Solving Graph Coloring Problems Using Cultural Algorithms

Reza Abbasian and **Malek Mouhoub**
Department of Computer Science
University of Regina
Regina, Canada
abbasiar@cs.uregina.ca, mouhoubm@cs.uregina.ca

Amin Jula
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
jula@ce.sharif.edu

Abstract

In this paper, we combine a novel Sequential Graph Coloring Heuristic Algorithm (SGCHA) with a non-systematic method based on a cultural algorithm to solve the graph coloring problem (GCP). The GCP involves finding the minimum number of colors for coloring the graph vertices such that adjacent vertices have distinct colors. In our solving approach, we first use an estimator which is implemented with SGCHA to predict the minimum colors. Then, in the non-systematic part which has been designed using cultural algorithms, we improve the prediction. Various components of the cultural algorithm have been implemented to solve the GCP with a self adaptive behavior in an efficient manner. As a result of utilizing the SGCHA and a cultural algorithm, the proposed method is capable of finding the solution in a very efficient running time. The experimental results show that the proposed algorithm has a high performance in time and quality of the solution returned for solving graph coloring instances taken from DIMACS website. The quality of the solution is measured here by comparing the returned solution with the optimal one.

1 Introduction

Graph coloring is a combinatorial optimization problem which has many applications such as scheduling (Leighton 1979), frequency allocation (Gamst 1986) and task scheduling (Chaitin 1981). Let G be a given graph; solving the Graph Coloring Problem (GCP) consists of determining its chromatic number denoted by $\chi(G)$. $\chi(G)$ is the minimal number of colors needed to color the graph vertices such that two adjacent (neighboring) ones have different colors (see Figure 1). Finding the chromatic number for a graph is NP-hard and deciding whether a graph is k-colorable or not is NP-complete (Garey and Johnson 1979).

Yet for such hard problems many algorithms and their variations have been developed and explored. Also, several heuristics have been proposed for solving the GCP. There are generally three main solving approaches. The first one consists of directly minimizing the number of colors by working in the legal colors space of the problem. The second approach comprises of first choosing a numbering color K, and then iteratively try to minimize the number of conflicts for the candidate K. Whenever a solution with zero conflicts has been found, K is reduced by one and this procedure continues until we reach a K where the number of conflicts cannot be equal to zero. As a result, the last legal K will be kept as the best solution. For the third approach, the number of colors is fixed and no conflict is allowed, thus, some vertices might not be colored. The objective of the algorithm is to maximize the number of colored vertices (Mabrouk, Hasni, and Mahjoub 2009).

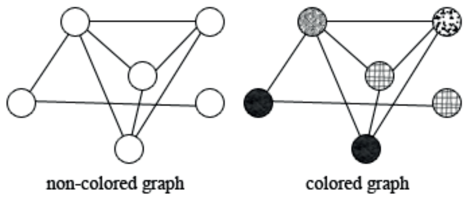

Figure 1: An example of graph coloring problem

In this paper, we propose a solving method including two phases: the **Estimation Phase (EP)** and the **Improvement Phase (IP)**. For the EP we have designed a new algorithm based on the idea of sequential graph coloring meaning that vertices will be colored sequentially with a systematic approach. The EP provides an upper bound for the graph's chromatic number and passes this prediction to IP. Then the IP uses the predicted number as one of its initial parameters and starts to improve the results. To design an algorithm for the IP, we used the second strategy in solving the GCP with non-systematic approach, but with a considerable difference compared to the idea described in that strategy. We combined the second strategy with the facilities provided by the cultural algorithm. In other words, the algorithm synchronously obtains legal coloring combinations for various values of K. Whenever the problem is solved for a value of K, all groups that are organized for solving the problem with a greater value are discarded.

