Algorithms on Graph Coloring Problem

Mingtao Yang
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang Province, 150001, China
*Corresponding author’s e-mail: angela@cas-harbour.org

Abstract. Graph coloring problem is one of the most famous NP-complete problem and it is an important problem in the study of set theory and graph theory. Since it was raised, lots of people has tried many methods to solve the problem or to optimize the solution to this problem. In this article, the author will introduce the concept of the graph coloring problem and two solutions of the problem. One is based on genetic algorithm and the other is based on greedy algorithm. Each solution has its own traits. Through the analysis, this paper aims to reduce the complicity of the algorithms for making them useful in various applications.

1. Introduction
Graph Coloring Problem (GCP) is one of the most famous Non-deterministic Polynomial Complete problems. The mathematical definition of GCP is that there is an undirected connected graph \( G = (V, E) \) and \( V \) is the vertex set while \( E \) is the edge set. GCP is to divide the set \( V \) into several color groups and each group is supposed to form an independent set which means there is no adjacent vertex in one set. To be brief, all the vertexes ought to be colored and the adjacent ones should be coloured in different colors. If the graph can be divided into \( m \) independent sets at least, or to say if \( m \) colors used at least can make the adjacent vertexes in different colors, \( m \) is named after chromatic number \( \chi(G) \)[1].

Take the following figures as examples.

![Figure 1: The uncolored graph](image1)
![Figure 2: The colored graph](image2)

If the adjacent vertexes are supposed to get colored in different colors with least colors, the basic method is enumeration. First, vertex 1 is colored with green, and vertex 2 and 3 are colored with yellow. As for vertex 4, its colour must be different from vertex 2 and 3, so it can be colored with green again. The last one is vertex 5 and its color must not be green and yellow, so it can be colored...
with blue. However, when coloring vertex 4, it can be colored with any one if it is not yellow, but the
graph is supposed to get colored with least colors, so green is used to color vertex 1. As a result, the
vertexes are colored in three colors, and that means the vertex set is divided into three independent
sets, so the chromatic number \( \chi(G) \) is 3.

However, if there are hundreds of vertexes, this method is too complete, so other algorithms are
needed.

2. Algorithm based on genetic algorithm

2.1. Genetic algorithm
Genetic algorithm is a calculation model simulating natural selection in the Darwinian biological
evolution and biological evolution in genetics. Before solving the problem, we are supposed to
conduct mathematical modeling first. A feasible solution is seen as a chromosome. Feasible solutions
are always made up of multiple elements and each element is called a gene.

As for the flow of the algorithm, at first, it will randomly generate a set of possible solutions, the
first generation of chromosomes, called a population\[2\]. After that, the fitness function, which is used
to evaluate the fitness of these chromosomes just like the seemingly god in the nature, will calculate
the fitness of each chromosome and the probability that each chromosome will be selected in the next
evolution according to the fitness.

Next, crossover will happen. Crossover means seeking two parent chromosomes, cutting them at
the same location and splicing together, thus forming a new chromosome. The new chromosome
contains some genes from the father chromosome and some genes from the mother chromosome. By
crossover, \( n-m \) pieces of new chromosome will get born.

After crossover, variation will take place on the new-born \( n-m \) pieces of new chromosome. Variation is to select good genes by swapping the combination order to break the current search limit
and the algorithm can find out the best solution.

At last, \( m \) pieces of new chromosomes will get born by copying, which means the chromosomes
with the highest fitness of the previous generation are directly copied intact to the next generation.
Assuming that \( n \) chromosomes need to be generated in each evolution, \( n-m \) chromosomes need to be
generated in each evolution by crossover, and the remaining \( m \) chromosomes are generated by
copying \( m \) chromosomes with the highest fitness of the previous generation\[3,4\].

This is an evolutionary process, followed by a new round of evolution.

2.2. Modularity
Modularity is a concept proposed by Newman in 2003. It reflects how well the outcome of a
community division is.

\[
Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j)
\]

In the formula, \( Q \) is modularity, and the bigger modularity is, the better the effect of the division is.
Suppose there are \( x \) vertexes and each vertex represents an input. We have divided these inputs into \( n \)
communities and there are \( m \) connections between vertexes. \( i \) and \( j \) are two random vertexes in the
graph. When the two vertexes are connected, \( A_{ij}=1 \); otherwise, \( A_{ij}=0 \). \( d_i \) represents the degree of the
vertex \( i \) and \( 2m \) is the degree of the whole graph. \( \delta(c_i, c_j) \) is used to judge whether vertex \( i \) and \( j \) are in
the same community. If they are in the same community, \( \delta(c_i, c_j) \) is supposed to be 1; otherwise,
\( \delta(c_i, c_j)=0 \).

