ISBN 978-1-84626-xxx-x
Proceedings of 2009 International Conference on Machine Learning and Computing
Perth, Australia, 10-12 July, 2009, pp. xxx-xxx

Cellular Learning Automata-based Graph Coloring Problem

Alireza Enami Eraghi[1], Javad Akbari Torkestani[2] and Mohammad Reza Meybodi[3]

[1]Computer Engineering Department, Islamic Azad University, Farahan Branch, Iran. alireza_enami@yahoo.com
[2]Department of Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran. j-akbari@iau-arak.ac.ir
[3]Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. mmeybodi@aut.ac.ir

Abstract. The vertex coloring problem is a well-known classical problem in graph theory in which a color is assigned to each vertex of the graph such that no two adjacent vertices have the same color. The minimum vertex coloring problem is known to be an NP-hard problem in an arbitrary graph. In this paper, an irregular cellular learning automata (ICLA) based approximation algorithm is proposed for solving the minimum (vertex) coloring problem. It is shown that by a proper choice of the parameters of the algorithm, the probability of approximating the optimal solution is as close to unity as possible. Our proposed algorithm is compared with some well-known coloring algorithms and the results show the superiority of the proposed algorithm both in terms of the color set size and running time of algorithm over the existing methods.

Keywords: Graph coloring problem, heuristic algorithms, cellular learning automata.

1. Introduction

The vertex coloring problem (VCP) is a well-known combinatorial optimization problem in graph theory. A legal vertex coloring of graph G=(V,E) , where V(G) is the set of |V|=n vertices and E(G) is the edge set including |E|=m edges, is a function $f: V \rightarrow C$ from the vertices of the graph G to the color-set C={c_1,c_2,…,c_p} such that f(u) $\neq$ f(v) for all edges (u,v) $\in$ E. That is, a legal vertex coloring of G is assigning one of p distinct colors to each vertex of the graph in such a way that no two endpoints of any edge are given the same colors. Formally, the vertex coloring problem can be either considered as an optimization problem or as a decision problem. The optimization version of the vertex coloring problem is intended to find the smallest number of colors by which the graph can be legally colored, and the decision problem aims at deciding for a given p whether or not the graph is p-colorable, and is called p -coloring problem. Vertex Coloring is a well-known NP-hard problem (see Garey and Johnson [1]) with real world applications in various engineering fields, including scheduling [2], timetabling [3], register allocation [4], frequency assignment [5] and communication networks [6].

A given graph G is p-colorable, if it can be legally colored with at most p different colors. The chromatic number χ(G) is the minimum number of colors required for coloring the graph, and a graph G is said to be p-chromatic, if χ(G)=p. A minimum coloring of G is a legal coloring in which the smallest number of colors (i.e., chromatic number) to be assigned to the vertices.

In [7], Akbari Torkestani and Meybodi proposed four approximation algorithms for solving the graph coloring problem based on learning automata, when each automaton updates its action probability vector using a $L_{R\text{-}I}$ reinforcement scheme. In this paper, we propose an algorithm based on irregular cellular learning automata for solving this problem. The proposed algorithm is compared with the algorithms of Caramia[8], Funabiki[9] and Malaguti[10]. The Computation results indicate that the proposed algorithm provides better results obtained from color set size, but in running time, the above mentioned algorithms has a better performance.

The rest of this paper is organized as follows: in section 2, a brief review of irregular cellular learning automata is given. In section 3, we describe proposed algorithm based on irregular cellular learning automata and The performance of the proposed algorithm is evaluated through the simulation experiments in section 4. Section 5 is conclusion.

2. Irregular Cellular Learning Automata

An irregular cellular learning automata (ICLA) [11] is a CLA in which the restriction of rectangular grid structure in traditional CLA is removed. This generalization is expected because there are applications such as wireless sensor networks, immune network systems, graph related applications, etc. that cannot be adequately modeled with rectangular grids. An ICLA is defined as an undirected graph in which, each vertex represents a cell which is equipped with a learning automaton. The learning automaton residing in a particular cell determines its state (action) on the basis of its action probability vector. Like CLA, there is a rule that the ICLA operate under. The rule of the CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in a cell. The neighboring LAs of any particular LA constitute the local environment of that cell. The local environment of a cell is nonstationary because the action probability vectors of the neighboring LAs vary during the evolution of the ICLA.

3. The Proposed Algorithm

For mapping a graph to the cellular learning automata, we use the irregular cellular learning automata, and associate a cell with each vertex of the graph. Then, we assign a learning automaton to each cell of irregular cellular learning automata. Two cell in the cellular learning automata are neighbours, if and only if they are adjacent to each other in the graph. The number of actions of each automaton in the cellular learning automata is equal to the number of colors required for coloring the graph. Thus, the color-set of each vertex is the action-set of its corresponding automaton. Each automaton updates its action probability vector using a L_{RP} learning algorithm whit the reward and penalty parameters 0.01. At first, for all learning automata, the initial probability of selecting an action is inversely proportional to the number of colors.

