Available online at http://sanad.iau.ir/journal/ijim/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol.16, No. 2, 2024, Article ID IJIM-1629, 13 pages
Research Article

Learning Automata Method for Channel Assignment in Wireless Cognitive Sensor Network

M. Gholipour[*] [1], H. Hadian Dehkordi[2], M. Shahriari[3]

[1] Department of Electronic and Computer, Islamic Azad University, Qazvin Branch, Qazvin, Iran,

[2] Department of Mathematics and Science Computer, Amirkabir University of Technology, Tehran, Iran,

[3] Department of Industrial Management, South Tehran Branch, Islamic Azad University, Tehran, Iran

Submission Date: 2016/11/02, Revised Date: 2017/09/12, Date of Acceptance: 2024/05/19

Abstract

One of the main challenges already faced by communication networks is the efficient management of increasing complexity. The recently proposed concept of cognitive networks has appeared as a candidate for addressing this issue. Cognitive networks are put forward in the framework of the evolution of network architectures as novel paradigms to provide autonomous systems in network reasoning to support end-to-end goals. This paper proposes a new method for modifying the spectral efficiency of topology control using the new vertex coloring algorithm in Thomas cognitive networks. This paper proposed a learning automata-based iterative algorithm for approximating the near-optimal solution to the vertex coloring problem in the channel control of the cognitive network. Vertex coloring is a well-known NP-hard optimization problem in graph theory in which each vertex is assigned a color so that no two adjacent vertices have the same color. As the proposed algorithm proceeds, the required number of colors tends to the chromatic number of the graph since the number of channels assigned to radios will be optimal. To show the performance of the proposed algorithm, we compare it with several existing vertex coloring algorithms in terms of the number of iterations and colors required for coloring the graphs and then map this solution to the channel assignment problem. Then, we compare the results of our algorithm with other channel assignment topology control methods. The results show that the proposed algorithm is superior to the others.

Keywords: Cognitive Network, Wireless Sensor Network, Channel Assignment, Graph Coloring, Learning Automat

[*] Corresponding author: Email: Gholipour@gmail.com

1. Introduction

The area of information and communication technologies is one of the fastest-changing areas, with related services and applications having enormous and almost immediate impact on diverse aspects of modern society, including inter-human relations, economy, education, and entertainment. In this respect, the development of reliable and robust yet flexible and future-proof communication infrastructure capable of real-time, secure, and cost-effective delivery of data is of utmost importance to increase the user's perceived quality of life by facilitating human-to-human as well as human-to-machine communication almost anywhere and anytime, providing services such as e-health, e-learning and e-payments. Future networks will be ever more complex, extending towards ubiquitous communications, and will provide a broad range of other services and applications, from remote managing of an intelligent house to advanced real-time navigation systems. Despite the increased complexity, future networks should be easily maintainable, and their capabilities should be continuously improved and upgraded by relying as little as possible on human intervention. To meet this demand, the networking research community proposed a new paradigm for networking: the cognitive network [1-4].

A cognitive system is a complex system with the ability for emergent behavior [5,6]. It processes data over time by performing the following steps: 1) perceive defined situations; 2) learn from defined situations and adapt to their statistical variations; 3) build a predictive model on prescribed properties; and 4) control the situations and do all of these procedures in real time to execute prescribed tasks.

Cognitive networks are motivated by complexity. Particularly in wireless networks, there has been a trend towards increasingly complex, heterogeneous, and dynamic environments [7]. While wired networks can also take on any of these characteristics and are not excluded from potential cognitive network applications because of the internode interactions and the size of the system state space, wireless networks are a natural focus of research on complex networks.