Evolutionary computation is the metaphorical use of concepts, principles and mechanisms extracted from our understanding of how natural systems evolve to help solve complex computational problems. Currently, much of this work has focused on the processes of natural selection and genetics (Reynolds 1994). Cultural algorithms are based on some theories proposed in sociology and archaeology to model cultural evolution. Such theories indicate that cultural evolution can be seen as an inheritance process that occurs at two levels: the micro-evolutionary level, and the macro-evolutionary level (Coello and Becerra 2003). At the micro-evolutionary level, there is a population of individuals, each described in terms of a set of behavioral traits. Traits can be modified and exchanged between individuals by means of a variety of socially motivated operators. At the macro evolutionary level, individuals experiences are collected, merged, generalized, and specialized in the belief space. This information can serve to direct the future actions of the population and its individuals. Therefore, cultural algorithms are useful in exploring large search space accumulating global knowledge about the problem space. Also, it is clear that cultural evolution proceeds at a faster rate than biological evolution (Chung and Reynolds 1996).

The remaining of the paper describes our efficient cultural algorithm combined with a new systematic method for solving GCP. First, basic concepts of cultural algorithms are introduced. Then, we introduce the EP and IP algorithms in detail in Sections 3 to 5. Finally, the experimental results are listed in Section 6 and a conclusion and possible future work are discussed in section 7.

2 Basic Concepts of Cultural Algorithm

Cultural algorithms were developed by Robert G. Reynolds as a complement to the metaphor adopted by evolutionary algorithms, which was mainly focused on genetic concepts and on the natural selection mechanism (Reynolds 1994). Cultural evolution can be seen as an inheritance process that occurs at two levels: the micro-evolutionary level, and the macro-evolutionary level. At the micro-evolutionary level, individuals are described in terms of behavioral traits (which can be socially acceptable or unacceptable). These behavioral traits are passed from generation to generation using several socially motivated operators. At the macro-evolutionary level, individuals are able to generate mappa (Renfrew 1994), or generalized descriptions of their experiences. Individual mappa can be merged and modified to form group mappa using a set of generic or problem specific operators. Both levels share a communication link. The micro-evolutionary level refers to the knowledge acquired by individuals through generations which, once encoded and stored, is used to guide the behavior of the individuals that belong to a certain population (Renfrew 1994; Durham 1994). Reynolds attempts to capture this double inheritance phenomenon through his proposal of cultural algorithms (Reynolds 1994). The main goal of such algorithms is to increase the learning or convergence rates of an evolutionary algorithm such that the system can respond better to a wide variety of problems (Franklin and Bergerman 2000). Therefore, the cultural algorithm can be viewed as a dual inheritance system with evolution taking place at the population level and at the belief level. The two components interact through a communications protocol. The protocol determines the set of acceptable individuals that are able to update the belief space. Likewise, the protocol determines how the updated beliefs are able to impact the adaptation of the population component. A component level description of the cultural algorithms is given in Figure 2 (Soza et al. 2011).

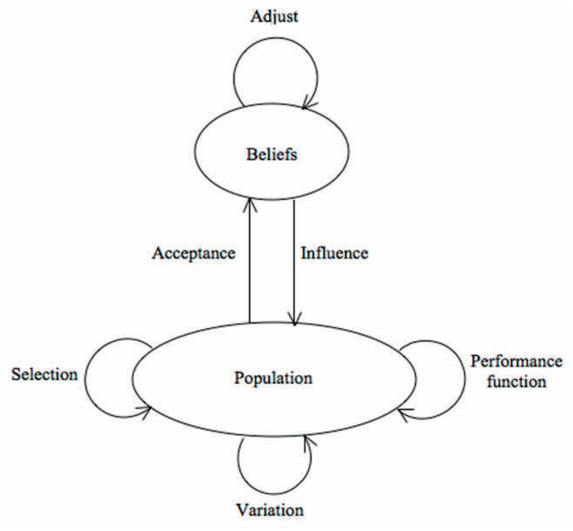

Figure 2: Spaces of a Cultural Algorithm

Cultural algorithms operate in two spaces. First, there is the population space, which consists of (as in all evolutionary algorithms) a set of individuals. Each individual has a set of independent features that are used to determine its fitness. Through time, such individuals can be replaced by some of their descendants, which are obtained through the application of a set of operators from the population. The second space is the belief space, which is where the knowledge, acquired by individuals through generations, is stored. The information contained in this space must be accessible to each individual, so that they can use it to modify their behavior.

```
Generate the initial population
Initialize the belief space
Evaluate the initial population
repeat
    Update the belief space (with the individuals accepted)
    Apply the variation operators (under the influence of the
    belief space)
    Evaluate each child
    Perform selection
until the end condition is satisfied
```

