USING automaton learning to implement a new algorithm in graph coloring

Mahdi Ahmed Ali
Technical College of Engineering - Baghdad
Middle Technical University

Abstract
Graph coloring is one of the NP-Complete problems. One of the main uses of this technique is coloring maps. This article proposed a new algorithm based on a learning automaton to color the graph. The learning process starts with several random automatons. Each automaton alone represents a random coloring and by repeating the learning process, the coloring process improves. Compared to genetic algorithms, An algorithm has been proposed that converges faster with the optimal solution, the reason is that genetic algorithms only look for the optimal chromosome in the population and do not care about the position of the genes in the chromosomes. On the contrary, object migrating automata (OMA) determine the optimal location of genes in the proposed algorithm.
Also, the proposed algorithm converges with the ideal solution in the presence of fewer educational iterations after comparing it with the number of generations of the genetic algorithm. The proposed algorithm that have the same results have been compared with the genetic algorithm results on the highlighted graphs.
Keywords: learning automata…object migration automata…genetic algorithm…random coloring...chromosomes.

1. Introduction

Automaton learning has many uses, one of which is to color the graph.
In our problem where the G graph has V nodes and E edges, the purpose of coloring the graph is to assign a color to each node in it to ensure that no two adjacent nodes have the same color.
The genetic algorithm is among the non-deterministic heuristic methods, and it is considered one of the important algorithms that have been used in the process of graph coloring. The genetic algorithm starts its work with several random populations and searches for the most suitable chromosome by repeating the generation. The goal of a genetic algorithm is to find the best chromosome in a population, but the location of genes in the chromosomes is not taken into account. If it is possible to determine the appropriate position of the genes in the chromosomes, it is possible to reach the optimal solution in a much smaller number of generations. Learning automaton can be used to find the appropriate position of genes in chromosomes.
Learning automaton can find the appropriate position of genes in chromosomes in a repetitive learning process and by applying penalty and reward operators.
The proposed algorithm uses machine learning to find the right place for genes. In the proposed algorithm, each chromosome is equivalent to an automaton and each gene is equivalent to action. Each automaton represents a random coloring. The proposed algorithm improves the appropriate location of genes by repeating the learning process.

2. Introducing the problem

A graph is represented in the form of a binary G= (V, E), where V stands for the set of nodes and $E \subseteq V \times V$ stands for the set of edges. The graph coloring problem is one of the NP-Complete problems.
In such problems, the number of nodes in the graph increases, the time required is growing exponentially to find the optimal result.
As a result, random search methods are used to find near-optimal solutions for such problems, and their execution time grows linearly.
Random search solutions do not guarantee obtaining the optimal solution, but they can obtain solutions that are close to optimal. In this problem.
The goal of the best solution is to minimize the number of colors needed to color the graph so that no two nodes are the same color.

3. Learning automaton [1]

A learning automaton; is a short form that randomly choose an action from a limited set of actions and executes it in the environment. The environment evaluates the action selected by the learning automaton, and the result of the evaluation is announced to the learning automaton by a reinforcement signal. The learning automaton uses the selected action to update its internal status and then selects its next action.
The environment can be represented by the triplet $E = \{a, b, c\}$ where $a = (a_1, a_2, \dots a_n)$ is the input set, $b = (b_1, b_2, \dots b_n)$ is the output set, and c= $(c_1, c_2, \dots c_n)$ is the set of penalty probabilities.
When b is a two-membered set, the environment is of type P, in this environment, here b1 = 1 represents the punishment, and b2 = 0 represents the reward. In a Q-type environment, group b has a limited number of members; In an S-type medium, group b has unlimited members. c_i means being punished for the action a_i. Learning automatons divides into two parts, i.e. fixed structure and variable architecture. [2]3][4][5][6]

