**K Graph Coloring**

Samir Banjanovic, Joseph Ouellet

**Abstract**

K Graph Coloring is taking a set of nodes with their respective connections and completely coloring each node. There is only one rule. The rule is a color cannot have any connection with the same color. This can be accomplished quite easily with a small amount of nodes and connections, but as the number of nodes and connections increase so does the complexity. For this reason, we have chosen to research this NP-Complete problem using Genetic Algorithm (GA) and Artificial Wisdom of Crowds (WOC). During the course of our research and development of this program we have found that given artificial experts using WOC we can see that even though multiple experts can have either an optimal or not optimal solution that as a general consensus it combines every crowd’s color choice and we will see how it could pick both the optimal and not optimal number of colors on different runs of the program.

1. INTRODUCTION

There are several ways to solve a K Graph Coloring problem. Through the course of our research we have found it was best for us to use the GA & WOC in combination with our WOC resolver that fixes problems that arise from combining all the artificial crowd results together based on popularity. Using this method we were able to solve a K Graph Coloring problem within milliseconds. As we get more complex graphs with more nodes and connections we can still, using this method, solve the graph in a timely manner with an optimal solution. Sometimes the WOC does not generate the optimum solution, but that is the very definition of WOC. Given enough experts and enough generations of the GA the GA & WOC combined algorithm will find the most optimal solution which is the amount of colors required to completely color the graph without violating the rule that given a connection the nodes cannot have the same color.

1. GRAPH DESIGN

The first decision to make when designing the graph is the amount of nodes it will contain. While we worked on this project we used 8, 9, and 10 nodes. The next portion was plotting the nodes using an (X, Y) coordinate plane. We decided to go with the matrix grid approach using 0 for no connection and 1 for a connection between two points going from 1 to the number of nodes. The actual position (being the X and Y coordinates) really don’t matter much other than to create a visually pleasing design and help the user understand where the connections are located.

The design is first planned out on paper and then converted into X and Y coordinates to make the designs more visually appealing. The matrix connections we had to plot manually which is why we only went up to 10 nodes. The matrix of the amount of nodes uses the formula n x n where n is the number of nodes. So, 10 nodes would result in 100 0’s and 1’s that we would have to manually plot in the data file. Initially, we planned on using 10, 20, and 30 points, but the matrix connections would be 100, 400, and 900 respectively so we limited our nodes to 8, 9, and 10.

1. USER INTERFACE

The user interface (UI) consists of basic elements for testing and data evaluation. There are three general sections. The left interaction, rendering, and results area. The interaction area allows a user to dictate multiple parameters used in the solving and rendering of results. The area is broken down into six sections. The sections are:

1. Iteration setter Options
2. Rendering Options
3. Coloring Approach Options
4. GA Parameter Options
   1. Population Size
   2. Generation Cap
   3. Mutation Rate
5. Wisdom of Crowds Options
6. Data/Results Write option

A file is loaded using the load button; once start is pressed the algorithm executes and searches for an answer. If "Show all members" is checked the UI will render every GA result it generates, regardless if it's valid or not.

The Rendering area consists of a large canvas/grid that renders the graph and displays the solution. In the lower left of the grid a small details section displays data such as:

1. Algorithm used
2. Execution time
3. Number of colors used
4. Completeness of graph

Due to the speed of the algorithm it is important to note that the UI is not synchronized with the execution. Results and rendering events are queued so that the UI can render them properly and in the correct order.

1. GENETIC ALGORITHM

The GA revolves around the idea of growing the number of colors each time it fails to find answer using a set number of colors (K). This cycle is done until an answer is found or the algorithm runs out of colors to use. There are currently 14 colors available for coloring; thus ensuring a high probability of finding an answer.

Generation zero coloring algorithm can either be random or intelligent. The random approach selects a random color, limited to K options, for each point. The intelligent (The Colorist) approach allows generation zero to be colored using simple rules.

Each time a new color is introduced the algorithm creates a new generation zero based on the selected coloring option and destroys all previous generations as they were not able to yield a result.