2.3. Analysis of the algorithm

2.3.1. The design of the fitness function.
There is an algorithm aimed to solve the graph coloring problem by the genetic algorithm. The first problem is to choose an adaptable fitness function. In order to divide the vertex set \( V \) into \( k \) independent sets \( \{ V_1, V_2, \ldots, V_k \} \), the modular function of this partition can be inferred from the modular function defined for Newman:

\[
Q(\pi_k) = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{d_id_j}{2m}) \delta(c_i, c_j) = -\frac{1}{4m^2} \sum_{ij} d_id_j \delta(c_i, c_j)
\]  

(2)

In the formula, \( \pi_k \) is the division of the vertex set; \( d_i \) is the degree of vertex \( i \) and \( c_i \) is the independent set which \( i \) belongs to \( c_i \in \{ V_1, V_2, \ldots, V_k \} \). The contribution of vertex \( i \) to \( Q(\pi_k) \) is

\[
q_i(\pi_k) = -\frac{1}{2m^2} \sum_j d_j \delta(c_i, c_j)
\]

(3)

Now expand \( k \) independent sets to \( k + 1 \) independent sets and what is supposed to do is that \( x \) non-adjacent vertexes are selected from \( k \) independent sets to form a new independent and \( V_{k+1} \).

\[
Q(\pi_{k+1}) = Q(\pi_k) - \sum_{i \in V_{k+1}} q_i(\pi_k) + \sum_{i \in V_{k+1}} q_i(\pi_{k+1}) = Q(\pi_k) + \sum_{i \in V_{k+1}} \sum_{j \in V_{k+1}} d_id_j \delta(c_i, c_j)
\]

(4)

Because \( d_i \) is bigger than 0,

\[
Q(\pi_{k+1}) > Q(\pi_k)
\]

(5)

In the absence of independent sets, the partition is measured by the ratio of the edges between vertex modules to the total edges of the graph, namely \( R/e \) and \( \delta(R,e) = 0 \).

When \( R/e = 1 \), an independent division \( \pi \) has been found. If it is not the best division, there must be a partition with smaller \( Q \) values and fewer independent sets \( \pi \), so the fitness function is designed as

\[
f = \frac{R}{e} - (Q-1)\delta(R,e)
\]

(6)

2.3.2. Population initialization.

In the process of establishing the initial population, the vertex set of the graph is randomly divided into \( k \) sets and all the numbers of the vertexes depend on the specific problem. The division is like the following figure.
In the initialization process, each vertex is randomly assigned a set number, but there must be a mechanism to ensure that while initializing the random assignment of set Numbers, vertexes should be reasonably assigned as far as possible according to the connection relation. After the random production of a stain, num gene sites are randomly selected for adjustment. Suppose there are \( g (g \leq k) \) different numbers and that means the vertexes are divided into \( g \) subset \( \{V_1, V_2, \ldots, V_g\} \). We use \( e_r(\pi_g, V_i) \) to denote the number of adjacent points of vertex \( r \) assigned to vertex set \( V_i \) under partition mode \( \pi_g \). The process of adjustment is as follows.

1) \( j = 1, \ e_{\text{min}} = 1, \ temp = 0; \)
2) \( i = 1, \) select a vertex \( r \) at random and place it in set \( V_i \). Calculate \( e_r(\pi_g, V_i) \), if \( e_r(\pi_g, V_i) < e_{\text{min}} \), \( e_{\text{min}} = e_r(\pi_g, V_i), \ temp = i; \)
3) \( i = i + 1; \) if \( i = g + 1, \) step to 4); otherwise, step to 2);
4) The value of \( \text{temp} \) is assigned to the gene location on the chromosome corresponding to vertex \( r \), so \( j = j + 1, \) if \( j = \text{num} + 1 \), it will stop, otherwise 1);

According to the above method, pop size chromosome individuals are generated to form an initial population (where pop size is the population size).

2.3.3. Coding.

The algorithm in this paper adopts natural number coding, because a normal \( k \) coloring of connected graph \( G \) is to divide the vertex set \( V \) of graph \( G \) into \( k \) independent sets, which can be numbered as 1, 2..., \( k \). The gene length of chromosomes is equal to the number of vertexes of graph \( G \).

2.3.4. Selection.

In each generation population, according to the fitness of chromosomes, the elite selection strategy is applied to save the chromosome with the highest fitness. This method can ensure that good chromosomes will not disappear after generation, and the algorithm can converge to the optimal value according to probability 1 in limited steps.