Figure 3: Pseudo-code of a Cultural Algorithm

Figure 3 shows the pseudo-code of a Cultural Algorithm (Soza et al. 2011). Most of the steps of a cultural algorithm correspond with the steps of a traditional evolutionary algorithm. It can be clearly seen that the main difference lies in the fact that cultural algorithms use a belief space. In the main loop of the algorithm, the belief space must be updated. It is at this point in which the belief space incorporates the individual experiences of a select group of members of the population. Such a group is obtained with the function accept, which is applied to the entire population.

On the other hand, the variation operators (such as recombination or mutation) are modified by the function influence. This function applies some pressure such that the children resulting from the variation operators can exhibit behaviors closer to the desirable ones and farther away from the undesirable ones, according to the information stored in the belief space (Soza et al. 2011).

3 Estimator Phase (EP)

The algorithm receives graph G as input and initializes an empty graph A and at each step adds a vertex to A according to vertices in G. The EP algorithm is as follows.

1. Create a list from vertices of G based on their degrees in a decreasing order.
2. Choose an uncolored vertex g_i from G with the maximum degree and add it to A and name it a_i.
3. Apply the constraints between the newly added vertex a_i and the rest of vertices in A according to G.
4. Solve the partial[1] GCP and mark g_i as colored in G.
5. While there exists an adjacent vertex adj_{gi} to g_i in G that does not have a corresponding vertex adjacent to a_i in A do the following.
   (a) Choose the vertex adj_{gi} with maximum degree and add it to A and name it adj_{ai}.
   (b) Apply the constraints between adj_{ai} and the rest of vertices in A according to its correspondence to G.
   (c) Solve the partial GCP and mark adj_{gi} as colored in G.
6. If there exists an uncolored vertex goto step 2.
7. Return the total number of colors used.

The idea behind the EP algorithm is to first identify the most constrained sub-graph of G (which is the sub-graph created by the most constrained vertex and its adjacent vertices) and then solve the whole sub-graph according to the constraints that we have so far in the partially constructed graph A. Once a sub-graph is solved, the algorithm moves to the next most constrained unsolved sub-graph. This process continues until the whole problem is solved and there is no other uncolored vertex left. The algorithm uses a greedy method for choosing a color for a vertex since it always seeks for the minimum available color.

Figure 4 shows the steps of the algorithm for a sample graph. For the sake of simplicity, we suppose that colors are enumerated starting with zero. In each step of the algorithm, we add one vertex to the partial graph A. As a result, we just need to check the adjacency matrix for newly added vertex and choose a color with the minimum possible number for the added vertex. The algorithm discussed above is rather conceptual as we can implement it without actually using the partial graph A. We only need to keep track of colored vertices; every colored vertex is in A. The algorithm can be implemented to run in $O(|V|^2)$ where $|V|$ is the number of vertices.

[1]The term partial graph (partial GCP) means a graph (GCP) is not completely constructed yet to be exactly like the original graph (GCP).