The Colorist has the option to select the first point at random and choose a random color and then selecting each consecutive point based on the number of connections each point has. Ties with same number of connections are broken by using the node with the smaller Id value. The other option is to select the first point based on the most connections and continue selecting subsequent points based on number of connections.

The crossover function acts on two parents. For both parents a random slice point is selected using a random number generator.



The slice creates two children; each being a representation of the slice combinations.

As the crossover occurs mutation is introduced using the set mutation rate (M.R.). The mutation function selects two points at random and colors them using any of the other available K colors. The chance of mutation is introduced for every child.

Mutation points (R) R = 3, R = 5



To promote diversity in our population different generations are allowed to coexist and crossbreed. The idea behind this approach is that the combination of a

Fitness is a very trivial evaluation. The fitness function evaluates the result and uses two criteria:

1. Completeness
2. Number of Colors (K)

A graph is considered complete if every point is colored and no connected points have the same color. The solution with the smaller K in conjunction with completeness is considered fitter than a solution that is complete and has a larger K.

Population size is managed in two different ways. After each generation the population is sorted from most to least fit, after the sort 20% of the population is killed. After every five generations the population size is reset to the original number, keeping only the fittest members.

1. WISDOM OF CROWDS

WoC works in conjunction with the GA to receive experts needed for finding a solution. Using the set number of experts (E) the GA will execute E times to find an expert. After each execution the expert data is handed to the WoC algorithm. When expert data is added the point coloring is analyzed and stored in a point color table. The table consists of rows representing the number of points (P) and columns represent the number of colors (K). Each row, column ({P,K}) combination represents how many times a color was chosen for a point.

Once all experts are received the algorithm evaluates each point and picks the color that has the highest occurrence counter. To improve the performance of WoC we implemented a WoC Resolver which is discussed in the next section.

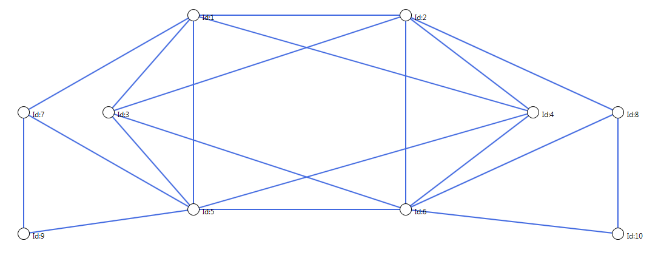
1. WISDOM OF CROWDS RESOLVER

If WoC is set to only rely on using point color counter then the answers received is incomplete majority of the time. This is due to the fact that color selection isn't limited based on neighboring points. To fix this problem we introduced a resolver that evaluates the selected color for the point. The resolver manages point colors by keeping track of used and available colors.

Anytime a points color is selected we add it to a color bucket. If a conflict arises the resolver looks in the bucket for another option. If no options are available a new color is introduced from the color wheel, which at the same time is added to the bucket.

1. RESULTS

For the initial results data was collected for a 10 point graph using two different GA settings and four different coloring settings.



The only GA setting changed was the number of generation the execution should run for. More test are underway to examine results with changes in other parameters. For these results the only parameter changed was the Generation Cap (GC), between 100 and 300. The stable population size was set to 50 and mutation rate to 0.5%.

Tests were run for ran for the following combinations:

1. GA + Random Coloring
2. GA + Colorist
3. GA + Random Coloring + WoC
4. GA + Colorist + WoC

Tests were repeated for 50 iterations and data was analyzed with the following results.



GA + Random Coloring (GC 100)

For a GC of 100 GA + Random Coloring was able to achieve a 3 color solution 46% and 4 color solution 54% of the time with an average execution time of 234.62ms.

GA + Colorist (GC 100)

GA + Colorist achieved a 3 color solution 100% of the time with an average execution time of 193.40ms.



GA + Random Coloring (GC 300)

When using a GC of 300 GA + Random Coloring achieved a 3 color solution 52% and 4 color solution 48% of the time with an average execution time of 714.40ms.