2.3.5. Crossover and variation.

According to the chromosome sequences obtained from the selection operation, they are cross-operated with the crossover probability \( P_c \). By means of crossover, information interaction between chromosomes can be carried out to produce new individuals and make the whole chromosome population show new traits.

When selecting two chromosomes during the crossover operation, it is supposed to select of a gene randomly and then to carry out one way propagation cross, which is to select two chromosomes as paternal chromosomes. One is known as the source of chromosome and the other is called target chromosomes. A collection is selected randomly from the source on chromosome number \( c \), and it is supposed to search iteratively for vertex \( v \) belonging to the set, and vertex \( v \) at target on chromosome number is set as \( c \).

Because set numbers are a relative property, different set numbers on different chromosomes may represent the same module, so it is necessary to adjust the numbers of the two chromosomes. The process of adjustment is listed as follows.

1) Set id as 1 and all genes on the chromosome are marked as unadjusted;
2) Search the chromosomes for unlabeled, unregulated genes. If found, the value of id is assigned to the gene locus and the gene locus with the same value as the gene locus, and these gene loci are marked as adjusted states, set id as id+1;
3) Repeat step 2) until all loci are marked as altered.

After each operation, the candidate individuals were ranked according to the fitness value, and the individuals with the largest fitness were selected as the offspring.

According to the fitness function, the fitness value of independent set partition is obviously higher than that of not independent set partition. In order to prevent the algorithm from falling into the local
optimal solution, the algorithm adopts the formula of adaptive crossover probability $P_c$ and mutation probability $P_m$[5].

$$P_c = \begin{cases}
0.8 - 0.3 \cdot \frac{f' - f_{\text{avg}}}{f_{\text{max}} - f_{\text{avg}}}, & f' \geq f_{\text{avg}} \\
0.8, & f' < f_{\text{avg}}
\end{cases} \quad (7)
$$

$$P_m = \begin{cases}
0.1 - 0.099 \cdot \frac{f_{\text{max}} - f}{f_{\text{max}} - f_{\text{avg}}}, & f' \geq f_{\text{avg}} \\
0.1, & f' < f_{\text{avg}}
\end{cases} \quad (8)
$$

3. Algorithm based on greedy algorithm

3.1. Greedy algorithm

The greedy algorithm is always to make the best choice in the current situation when solving a problem. In other words, instead of thinking about the overall optimal, what it does is a locally optimal solution in a sense. For the greedy algorithm, in order to get a optimal solution, the problem must meet the following two conditions: greedy selectivity and optimal substructure.

Greedy selectivity means that the overall optimal solution of the problem can be achieved through a series of local optimal choices, namely greedy choice. Greedy selectivity is to make successive choices from the top down and iteratively, and each greedy selectivity simplifies the problem to a smaller sub-problem. As for optimal substructure, when the optimal solution of a problem contains the optimal solution of its sub-problems, the problem is said to have the optimal substructure property.

The basic idea of greedy algorithm is to proceed step by step from an initial solution of the problem. According to an optimization measure, the local optimal solution should be obtained at each step. Only one datum is considered in each step, and its selection should meet the conditions of local optimization. If the next datum and partial optimal solution are no longer feasible, the datum is not added to the partial solution until all data are enumerated, or the algorithm cannot be added to stop.

3.2. The RLF Algorithm

The Recursive Largest First (RLF) algorithm was put forward in 1979 by F. Leighton[6,7]. It builds several stable sets which correspond to several classes. It is supposed that C is the next color class to be built, U is the set of uncolored vertices and W is the set (initially empty) of uncolored vertices which owns no less than one neighbor belonging to C, and W is initially empty. Every time a vertex in U will be moved to C and all its neighbors in U will be moved from U to W. The first vertex $v \in U$ in C is the vertex with most neighbors in U. After that, while U is not empty, move the vertex in U with most neighbors in W from U to C. If possible, ties will stop after the choice of a vertex whose number of neighbors in U is the smallest.

For a vertex $u \in U$, the author uses $A_u$ to denote its number of neighbors in U and $A_w(u)$ to represent its number of neighbors in W. Besides, if $v$ is the first vertex in a color class, it is able to use C to denote the color class that contains it. The following algorithm shows how to construct $C_v$ when given a specific vertex $v$ by the RLF algorithm.