In [1,3], Thomas presented a new method with the cognitive network to minimize the transmission power and spectral impact of a wireless network topology under static and dynamic conditions. Unlike most of the literature on topology control [8,9], he jointly examined lifetime and spectral efficiency objectives. His goal is to establish a distributed framework for minimizing the maximum transmission power and spectral footprint while achieving interference-free connectivity. This paper tries to modify the channel assignment in topology control. We use graph coloring as a model for channel assignment in topology control and propose an iterative algorithm based on learning automata for approximating a near-optimal solution to the vertex coloring problem. Graph coloring [10,11] is a well-known NP-hard optimization problem in graph theory in which each vertex is assigned a color so that no two adjacent vertices have the same color. As the proposed algorithm precedes the number of stages per iteration and so the required number of colors tends to the chromatic number of graphs, using learning automata since the number of vertices that are colored at each stage is maximized. To show the performance of the proposed algorithm we compare it with several existing vertex coloring algorithms [12,14] in terms of the number of iterations and colors required for coloring graphs. The obtained results show the superiority of the proposed algorithm over others.

The rest of this paper is organized as follows: Section 2 describes the cognitive topology control. Section 3 describes the variable action-set learning automaton. Section 4 describes the proposed learning automata-based algorithm. Section 5 evaluates the performance of the proposed algorithm through simulation experiments, and section 6 concludes the paper.

2. Cognitive topology control

Wireless networks are unstructured in the most general sense [6,7]. With increasing programmability, radios can autonomously adapt to their environment, setting transmission power and selecting their frequency of operation. Topology control attempts to harness this programmability to build structure into the wireless network.

Traditionally, the field of topology control has examined power control problems that disregard spectral efficiency or vice-versa. Unlike most of the literature on topology control [8,9], the cognitive network method [2] jointly examines lifetime and spectral efficiency objectives. The goal is to establish a distributed framework for minimizing the maximum transmission power and spectral footprint while achieving interference-free connectivity. To address these spectrum and power control issues, there is a two-phased CN approach in which cognitive elements are distributed on each network radio. CN uses two potential class cognitive elements: TopoPower Control, which controls the transmission power of a radio, and Topo Channel Control, which chooses the transmission channel for a radio. CN aims to establish a distributed framework for interference-free communication between selfish radios. Each cognitive element observes network conditions and then selfishly chooses the reduced power level that still maintains topological connectivity (in the case of TopoPower Control) and then chooses channels that allow interference-free connectivity with all desired receivers (in the case of Top Channel Control). Pursuing these selfish objectives, the network reaches a topology state that minimizes the maximum transmission power, and the number of orthogonal channels required to achieve interference-free connections. This paper tries to modify the channel assignment in cognitive topology control. We use graph coloring as a model for channel assignment in topology control. We propose an iterative algorithm based on learning automata to approximate a near-optimal solution to the vertex coloring problem.

3. Variable Action-Set Learning Automata

A variable action-set learning automaton is one in which the number of actions available at each instant changes with time. It has been shown in [16] that a learning automaton with a changing number of actions is reasonable and ε-optimal when the reinforcement scheme is L_{R-I}. Such an automaton has a finite set of n actions, $\alpha = \{\alpha_1,\alpha_2,\alpha_n\}$. $A= \{A_1,A_2,A_m\}$ denotes the set of action subsets, and $A(k) \subseteq \alpha$ is the subset of all the actions that can be chosen by the learning automaton at each instant k. The selection of the particular action subsets is randomly made by an external agency, according to the probability distribution $q(k)=\{q_1(k),q_2(k),\ldots,q_m(k)\}$defined over the possible subsets of the actions, where $q_i(k)=prob[A(k)=A_i| A_i \in A, 1 \leq i \leq 2^n-1]$.

$\hat{p}(k)\ prob[\alpha(k)=\alpha_i \mid A(k), \alpha_i \in A(k)]$ is the probability of choosing action α_i, conditioned on the event that the action subset $A(k)$ has already been selected and $\alpha_i \in A(k)$. The scaled probability $\hat{p}_i(k)$ is defined as:

$$\hat{p}_i(k) = p_i(k) / K(k) \quad (1)$$

Where $\sum_{a_i \in A(k)} p_i(k)$ Is the sum of the probabilities of the actions in subset A (k), and $p_i(k)=prob[\alpha(k)=\alpha_i]$