GA + Colorist (GC 300)

GA + Colorist achieved a 3 color solution 100% of the time with and average execution time of 588.90ms

When using GA and coloring method in conjunction with WoC we noticed a large change in the success rate between using the random coloring option and using the Colorist. Each run the WoC Resolver in order to achieve a complete graph.



GA + Random Coloring + WoC (GC 100, E 10)

When using GA + Random Coloring + WoC with 10 experts (E) a solution of 3 colors was achieved by WoC 14% and 4+ color solution 86% of the time with an average execution time of 2407.31ms.



GA + Colorist + WoC (GC 100, E 10)

GA + Colorist + WoC was able to achieve a 3 color solution 100% of the time with an average execution time of 1925.45ms.

When GC was increased to 300 a slight increase in performance was seen when using WoC; however, the execution time increased greatly.



GA + Random Coloring + WoC (GC 300, E 10)

GA + Random Coloring + WoC was able to achieve a 3 color solution 18% and 4+ color solution 82% of the time with an average execution time of 7195.24ms.



GA + Colorist + WoC (GC 300, E 10)

GA + Colorist + WoC was able to achieve a 3 color solution 100% of the time with an average execution time of 5974.40ms.

1. OTHER RESEARCH CONCLUSIONS

From our research we can say there are several ways to accomplish k graph coloring. Some focused on the minimum k to solve the problem regardless of time, while others looked into optimizing time and risk not getting an optimum color count but close enough. Most of the research involved this “approximate” coloring of the graph like we did where it would cycle through colors starting at 2 colors and implementing a new color each time their algorithms hit a wall.

While there is no absolute best method to solve this problem in NP we can say that the algorithm we used gives a very close to optimum solution similar to several others who performed research before us. There are several ways to change the k graph coloring problem to include for example image processing and or color histograms to make sure all colors are equal and not given favoritism over others. With our algorithm we added a ton of custom options that made our solution unique while still remaining close to optimum for the given parameters and time constraints given.

1. CONCLUSION

When evaluating our findings for a 10-point graph we can see that applying GA with random coloring or aid of the Colorist function represents a viable solution to finding minimal K. With the colorist we are able to achieve 100% minimal K over 50 runs with the tested GA settings; however, when tested for 100 iterations it’s success rate was 95.6%. Both the Colorist and random coloring have a high dependency on the Generation Cap (GC) and stable population size (SPS). It is in our benefit to conduct more tests to observe performance for variations in GC and SPS.

Since we were able to achieve a minimal K rate of 52% with a GC of 300 it’s wise to look for a combination of GA settings that can greatly increase the success rate of the GA + Random Coloring. In small preliminary tests we saw an even higher increase in minimal K usage when we increased the SPS and GC together. This change in variation gives the GA a large pool of diverse members, which improves results. With further testes we can evaluate if it’s possible to achieve the same success rate with GA + Random Coloring as we do with GA + Colorist.

REFERENCES

[1] M. de Longueville, “A Course in Topological Combinatorics”, [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-1-4419-7909-4](http://en.wikipedia.org/wiki/Special:BookSources/978-1-4419-7909-4), Chapter 2

[2] “Graph Theory, Part 2,” www.csun.edu\_~danielk\_teaching\_graph-theory\_notes02.pdf

[3] “Chapter 8: Graph colouring (Book Chapter),” www-sop.inria.fr\_members\_Frederic.Havet\_Cours\_coloration

[4] H. Ayanegui, A. Chavez-Aragon, “A complete algorithm to solve the graph-coloring problem”

[5] I. M. Diaz, P. Zabala, “A Generalization of the Graph Coloring Problem”

[6] M. M. Hindi, R. V. Yampolskiy, “Genetic Algorithm Applied to the Graph Coloring Problem”

[7] S. M. Douiri, “Solving the graph coloring problem via hybrid genetic algorithms”

[8] Z. Kokosiński, K. Kwarciany, M. Kołodziej , “Efficient Graph Coloring with Parallel Genetic Algorithm”

[9] A. Wigderson, “A New Approximate Graph Coloring Algorithm”

[10] M. Krivelevich, “Coloring random graphs – an algorithmic perspective”