Construction of $C_v$

- Input A set U of uncolored vertices and a vertex $v \in U$.
- Output A stable set $C_v$ containing $v$.
- 1) Initialize W as the set of vertices in U adjacent to $v$.
- 2) Remove $v$ and all its neighbors from U and set $C_v \leftarrow \{v\}$.
- 3) while U ≠ ∅, do
- Select a vertex $u \in U$ whose value $A_w(u)$ is the largest. In case of ties, choose one with smallest value $A_u(u)$.
Move vertex u from U to C, and move all neighbor vertices w ∈ U of vertex u to W.
end while

When W=∅ initially, A_w(x) for all x ∈ U equals to 0 and we can get the initial values A_w(x) in O(m) time. After that, when a vertex w is moved from U to W, for all neighbors x ∈ U of w, its A_w(x) will be incremented by one unit and A_u(x) is decreased by one unit. Similarly, when a vertex u ∈ U is moved from U to C, A_u(x) is decreased by one unit for all neighbors x ∈ U of u. Therefore, there are O(m) such updates. Besides, the selection of the next vertex to be moved to C_v will take O(n) time. As a result, the total complexity of the construction of C_v is O(m + n|C_v|). The RLF algorithm is listed as follows.

Algorithm RLF
Input A graph G.
Output A coloring of the vertices of G.
1) k←0.
2) while G contains uncolored vertices do
   Let U be the set of uncolo red vertices. Set k ← k + 1. Choose a vertex v ∈ U with largest value A_w(v).
   Build C_v and assign color k to all vertices in C_v.
end while
Because each vertex belongs to a color class, the overall complexity of the RLF algorithm is O(km + n^2) and k is the total number of the colors used in the graph. When it is the worst case, the complexity will be O(mn).

3.3. Alternative Greedy Choices
In the follow parts, we use the same notations as the RLF algorithm. For a vertex x ∈ U, A_w(x) denotes the number of neighbors of x in W. If a vertex x is moved to W from U, its A_w(x) will not change any more. Therefore, the value A_w(x) of every vertex equals to the value A_w(x) before the movement to W.

Now, the following part introduces two improvements of the greedy choices made in RLF[7].

3.3.1. Alternative greedy choice for the selection of the next vertex to be placed in C_v.
This greedy choice is to select which vertex when selecting a vertex w ≠ v to be placed in C_v. The RLF algorithm selects the vertex u ∈ U whose value A_w(u) is largest. However, there is a new and better way to select the vertex. For every vertex u ∈ U, let

\[ B(u) = \sum_{\omega \in W \cap N(u)} (d(\omega) + A_w(\omega)) \]  

In the formula, N(u) represents the set of neighbor vertices of u while d(\omega) is the number of uncolored neighbors of w during the initial stage of the building of C_v. The next vertex to be moved to C_v is the one with maximum value B(w).

Let G’ be the part of the graph which has not been colored after the construction of C_v. By maximizing \[ \sum_{\omega \in W \cap N(u)} d(\omega) \], we can choose a vertex u with uncolored neighbors w of large degree in W so that the maximum degree in the graph G’ is the smallest. Besides, by making \[ \sum_{\omega \in W \cap N(u)} A_w(\omega) \] largest, many vertices in the G’ are selected out and all the vertices selected are similar to those in C_v, thus making the next color class as same size of C_v.

Let L = \{ u ∈ U | B(u) = \max_{x ∈ U} B(x) \} and L’ = \{ u ∈ L | A_w(u) = \max_{x ∈ L} A_w(x) \}. The next vertex u moved to be moved to C_v is one in L’ with the smallest value A_u(u). Just like the values A_w(u) and A_u(u) in the RLF algorithm, the acquisition of the initial values for B(u) takes O(m) time. Each time a vertex u is moved to W, B(x) is incremented by A_w(x) units for all neighbors x ∈ U of u. Therefore, the
overall complexity of the construction of the stable set $C_v$ does not change at all. The whole procedure is listed as follows.

Construction of $C_v$ based on function $B$

Input: A set $U$ containing uncolored vertices and a vertex $v \in U$.
Output: A stable set $C_v$ that contains $v$.

1) Initialize $W$ which is set of vertices in $U$ adjacent to $v$.
2) Remove $v$ and all the vertices adjacent to vertex $v$ from $U$ and set $C_v \leftarrow \{v\}$.
3) While $U \neq \emptyset$ do
   Suppose $L = \{u \in U \mid B(u) = \max_{x \in U} B(x)\}$ and $L' = \{u \in L \mid A_u(u) = \max_{x \in L} A_u(x)\}$.
   Select a vertex $u \in L'$ whose value $A_u(u)$ is the smallest.
   Move $u$ from $U$ to $C_v$, and move all neighbors $w \in U$ of $u$ to $W$.
end while

3.3.2. Alternative selection of the first vertex of a color class.

The second greedy choice is to select which vertex as the first vertex $v$. In the RLF algorithm, $v$ is a vertex with a maximum value $A_0(v)$ of neighbors in $U$. Several possible alternatives are listed as follows.