4. The proposed algorithm [7]

To color a graph with n nodes using m colors, there are m^m different coloring parameters. If the learning automaton is used to solve the graph coloring problem, the learning automaton must have m^m action. A large number of actions slow down the convergence speed. For this purpose, object migrating automata (OMA) is used, which was proposed by ommen[5] and Ma[6]. One of the OMAs is based on the Test line[7] automaton. The proposed algorithm uses the Test line automaton to solve the graph colorization problem. In this automaton, $a = (a_1, a_2, \dots a_k)$ action set is allowed for learning automaton. The k element represents the actions number in the automaton performs, which is equal to the nodes number in the graph.
Also, N is the memory depth for the automaton. The set of states of each automaton is divided into K sub-sets. Each node is classified according to its status. In the set of states of action k, the state $\phi_{(k-1)N+1}$ is called the internal state, and the state ϕ_{kN} is called the border state. A node that is in the state of $\phi_{(k-1)N+1}$ is called the most important node and a node that is in the state of ϕ_{kN} is called the node of least importance.
The pseudo-code of the proposed algorithm is clearly shown in procedure 1as follows:

Algorithm gcula (graph coloring using learning automata)
Input g: graph
Output s: almost perfect coloring for g
Method
1. Generate an initial population, p, that is randomly generated
OMA staining of g.
2. For Generation = 1 to the maximum no. generations,
m, do
3. For any automaton, a, in p do
4. choose an action, α_i randomly
5. Apply relation 1 mentioned in paragraph II.2 to

reward or punish αi.

6. if rewarded and J < N, upgrade its state from ϕ_{iJ} to ϕ_{iJ+1}
7. else if penalized J > 1 downgrade its state from ϕ_{iJ+1} to ϕ_{iJ}
8. else if and J = 1 swap the action α_i with α_J such that satisfy color α_i

End

procedure 1. Pseudo-code of the proposed algorithm

Proposed algorithm is divided into two parts:

I. Primary population production.

II. Penalty and reward operator.

I. Primary production of the population

At first, P random automaton is created and graph nodes are assigned to automaton actions figure1.

Figure 1. A random automata

After assigning the nodes to automaton actions, the following algorithm as shown in figure3 is executed on all automatons.

Procedure acta (assign colors to actions)

Input N: Nodes

Output a: automata

Method

1. For i = 1 to N do
2. assign i to α_i
3. For i = 1 to N/2
4. y1 = random number
5. y2 = random number
6. If ($y1 \neq y2$) then
7. Swap α_{y1} value with α_{y2}

End

Figure 2. Assigning colors to actions

By running the following algorithm, a random number is assigned to each of the actions. Each number represents a color. The previous code shows a random automaton that represents a random coloring. The resulting coloring of this automaton can be seen in figure 3 as follows:

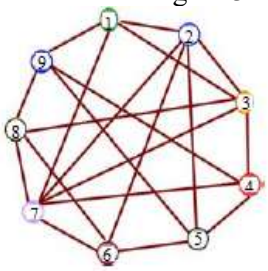


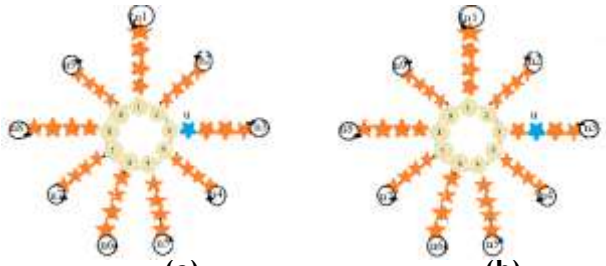
Figure 3. Coloring resulting from the automaton

II. Penalty and Reward Operator

In each automaton, an action is randomly selected and rewarded or penalized. As a result of rewarding or penalizing an action, the status of that action changes in the set of action statuses. If an action is in a border state, its penalty will change its action and thus create a new coloring. The penalty and reward operators are different according to the type of learning automaton.

II.1 Reward operator

When a reward is given for the action, the action may be in the internal or non-internal state. If the desired action is in a non-internal state, then awarding the action will change its state to an internal state. Figure 4 shows how to reward the action in a non-domestic situation. If the desired action is in the internal state, no change will be made in the state of the action by applying the reward.