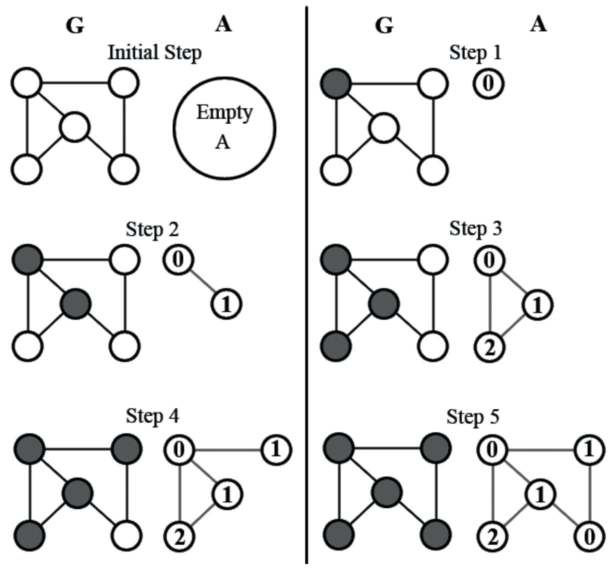

Figure 4: Steps of EP algorithm for a sample graph

4 Components of the Improvement Phase Algorithm

Prior to formulating the problem, the following terms must be described.

- *Belief of an Individual:* The belief of an individual is the number of colors that it suggests as the minimum colors for coloring the graph.
- *Belief of a Group:* The belief of a group is the number of colors that the group suggests as the answer of graph coloring.
- *General Belief:* The general belief is the best solution found so far by the algorithm, that is the current minimum number of colors.
- *Group:* A group can be defined as a set of individuals having the same belief.

Next, the following components of the algorithm are discussed in detail.

1. *Representation of Individuals*

   Each individual in the population is represented with an integer array, which has a length equal to the number of graph vertices. The value of each column is a color number less than or equal to the general belief. Figure 5 illustrates an example of an individual for an eight vertex graph with the general belief equal to three.

2. *Fitness Function*

   The fitness function evaluates the number of conflicts that might exist among adjacent vertices which have the same

0	2	2	1	0	1	0	2

Figure 5: Example of an individual (eight vertex graph)

color. In order to compute this value we simply find adjacent vertices from the graph adjacency matrix and check their color number in the integer array of the individual. Therefore, the total number of color conflicts is the result of the fitness function. When the result is equal to zero, a solution is found. The fitness of a solution, f_s, is defined as follows:

$$f_s = \sum_{i \in V_G} \sum_{j \in adj_i} conflict(i,j)$$

where V_G is the set of all vertices of the graph and adj_i is the set of all adjacent vertices to vertex i.

The conflict function is defined as follows:

$$conflict(i,j) = \begin{cases} 1 & \textit{if i and j have the same color} \\ 0 & \textit{otherwise} \end{cases}$$

3. *Reproduction and Crossover*

Reproduction takes place amongst a number of fittest individuals of each group and amongst fittest individuals of different groups. The chosen individuals are then passed to crossover as parents of new individuals. To produce each new child, the crossover operator randomly chooses a pivot. It then finds the vertex colors of the first parent's vertices which have a vertex number less than or equal to the pivot and concatenates them with the vertex colors of the second parent's vertices which have a vertex number greater than or equal to the pivot. Figure 6 shows an example of the crossover with a pivot equal to 2 on two individuals of a five vertex graph coloring problem.

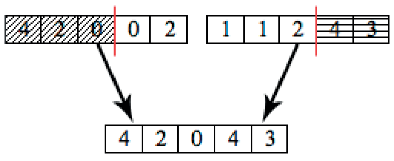

Figure 6: Example of crossover (five vertex graph)

The reproduction and crossover operations are performed under the influence of the belief space. This assures that each newly produced individual has a belief which is less than or equal to the general belief. To proof this, we rely on the fact that each parent passed to the crossover function has color numbers that are less than or equal to the general belief. When the crossover is applied to individuals, it only concatenates them regarding to the pivot value and it never modifies the color of any vertex. As a result, it can be deduced that the vertices of the produced child have color values between 1 and the general belief.

4. *Mutation*

We propose two different methods for the mutation.

(a) *Mutation to minimize the number of conflicts.* In this type of mutation, some random vertices of the individual is selected and the number of color conflicts around the chosen vertices and their adjacent vertices are minimized.

(b) *Stochastic color change.*This mutation method chooses some vertices and randomly changes their color. The color number is less than or equal to the general belief.