In [12], it is shown that the minimum number of colors required for a legal coloring, χ, is less than or equal to $\Delta+1$, when Δ is the maximum degree of a node. That is $\chi \leq \Delta + 1$. Therefore, in this algorithm, the action-set size is at least $\Delta+1$.

In this algorithm, we use the degree of a cell as a criterion for rewarding or penalizing the action taken by the cell. The degree of a cell is defined as the number of its neighboring cells that select different actions (or different colors). In our proposed algorithm, each learning automaton selects an action from its set of actions according to its probability vector. If the action selected by a cell is different from the actions taken by all its neighbors it is rewarded. If the selected action by a cell is chosen by some of its neighboring cells, then it is rewarded if the degree of the cell is greater than the degree of all it neighboring cells (that choose the same color). Otherwise, the action taken by the cell is penalized. The algorithm continues until the action taken by all cells are reward. In section four, to improve the performance of our algorithm we changed it stop condition as the probability of choosing a legal coloring. In this improvement, the proposed algorithm stops when the probability of a legal coloring exceeds a predefind threshold. The proposed algorithm consists of a number of stages which is described in detail bellow

Step1: Mapping the input graph to an irregular cellular learning automata

In this step, to each vertex of graph are assigned one cell and to each cell a learning automaton, and each cell in the cellular learning automata are neighboring with other cells if and only if they are neighbor with each other in graph.

Step2: Color selection

In this step, each learning automaton selects an action from its set of actions according to its probability vector for coloring the graph.

Step3: To evaluate the legal colorings.

A legal coloring is to assign a color to each vertex; where colors on adjacent vertices are different.

Step4: Rewarding or penalizing the selected colorings.

If the selected action by cell's learning automaton is different from neighbor cells' action, then, it is rewarded, but if the selected action for a cell is done by some its neighboring cells, then, it is rewarded, only if the degree (number of its neighboring cells that select different action) of the cell is greater than the degree of all neighbor cells which have same color with that cell; and, otherwise, the cell action is penalized.

Step5: To study stop of algorithm.

The algorithm repeat until all cells are rewarded, otherwise, algorithm resumes from step 2.

Pseudo code of this algorithm is given in Fig. 1.

```
Algorithm CLA
Input : Graph G(V,E)
Output : The number of colors needs to be chosen for coloring the graph
Procedure Saturation degree(cell):
            Determines the number of neighbors for a given cell, which they choose different action.
Begin
  Construct an irregular CLA isomorphic to the input graph
  Repeat
    For all cells do in parallel
      Each cell chooses one of its actions according to its action probability vector
       If (The chosen action is different from those chosen by all its neighbors)
         Reward the chosen action
       Else
         If (The saturation degree(cell) is greater than those of  all its neighbors)
           Reward the chosen action
          Else
            Penalize the chosen action
         End if
       End if
  Until all cells are rewarded by the random Environment
  Return the number of  used colors
End.
```

Fig. 1: Pseudo code of the proposed algorithm

4. Simulation Results

To study the efficiency of our proposed algorithm, we have conducted a group of simulation experiments on a subset of hard-to-color benchmarks reported in DIMACS [13] such as random graphs (DSJC*n.x*), geometric random graphs (DSJR*n.x* and R*n.x*[*c*]), quasi-random graphs (flat*n*-*x*-0), artificial graphs (le*n*-*x* and latin-square-10), and a number of graphs from the real life applications (school1 and school1-nsh). The simulation are executed on a P IV 2.4 MHz computer, with 512 MB RAM, under Windows XP. The results of our proposed algorithm are compared with ther results of some well-known existing graph coloring algorithms like Caramia[8], Funabiki [9] and Malaguti [10] both in terms of the number of colors required for coloring the graph (C), and the running time of algorithm in second (T). Each result given in table 1 is averaged over 50 runs. The first and second columns of table 1 represents the number of vertices (|V|) and the number of edges (|E|) of the benchmark graphs, respectively. The third column of this table (denoted as best) is the best-known result (minimum number of color) reported in the literature for each graph. The last column of table 1 (diff.) represents the difference between the number of colors used in our proposed algorithm for coloring the graph and the best number of color reported in literature.

The results of our proposed algorithm on 19 benchmark graphs reported in DIMACS show that the proposed algorithm, in comparison with the other algorithms, decreases the number of colors in 5 instances, increases it in 2 instances, and does not change the previous results in12 instances.