(a) **(b)**

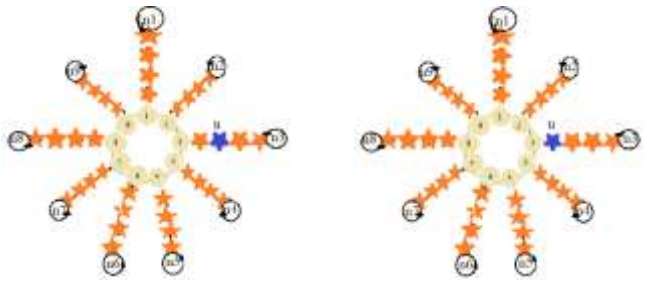
Figure 4. (a) Node state before reward (b) Node state after reward

II.2 Penalty operator

When an action is penalized, two possible situations occur:

1. The action should be in a non-boundary state.
2. The action should be in the border state.

If the desired action is in a non-border state, by applying a penalty, the state of that action will be changed to a border state. Figure 5 shows the procedure. If the desired action is in a border situation, the value assigned to the desired action (the color assigned to the action) is replaced by the number assigned to the action. After replacing that number, no two adjacent nodes are the same color.



(a) (b)

Figure 5. (a) Node state before penalty (b) Node state after penalty

If the action to be replaced is in its border state, then its value is changed to the penalized action, but if the action to be replaced is not in its border state, it is first transferred to its border state and its value changes to a penalized action. Figure 6 shows how the action is penalized in the border state.

(a) **(b)** **(c)**

Figure 6. (a) Node form before penalty (b) Move node to boundary state (c) Replacing u with U

In this section, it is determined on what basis penalties and rewards are applied.

One of the most important principles in learning automatons is determining the appropriate relations for penalty and reward operators. Using equation 1, it is determined that node Ni of the graph should be penalized or rewarded. We consider the following assumptions:

£: The sum of all the colors of nodes that are connected to node n

K: The number of edges related to node Ni

F: Represents the sum of the edges in the graph.

And below is an explanation of the relationship between Contingency

Penalty and reward to action Ni

$$Ni\begin{cases} \text{Reward} & \text{if } (£/K \le K/F) \\ \text{Penalize} & \text{if } (£/K > K/F) \end{cases}$$

5. Simulation results [8][9][10]

The proposed algorithm is implemented in a simulator to evaluate the results. This simulator finds the graph in the form of a text file and after running the proposed algorithm, it returns the minimum colors required to color the input graph as output. The two algorithms (proposed and genetic) are compared in this field.

Considering graphs number 1 to 4 in the table1, we can observe the results of the genetic algorithm and the proposed algorithm. As is clear, the proposed algorithm field results are almost better than those in the genetic algorithm field.

Table1. Comparison of the genetic algorithm and the proposed algorithm for graphs.

Graph number	The proposed algorithm	Genetic algorithm	Number of edges	Number of vertexes	Reference
1	10	11	406	87	(8)
2	10	11	301	74	(9)
3	9	10	254	80	(10)
4	4	4	20	11	(10)

Table2. The parameters of the proposed algorithm for simulating the graphs in table١ are shown in this table.

Graph number	Iteration	Primary population
١	٢٥٠	٣٠٠
٢	250	٢٥٠
٣	٢٠٠	250
٤	٥٠	١٠٠

We conclude from these tables, that the proposed algorithm approaches the optimal solution with less repetition, or in other words, it has a high convergence speed.[10]