1) The first alternative is to construct a stable set $C_v$ for every uncolored vertex $v$ and to pick out a vertex which can produce a residual graph with a minimum number of edges, which makes the total complexity of the algorithm $O(mn^2)$.

2) In order to avoid increasing the complexity from $O(mn)$ to $O(mn^2)$, it is able to construct a stable set $C_v$ for a constant number $M$ of uncolored vertices $v$ with the largest values $A_U(v)$.

A solution in-between is to follow alternative (2), but with $M = \lfloor pn \rfloor$ and $0 < p < 1$, which makes the overall complexity $O(mn^2)$, but approximately decreases the total computing time by a factor $p$ when compared with alternative (1).

4. Discussion

From what the author has said above, the two algorithms are both solutions aiming to solve the graph coloring problem and they have different characteristics. As for the algorithm based on genetic algorithm, genetic algorithm has the characteristics of group search. At the beginning of the search, there is an initial group which include multiple individuals. From one aspect, it can effectively avoid searching some points which are not supposed to be searched. Nevertheless, genetic algorithm is easy to be irregular and inaccurate in coding. Besides, the efficiency of genetic algorithm is usually lower than other traditional optimization methods. When it comes to the algorithm based on greedy algorithm, it is efficient and easy to understand and finish. However, it has weaknesses either. It relies on previous steps too much. In addition, the final solution is a collection of the optimal solution of each step, but it may not be the global optimal solution. All in all, the algorithm that is selected to solve the problem depends on the specific demands and physical truth.

5. Conclusion

As a classical NP-complete problem, graph coloring problem has a good application background in both theoretical and engineering application, such as frequency distribution problem of transmitting station[8]. There are several possible solutions but no one knows for sure which solution is the best one. In the article, the author introduces two of them. One based on genetic algorithm is to use modular function to make the number of independent set as small as possible and it has a good convergence rate. The other based on greedy algorithm is to get the optimal solution of every step, but it may not be the optimal solution of the whole problem. During the study of this problem, many new algorithms are raised and several new fields are broaden. In this article, the author introduces two algorithms based on genetic algorithm and greedy algorithm. Perhaps they are not the simplest ones, but they are the most fundamental ones. Having a complete understanding of these algorithms can help study other problems and in the future, study of graph coloring problem as well as more algorithms
will be raised. The direction of the study is to reduce the complexity of the algorithms and to make them useful in various applications.

Acknowledgement
First and foremost, I would like to express my deepest gratitude to my teachers and professors in and out my university, who have provided me with valuable guidance in every stage of the writing of this thesis. Furthermore, I would like to thank my friends and parents for their encouragement and support. Without all their enlightening instruction and impressive kindness, I could not have completed my thesis.

References
[1] Isabel Méndez-Díaz, Paula Zabala, A Branch-and-Cut algorithm for graph coloring, Discrete Applied Mathematics 154 (2006) 826-847.
[2] Thang Nguyen Bui and Byung Ro Moon, Genetic Algorithm and Graph Partitioning, IEEE Transactions on Computers, Vol 45, No 7, July 1996.
[3] Georges R. H., Fernando G. L. and David E. G. The Compact Genetic Algorithm, IEEE Transactions on Evolutionary Computation,Vol 3, No 4, November 1999.
[4] Eiben A. E., Vender Hauw J. K., Van Hemert J. I. Graph Color ring with Adaptive Evolutionary Algorithms [J]. Journal of Heuristics, 1996, 4 (1) : 16224.
[5] Li, X. Q., Zhang, N. Graph Coloring Algorithm Based on Parting Vertexes into Several Independent Sets, Journal of Harbin University of Science And Technology, Vol 15, No 5 October 2010.
[6] Leighton, F. T. “A graph coloring algorithm for large scheduling problems”, Journal of Research of the National Bureau of Standards, 84 (1979) 489-503.
[7] Mourchid Adegbindin, Alain Hertz, Martine Bellá´iche, A New Efficient RLF-like Algorithm For the Vertex Coloring Problem, Yugoslav Journal of Operations Research 26 (2016), Number 4, 441-456.
[8] Han, L. X., Wang, Y. P. A new genetic algorithm for graph coloring problem, Journal of Xidian University, April 2008, Vol 35, No 2.