The procedure of choosing an action and updating the action probabilities in a variable action-set learning automaton can be described as follows. Let A (k) be the action subset selected at instant k. Before choosing an action, the probabilities of all the actions in the selected subset are scaled as defined in equation (1). The automaton then randomly selects one of its possible actions according to the scaled action probability vector $p\hat{}$ (k). Depending on the response received from the environment, the learning automaton updates its scaled action probability vector. Note that the probability of the available actions is only updated. Finally, the probability vector of the actions of the chosen subset is rescaled as:

$$Pi\ (k+1) = p\hat{}_i\ (k+1).\ K\ (k) \qquad (2)$$

For all $\alpha_i \in A$ (k), [16] has proven the method's absolute expediency and ε-optimality.

4. Proposed Learning Automata-based Graph Coloring

In this section, we propose a learning automat-based approximation algorithm called LAGC for solving the minimum vertex coloring problem and using it for channel assignment. The proposed algorithm is iterative and exploits the variable action-set learning automata to color the graph.

It is mentioned that in the directed graph G (V, E), the set of nodes is V (|V|=n), the edges are E, and arcs e_{ij} in E represent connections from a transmitter to a receiver. In this algorithm, the goal is vertex coloring in which each vertex is assigned a color so that no two adjacent vertices have the same color. The input topology in our problem is a connected graph, which can be obtained from power control. A bidirectional connection exists between two nodes when arcs exist in both directions. G is connected if a bi-directed path (a collection of contiguous bi-directed arcs) exists between any two radios in G. Our algorithm is an iterative algorithm that proposes a valid coloring in each iteration. Valid coloring means that no two adjacent vertices have the same color.

In the proposed algorithm, each node of the graph is first assigned a learning automaton, so a network of learning automata isomorphic to the graph is initially constructed. This network of learning automata can be described by a duple, which $A=\{A_1, A_2, ..., A_n\}$ denotes the set of learning automata corresponding to the vertex-set and $\alpha=\{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n\}$ denotes the action set in which $\forall \alpha_i \in \alpha\ \alpha_i=\{\alpha_i^1, \alpha_i^2, ..., \alpha_i^{r_i}\}$ Defines the set of actions that can be taken by the learning automaton A_i. It should be noted that a learning automaton is associated with a vertex (one-to-one). So, hereafter, vertex v_i may be referred to as learning automaton A_i and vice versa.

As described earlier, the proposed coloring algorithm consists of several iterations, at each of which the graph is thoroughly colored.

In this algorithm, the action set of each learning automaton includes a set of all the potential colors that can be assigned for each node and the nodes that we can reach through the next automata. So, for each node, the size of the action set corresponds with the product of the number of node edges and colors. For example, as shown in Figure 1, if the set of colors is

{a, b, c, d, e}, for node 1, the action set is: , a2, b2, c2, d2, e2, a3, b3, c3, d3, e3, a5, b5, c5, d5, e5}

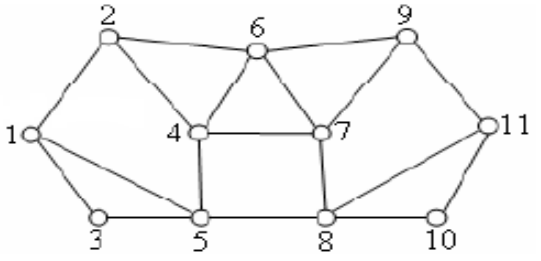

Figure 1- A sample graph

For instance, action a2 (in node 1) means that this node is colored with "a," the next active automata will be node 2. In this case, each action includes node one, and the color "a" will be deleted from the action set of nodes 2 (lines 16, 17, and 18). This avoids assigning the same color to the neighboring nodes, one at a time; each automaton (node) selects one of its actions (line 19) until no uncolored node remains, and finally, we can achieve a valid coloring of the graph (line 23).