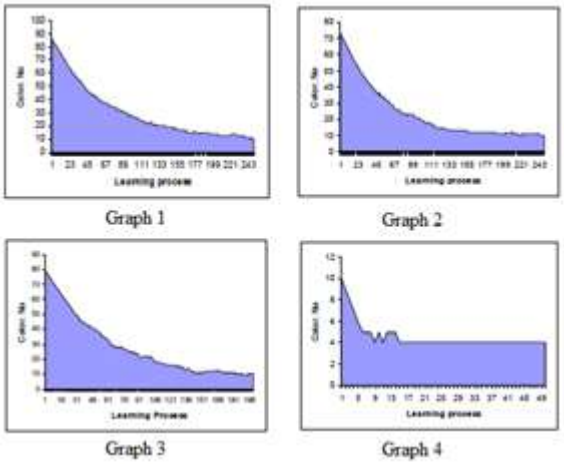


Figure 7. The learning process of the proposed algorithm for the graphs in Table1.

6. Conclusion

In this research we proposes a non-deterministic algorithm that uses automaton learning for graph performance. This algorithm reduces the number of colors needed to color the graph, by using reward and punishment factors. This research uses Object Migration Automation (OMA) based on Automated Line Learning Test as it was applied in the proposed genetic algorithm. Finally, the proposed algorithm was compared with the genetic algorithm, for the graph, and through our reading of the result of the comparison in the tables, it was found that the best results are that using the proposed algorithm is better and faster than the genetic algorithm in coloring graphs.

Acknowledgment

In the name of Allah the Merciful

I am very grateful and grateful to my colleagues abroad and at home. My colleagues in the General Secretariat of the Middle Technical University and my brothers in the Administrative Technical College of Management/ Baghdad helped me in writing and publishing this research. I invite them to help them and preserve them as a source of nurturing science, research, and researchers in the service of our dear country.

References

[1] Learning automata: an introduction. Narendra, Kumpati S., and Mandayam AL Thathachar. Courier corporation, 2012.

[2] "Graph coloring problem based on learning automata." Turkestan, J. Akbari, and M. R. Meybodi In 2009 International Conference on Information Management.

[3] "New Class of Learning Automata Based Schemes for Adaptation of Backpropagation Algorithm Parameters. Meybodi, M. R, and H. Beigy, Proc. Of EUFIT-98, Sep. 7-10, Achen, Germany, pp. 339-344, 1998.

[4] Oommen, B. J., and Ma, D. C. Y., "Deterministic Learning Automata Solution to the Keyboard Optimization Problem", IEEE Trans. On Computers, Vol. 37, No. 1, pp. 2-3, 1988.

[5] "Optimization of Topology of Neural Networks Using Learning Automata", Beigy, H. and Meybodi, M. R. Proc. Of 3rd Annual Int Computer Society of Iran Computer Conf. CCC-98, Tehran, Iran, pp. 417-428, 1999.

[6] Optimization by simulated annealing: an experimental evaluation; part II graph coloring and number partitioning, Operation Research Vol 39, David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, Catherine Schevon, No 3 May-June 1991.

[7] Graph_Coloring_Problem_Based_on_Learning_Automata.Dr-Javad-Akbari/Islamic_Azad_University, Mohammad-Meybodi /Amirkabir-University-of-Technology, Publication/224506759.

[8] "A new genetic algorithm for graph coloring." Raja Marappan, and Gopalakrishnan Sethumadhavan, in 2013 Fifth International Conference on Computational Intelligence, Modeling, and Simulation, pp. 49-54. IEEE, 2013.

[9] "Genetic Algorithms in Search, Optimization and Machine Learning", David E Goldberg, University of Illinois Urbana-Champaign, The University of Alabama, University of Michigan, Ann Arbor, Reading, MA: Addition-Wesley, 1989.

[10] A new algorithm for graph coloring using learning automata, Habib Mati Qadir Abbas Mirzaei Samrin Ali Akbar Dadjoyan Islamic Azad University Tabriz Branch – Researchers ClubYoung scholar of Islamic Azad University, Ardabil Branch, Islamic Azad University, Shebastar Branch, Scholarship. Aliakbar_z2004@yahoo.com a.mirzaei@iaut.ac.ir Habib_moti@yahoo.com. [Translated from Persian].