As discussed before, in a cultural algorithm, the mutation operator is influenced by the belief space. Here the influence means operating in the range of colors which are less than or equal to the general belief. The first method proposed for the mutation has been implemented such that the number of used colors for individuals does not exceed the general belief. In other words, during the process of minimizing the number of conflicts, when a vertex is chosen for color change, its new color is selected from available colors within the individual. Therefore we can claim that the numbers of used colors remain less than or equal to the general belief. It is also intuitive that the second proposed mutation method always results in individuals that have color numbers less than or equal to the general belief. Thus, it can be concluded that the proposed mutation operator always produces individuals that satisfy the constraints of the belief space.

5. *Self Adaptiveness*

The proposed algorithm keeps track of the ratio of improvement of each group (r_g) and best individuals in available groups. Whenever the algorithm finds out that there is not any improvement in the groups, it adds K_g newly random generated individuals to each group and increases the mutation chance until an improvement is achieved in a group; once this condition is met, it removes the additional individuals from the population and resets the mutation chance to its initial value. The algorithm calculates a new value for each group's r_g at the end of each iteration according to the following formula:

$$r_g = 1 - \frac{(\textit{current best fitness})_g}{(\textit{previous best fitness})_g}$$

It is obvious that $0 \leq r_g \leq 1$. K_g is calculated with the following formula:

$$K_g = \begin{cases} \lfloor K \times (r_g + 0.1) \rfloor & r_g \leq 0.9, \\ & \textit{belief}_g \geq \textit{general belief} - 2 \\ \lfloor K \times r_g \rfloor & \textit{otherwise} \end{cases}$$

The value of K_g ranges over $[0, K] \subset \mathbb{Z}^+$ where K is the maximum number of individuals that can be added to a group. The way K_g and r_g are calculated implies that each time more focus is given to the groups with bigger ratio of improvement and especially to groups with beliefs near the general belief.

6. *Stopping Criteria*

The algorithm is allowed to enter the Self Adaptive phase S consecutive times. If after S consecutive times no improvement has been made in the groups, the algorithm terminates and returns its latest result as the final solution.

5 Algorithm for the Improvement Phase (IP)

1. Set B to the result of the EP (B now is equal to the estimated minimum number of colors). Randomly generate a population of size P which has virtually B groups; meaning that there exists B different beliefs among individuals and each group consists of P/B individuals.
2. Set the current general belief to B.
3. If the stopping criteria are satisfied (which means $T = S$, where T and S are respectively the current and the maximum number of tries) stop the algorithm and return the best result found so far. Else go to step 4.
4. Update the belief space by finding a new possible belief that is better than the current one.
5. Check the fitness value of the best individual in each group and compare it to their previous fitness value. If no progress has been made in the groups, advance T by one and run self adaptive mechanism. Otherwise, remove the changes of self adaptive mechanism (if any) and set T to zero and reset the mutation chance to its default value.
6. If the general belief is changed, influence the population by removing individuals belonging to the groups with a belief greater than the new general belief.
7. Perform reproduction and mutation and go to step 3.

6 Experimental Results

Our proposed algorithm has been implemented using Java language and has been applied to a variety of graph coloring instances. The test machine had an Intel Core 2 Duo CPU of 2.5 GHz with 4 GB of Memory and JDK 1.6. Furthermore, the graph coloring instances used in this section are from an Internet website formally named DIMACS graphs[1].

Table 1: Comparison of EP and DSATUR algorithms

Problem	χ_{EP}	χ_{DSATUR}	χ
zeroin.i.2.col	31	31	30
mulsol.i.1.col	49	50	49
queen10_10.col	15	15	?[1]
mulsol.i.2.col	31	32	31
2-Insertions_4.col	5	5	4
1-Insertions_5.col	6	6	?
myceil7.col	8	8	8
miles250.col	8	9	8
miles1500.col	73	73	73
le450_25b.col	25	25	25

1.The chromatic number is not reported by DIMACS.