4.1. Improvements of the Proposed Algorithm

In this section, we study the impact of the reward/penalty parameters and stop condition on the performance of the proposed algorithm. To do so, we changed the learning parameters to 0.01, 0.1, 0.2, and 0.3, and measured the algorithm performance. We also changed the stop condition from 0.2 to 0.8 and repeated the same experiments. The obtained results are reported in table 2. In all experiments, the results show that the number of colors used for coloring the benchmark graphs increases as the reward/penalty parameters increases. On the other side, the results show that the running time of algorithm decreases as the reward/penalty parameters increases. The main advantage of this method is that with a proper choice of the reward/penalty parameters, a trade off between the running time of algorithm and number of colors can be made.

Table 1: number of colors and running time for graph coloring with several algorithms

graph	*|V|*	*|E|*	*best*	*proposed algorithm*		*caramia [8]*		*fanabiki[9]*		*malaguti[10]*		*diff.*
				T	*C*	*T*	*C*	*T*	*C*	*T*	*C*	
DSJC1000.1	1000	49629	20	312.01	20	140	22	90	21	260	20	0
DSJC1000.5	1000	249826	84	931.81	85	531	95	4658	88	8407	84	1
DSJC1000.9	1000	440449	225	1035.27	225	857	241	1565	228	3234	225	0
DSJR500.1	500	3555	12	81.53	12	2	12	0	12	25	12	0
DSJR500.1C	500	121275	85	211.61	85	7	85	6	85	88	85	0
DSJR500.5	500	58862	122	137.92	121	269	129	276	122	163	122	-1
le450-15a	450	8168	15	169.53	15	42	16	1	15	0	15	0
le450-15b	450	8169	15	173.21	14	19	16	1	15	0	15	-1
le450-15c	450	16680	15	191.19	15	213	17	11	15	3	15	0
le450-15d	450	16750	15	243.59	16	68	18	5	15	4	15	1
r1000.1	1000	14378	20	121.4	20	13	20	0	20	37	20	0
r1000.1c	1000	485090	98	859.31	98	173	99	557	98	518	98	0
r1000.5	1000	238267	234	615.27	233	727	256	1345	237	753	234	-1
school1	385	19095	14	113.5	14	0	14	0	14	0	14	0
school1-nsh	352	14612	14	94.25	14	1	14	1	14	21	14	0
latin-square-10	900	307350	99	726.43	98	687	107	938	99	5156	101	-1
flat1000-50-0	1000	245000	50	863.14	50	551	92	14	50	1417	50	0
flat1000-60-0	1000	245830	60	927.01	58	562	93	59	60	3645	60	-2
flat1000-76-0	1000	246708	83	1042.19	83	293	94	2499	87	7325	83	0

We define the stop condition as the probability of choosing a legal coloring. That is, the stop condition is the product of the probability of choosing the colors of a legal coloring. To choose the optimal stop condition, we changed it from 0.2 to 0.8 with increment step 0.2 and tested our algorithm. The results are given in table 3. the obtained results show that the number of colors required for coloring the graphs decreases as the stop condition increases. On the other side, the results show that the running time of algorithm increases as the stop condition increases.

Table 2: Number of colors and running time for graph coloring with the proposed algorithm (L_{RP}) and different coefficients of reward and penalty.

graph	*0.01*		*0.1*	*0.2*	*0.3*	*graph*	*0.01*		*0.1*	*0.2*	*0.3*
	T	*C*	*C*	*C*	*C*		*T*	*C*	*C*	*C*	*C*
DSJC1000.1	312.01	20	20	21	24	*r1000.1*	121.4	20	21	29	38
DSJC1000.5	931.81	85	89	91	97	*r1000.1c*	859.31	98	110	105	113
DSJC1000.9	1035.27	225	225	229	235	*r1000.5*	615.27	233	239	239	245
DSJR500.1	81.53	12	12	13	15	*school1*	113.5	14	14	16	20
DSJR500.1c	211.61	85	86	86	87	*school1-nsh*	94.25	14	15	19	20
DSJR500.5	137.92	121	123	124	127	*latin-square-10*	726.43	98	99	111	118
le450-15a	169.53	15	15	15	15	*flat1000-50-0*	863.14	50	62	53	59
le450-15b	173.21	14	14	16	19	*flat1000-60-0*	927.01	58	58	62	67
le450-15c	191.19	15	17	18	22	*flat1000-76-0*	1042.19	83	85	89	92
le450-15d	243.59	16	19	20	23						

5. Conclusion

In this paper, we proposed an algorithm based on irregular cellular learning automata for solving the graph coloring problem. The proposed algorithm was compared with the algorithms of Caramia, Funabiki and Malaguti. The results of experiments show that the proposed algorithm outperforms the other existing algorithms. We tested the proposed algorithm with different learning parameters and stopping condition also. The results showed that the color-set size decreases as the stopping condition increases. On the other hand, the running time of algorithm increases as the stopping condition increases.

