree of saturation of the neighboring vertices so that the order of coloring changes
5) Improve the coloring using Iterated Greedy techniques
6) Improve the coloring using min-conflicts local search
7) Report the coloring

The problem instances are available here: https://mat.tepper.cmu.edu/COLOR/instances.html and http://sites.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm.

References

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, *22*(4), 251–256. doi:10.1145/359094.359101

Culberson, J. (1992). Iterated greedy graph coloring and the difficulty landscape. doi:10. 7939/R3M32NH6Q

Gebremedhin, A., Nguyen, D., Patwary, M., & Pothen, A. (2010). ColPack: Software for graph coloring and related problems in scientific computing. *Submitted to ACM TOMS*. doi:10.1145/2513109.2513110

Malaguti, E., & Toth, P. (2010). A survey on vertex coloring problems. *International transactions in operational research*, *17*(1), 1–34. doi:10.1111/j.1475-3995.2009.00696.x