First, we have compared our proposed EP algorithm with a well-known sequential graph coloring algorithm, namely DSATUR of Brèlaz (Brélaz 1979) in terms of resulting estimations of chromatic number (χ). Table 1 which shows the results of this comparison suggests that in some cases our proposed algorithm returns better results. However, in most cases both algorithms return the same result. Note that in theory, the complexity of our EP algorithm is $O(|V|^2)$ while the complexity of DSATUR is $O(|V|^3)$ (Klotz 2002).

[1]http://mat.gsia.cmu.edu/COLOR03/

Next, we have compared our algorithm with the parallel genetic-tabu algorithm (PGTA) designed to solve GCP (Mabrouk, Hasni, and Mahjoub 2009). The tests have been conducted without and with our estimator EP and the results are respectively reported in Table 2 and Table 3.

Table 2: Comparative Results without EP

Problem Instances			**IP Only EP Disabled**[1]			**PGTA Algorithm**[2]	
Instance	V	E	$T(s)$	χ	UC[3]	$T(s)$	χ
queen5_5	25	160	1.10	5	0	2.18	5
queen6_6	36	290	3.98	8	1	5.05	7
queen7_7	49	476	10.15	8	1	11.18	8
games120	120	638	122.81	9	0	153.47	9
miles250	128	387	55.05	8	0	185.46	8
anna	138	493	128.33	11	0	229.85	11

1. To disable the EP, at the first step of the IP algorithm, we initialize B to the number of vertices instead of the result of the EP.
2. The results are taken from experiments with two processors (Mabrouk, Hasni, and Mahjoub 2009).
3. The number of Unresolved Conflicts (UC) of the best individual in the group with correct belief.

Table 3 lists the results of the second experiment with different problem instances.

Table 3: Comparative Results with EP

Problem Instances				**Proposed Algorithm (EP+IP)**		
Instance	V	E	χ	$T_{EP}(s)$	$T_{EP+IP}(s)$	χ
myciel3	11	20	4	0.001	0.18	4
myciel4	23	71	5	0.001	0.40	5
queen5_5	25	160	5	0.001	0.95	5
queen6_6	36	290	7	0.002	2.84	7
myciel5	47	236	6	0.002	1.24	6
queen7_7	49	476	7	0.002	7.81	8
huck	74	301	11	0.002	3.12	11
jean	80	254	10	0.004	2.73	10
david	87	406	11	0.003	5.55	11
games120	120	638	9	0.008	16.23	9
miles250	128	387	8	0.007	8.99	8
anna	138	493	11	0.008	9.67	11

Results of Table 2 and Table 3 are shown together in Figure 7. According to the chart, it is obvious that the proposed algorithm with EP is more efficient. The main reason for such efficiency is that the other algorithms proposed in the literature (e.g. PGTA (Mabrouk, Hasni, and Mahjoub 2009)) start the solving process assuming the graph is $|V|$ colorable at the beginning. They then iteratively try to improve the results. In our proposed algorithm, we provide a rough estimate for the chromatic number using a systematic algorithm (EP) which is very fast. Then, the IP algorithm tries to improve this by estimation working on different possible chromatic numbers at the same time. This way, the algorithm saves a lot of time reaching the solution. Finally, in Figure 8 we compare the running time results of our algorithm which are shown in Table 3 with the running time results of PGTA gathered with 24 processors. The figure shows that our algorithm performs better in most cases.