Table 3: Number of colors and running time for graph coloring with the proposed algorithm (L_{RP}) and different threshold.

graph	*pure*		*0.2*		*0.4*		*0.6*		*0.8*	
	T	*C*	*T*	*C*	*T*	*C*	*T*	*C*	*T*	*C*
DSJC1000.1	312.01	20	321.16	20	361.94	20	391.5	20	415.62	20
DSJC1000.5	931.81	85	942.7	84	949.32	84	976.01	84	1011.19	83
DSJC1000.9	1035.27	225	1051.93	225	1082.09	225	1131.59	225	1192.43	225
DSJR500.1	81.53	12	95.27	12	123.25	12	159.03	12	185.41	12
DSJR500.1C	211.61	85	219.5	85	245.43	85	281.71	84	305.16	84
DSJR500.5	137.92	121	152.18	121	178.5	121	192.94	121	237.81	121
le450-15a	169.53	15	175.13	15	192.05	15	237.13	15	281.95	15
le450-15b	173.21	14	181.12	14	205.24	14	214.2	14	263.2	14
le450-15c	191.19	15	201.35	15	231.07	15	289.11	15	304.35	15
le450-15d	243.59	16	253.07	16	265.14	16	281.07	16	297.37	14
r1000.1	121.4	20	137.26	20	151.72	20	162.18	20	192.51	20
r1000.1c	859.31	98	861.17	98	891.5	98	701.13	98	734.49	98
r1000.5	615.27	233	618.49	233	652.18	233	679.12	233	693.23	233
school1	113.5	14	125.31	14	167.31	14	185.53	14	207.9	14
school1-nsh	94.25	14	98.61	14	129.12	14	142.7	14	151.52	14
latin-square-10	726.43	98	731.15	98	782.53	98	809.61	98	813.34	98
flat1000-50-0	863.14	50	873.26	50	891.62	49	922.35	49	973.42	49
flat1000-60-0	927.01	58	932.91	58	982.19	58	1012.73	58	1054.17	58
flat1000-76-0	1042.19	83	1059.33	83	1081.34	83	1127.19	83	1186.09	83

References

[1] M.R. Garey and D.S. Johnson, ”Computers and Intractability: A Guide to the Theory of NP-Completeness”, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Freedman, New York, 1979.

[2] F.T. Leighton, “A Graph Coloring Algorithm for Large Scheduling Problems”, Journal of Research of the National Bureau of Standards, Vol. 84, No. 6, pp. 489-503, 1979.

[3] D. deWerra, “An Introduction to Timetabling”, European Journal of Operational Research, Vol. 19, pp. 151-162, 1985.

[4] F.C. Chow and J.L. Hennessy, “The Priority-based Coloring Approach to Register Allocation”, ACM Transactions on Programming Languages and Systems, Vol. 12, No. 4, pp. 501-536,1990.

[5] A. Gamst, “Some Lower Bounds for a Class of Frequency Assignment Problems”, IEEE Transactions on Vehicular Technology, Vol. 35, No. 1, pp. 8-14, 1986.

[6] T.K. Woo, S.Y.W. Su, and R. Newman Wolfe, “Resource Allocation in a Dynamically Partitionable Bus Network Using a Graph Coloring Algorithm”, IEEE Trans. Commun., Vol. 39, No. 12, pp. 1794-1801, 2002.

[7] J. Akbari Torkestani, and M. R. Meybodi, “Coloring Problem Based on Learning Automata,” Proceedings of International Conference on Information Management and Engineering *(ICIME 2009)*, Malaysia, pp. 643-647, 2009.

[8] M. Caramia and P. Dell'Olmo, “A Fast and Simple Local Search for Graph Coloring”, Proc. of the 3rd Workshop on Algorithm Engineering WAE'99, Lecture Notes in Computer Science, Vol. 1668, pp. 319-313, 1999.

[9] N. Funabiki and T. Higashino, “A Minimal-state Processing Search Algorithm for Graph Coloring Problems”, IEICE Trans. Fundamentals, Vol. E83-A, No. 7, pp. 1420-1430, 1990.

[10] E. Malaguti, M. Monaci, and P. Toth, “A Metaheuristic Approach for the Vertex Coloring Problem”, Technical Report OR/05/3, DEIS University of Bologna, 2005.

[11] M. Asnaashari and M.R. Meybodi, "Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks", Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Volume on Communication, Telecommunication Research Center, Tehran, Iran, May 15-17, 2007.

[12] R. Diestel, "Graph Theory", 3rd Edition, Springer-Verlag, New York, 2005.

[13] Ftp://dimacs.rutgers.edu/pub/challenge/graph/.