[11] Barenboim, L.; Elkin, M. (2009), "Distributed (Δ + 1)-coloring in linear (in Δ) time", *Proceedings of the 41st* [*Symposium on Theory of Computing*](http://en.wikipedia.org/wiki/Symposium_on_Theory_of_Computing), pp. 111–120, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1145/1536414.1536432](http://dx.doi.org/10.1145%2F1536414.1536432), [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-1-60558-506-2](http://en.wikipedia.org/wiki/Special:BookSources/978-1-60558-506-2)

[12] Brooks, R. L.; Tutte, W. T. (1941), "On colouring the nodes of a network", [*Proceedings of the Cambridge Philosophical Society*](http://en.wikipedia.org/wiki/Proceedings_of_the_Cambridge_Philosophical_Society) **37** (2): 194–197, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1017/S030500410002168X](http://dx.doi.org/10.1017%2FS030500410002168X)

[13] Dailey, D. P. (1980), "Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete", [*Discrete Mathematics*](http://en.wikipedia.org/wiki/Discrete_Mathematics_%28journal%29) **30** (3): 289–293, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/0012-365X(80)90236-8](http://dx.doi.org/10.1016%2F0012-365X%2880%2990236-8)

[14] Duffy, K.; O'Connell, N.; Sapozhnikov, A. (2008), ["Complexity analysis of a decentralised graph colouring algorithm"](http://www.hamilton.ie/ken_duffy/Downloads/cfl.pdf), *Information Processing Letters* **107** (2): 60–63, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/j.ipl.2008.01.002](http://dx.doi.org/10.1016%2Fj.ipl.2008.01.002)

[15] Fawcett, B. W. (1978), "On infinite full colourings of graphs", [*Can. J. Math.*](http://en.wikipedia.org/wiki/Canadian_Journal_of_Mathematics) **XXX**: 455–457

[16] Fomin, F.V.; Gaspers, S.; Saurabh, S. (2007), "Improved Exact Algorithms for Counting 3- and 4-Colorings", *Proc. 13th Annual International Conference, COCOON 2007*, [Lecture Notes in Computer Science](http://en.wikipedia.org/wiki/Lecture_Notes_in_Computer_Science) **4598**, Springer, pp. 65–74, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1007/978-3-540-73545-8\_9](http://dx.doi.org/10.1007%2F978-3-540-73545-8_9), [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-3-540-73544-1](http://en.wikipedia.org/wiki/Special:BookSources/978-3-540-73544-1)

[17] Guruswami, V.; Khanna, S. (2000), "On the hardness of 4-coloring a 3-colorable graph", *Proceedings of the 15th Annual IEEE Conference on Computational Complexity*, pp. 188–197, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1109/CCC.2000.856749](http://dx.doi.org/10.1109%2FCCC.2000.856749), [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-7695-0674-7](http://en.wikipedia.org/wiki/Special:BookSources/0-7695-0674-7)

[18] Halldórsson, M. M. (1993), "A still better performance guarantee for approximate graph coloring", *Information Processing Letters* **45**: 19–23, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/0020-0190(93)90246-6](http://dx.doi.org/10.1016%2F0020-0190%2893%2990246-6)

[19] Holyer, I. (1981), "The NP-completeness of edge-coloring", [*SIAM Journal on Computing*](http://en.wikipedia.org/wiki/SIAM_Journal_on_Computing) **10** (4): 718–720, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1137/0210055](http://dx.doi.org/10.1137%2F0210055)

[20] [Khot, S.](http://en.wikipedia.org/wiki/Subhash_Khot) (2001), "Improved inapproximability results for MaxClique, chromatic number and approximate graph coloring", *Proc. 42nd Annual* [*Symposium on Foundations of Computer Science*](http://en.wikipedia.org/wiki/Symposium_on_Foundations_of_Computer_Science), pp. 600–609, [doi](http://en.wikipedia.org/wiki/Digital_object_identifier):[10.1109/SFCS.2001.959936](http://dx.doi.org/10.1109%2FSFCS.2001.959936), [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-7695-1116-3](http://en.wikipedia.org/wiki/Special:BookSources/0-7695-1116-3)