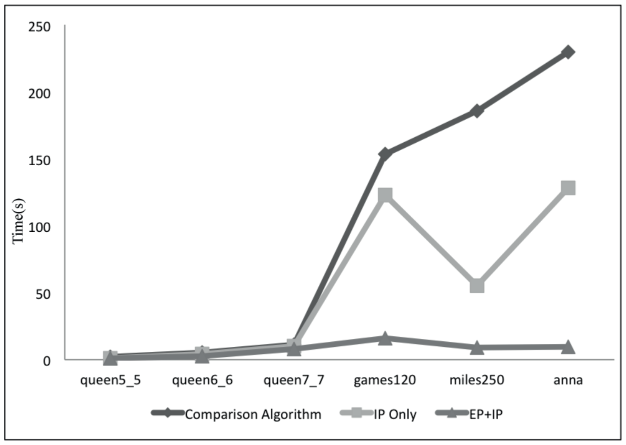

Figure 7: Comparison of different algorithms' results

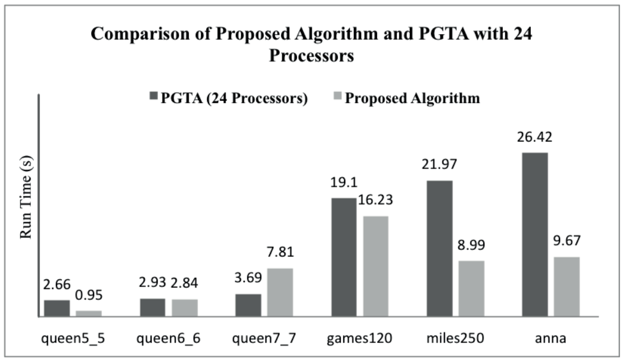

Figure 8: Comparison of the proposed algorithm and PGTA with 24 Processors in terms of running times

7 Conclusion and Future Work

This paper proposes a new approach for solving GCP using a combination of a systematic method and a cultural algorithm. The systematic method is based on an estimator that we propose. The use of the cultural algorithm resulted in reaching a solution in an acceptable time. However, when combined with our systematic method, the proposed algorithm is capable of finding the solution in a very efficient running time. The experimental results were gathered mainly on medium sized problem instances primarily because our algorithm is sequentially implemented. Due to the nature of cultural algorithms, further work can be done to parallelize the IP. Solving very large instances of the graph coloring problem seems to be unfeasible without parallelization. Moreover, for large problem instances one can develop a scheme for finding a lower bound for the graph's chromatic number. This would allow the cultural algorithm to be more focused on possible chromatic numbers and as a direct result the algorithm would use less memory and processor time.

References

Brélaz, D. 1979. New methods to color the vertices of a graph. *Commun. ACM* 22:251–256.

Chaitin, G. 1981. Register allocation via coloring. *Computer Languages* 6:47–57.

Chung, C.-J., and Reynolds, R. G. 1996. A testbed for solving optimization problems using cultural algorithms.

Coello, C., and Becerra, R. 2003. Evolutionary multiobjective optimization using a cultural algorithm. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, 6 – 13.

Durham, W. H. 1994. *Co-evolution: Genes, Culture, and Human Diversity*. Stanford, California: Stanford University Press.

Franklin, B., and Bergerman, M. 2000. Cultural algorithms: concepts and experiments. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, 1245 –1251 vol.2.

Gamst, A. 1986. Some lower bounds for a class of frequency assignment problems. *Vehicular Technology, IEEE Transactions on* 35(1):8 – 14.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Instractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.

Klotz, W. 2002. Graph coloring algorithms. In *Mathematics Report*, 1–9. Technical University Clausthal.

Leighton, F. 1979. A graph coloring algorithm for large scheduling problems. *Journal for Research National Bureau of Standards* 84:489–505.

Mabrouk, B. B.; Hasni, H.; and Mahjoub, Z. 2009. On a parallel genetic-tabu search based algorithm for solving the graph colouring proble. *European Journal of Operational Research* 197(3):1192–1201.

Renfrew, A. C. 1994. Dynamic modeling in archaeology: What, when, and where? *Dynamical Modeling and the Study of Change in Archaelogy*.

Reynolds, R. G. 1994. An introduction to cultural algorithms. In Fogel, L. J., ed., *Proceedings of the Third Annual Conference on Evolutionary Programming*, 131–139. River Edge, New Jersey: World Scientific.

Soza, C.; Becerra, R. L.; Riff, M. C.; and Coello, C. A. C. 2011. Solving timetabling problems using a cultural algorithm. *Applied Soft Computing* 11(1):337–344.