This process guarantees that the graph can be legally colored at each iteration. The set of colors used to color the graph is called a color set. After all nodes are colored, the number of colors used in the current iteration |Ct| is compared with the minimum number of colors in previous iterations |χ|. If the cardinality of the selected color set |Ct| is less than |χ|, LAGC rewards the actions chosen by all the activated learning automata (lines 24,25), sets χ, and then removes all the actions, including unused colors from action sets of all nodes and colors used in this iteration become optimal color set (line 28). To ensure we have a valid coloring, assume the cardinality of colors at the first iteration equals the degree of the graph plus one.

There are enormous action sets for each automaton at the preliminary iterations, and this is why unused color is removed from action sets, improving the speed of convergence to a near-optimal solution.

If the number of colors used in each iteration is equal to those in previous iterations, the algorithm rewards selected actions and doesn't change action sets of automata in each node (lines 29, 30). Therefore, in the forthcoming iterations, these actions are selected with a higher probability as the choice probability of the other actions decreases. It should be mentioned that at the end of each iteration and after each valid coloring process, all nodes will be uncolored and ready to enter the new iteration (line 32). In this algorithm, the learning scheme LR-P is used to update the action probability vectors in which the parameters for reward and penalty are equal; in this algorithm, the actions that form a smaller color set are rewarded.

Coloring will certainly be valid during the first iteration because the cardinality of colors is numerous. As the algorithm proceeds, the cardinality of colors will be less, and the action sets will be significantly smaller. This is because the maximum number of required colors decreases as the algorithm approaches the end. Indeed, after several iterations, the network of learning automata converges to an optimal action selection strategy at each iteration. Here, the kth iteration of LAGC is completed or acceptable convergence to a solution, meeting the stopping condition of the algorithm. The pseudo-code of the proposed coloring algorithm is shown in Figure 2.

Algorithm LAGC

1. **Input:** Graph *G (V, E)*
2. **Input:** $\chi = \{1,2,3,...,m+1\}, m = \text{maximum degree in G}$
3. **Output:** Chromatic number $|\chi|$
4. **Assumptions:**
5. Assign a learning automaton A_i to each vertex v_i
6. Let α_i denotes the action-set of automaton A_i
7. Let C_t denotes used colors in iteration t
8. **Begin Algorithm**
9. $t \leftarrow 0, \chi = \{\alpha_1, \alpha_2, ..., \alpha_{m+1}\}$
10. **For** each $v_i \in V$ **do**
11. $\alpha_i = \chi \times (neigbors\ \ of\ \ v_i)$
12. **Repeat**
13. Vertex v_i randomly is selected
14. $c_t \leftarrow 0$
15. **Repeat**
16. **For** each action $a_k j \in \alpha_i$ **do**
17. **If** (color(neighbors of v_i) = k) **then**
18. Deactivate $a_k j$
19. Automaton A_i (corresponding to v_i) randomly chooses action a_kj from its active actions
20. Color(v_i) $\leftarrow a_k$
21. $c_t \leftarrow c_t + \{a_k\}\quad v_t \leftarrow v_t - \{v_i\}$
22. $v_i \leftarrow v_j$
23. **Until** $v_t \neq 0$
24. **If** $|c_t| < |\chi|$ **then**
25. All learning automata reward their chosen actions
26. **For** each $v_i \in V$ **do**
27. $\alpha_i \leftarrow \alpha_i - [(\chi - c_t) \times (Neighbors\ of\ \ v_i)]$
28. $\chi \leftarrow c_t$
29. **Else**
30. $v_t \leftarrow v$
31. Discolor all of nodes
32. $t \leftarrow t+1$
33. **Until** Stop Condition = **TRUE**
34. **Return** $|\chi|$
35. **End Algorithm**

Figure 2. The pseudo code of proposed algorithm

5. Numerical Results

We have conducted several simulation experiments to study the proposed algorithm's efficiency. In these experiments, the proposed coloring algorithm is tested on a subset of hard-to-color benchmarks [17] like Fullln s [18], Queen [18], Insertions [18], Leighton [19], and Wap [20]. The performance of the proposed algorithm is measured both in terms of the number of iterations and colors required for coloring the benchmarks and compared with those of LAVCA [12], GLS [14], and AMACOL [13]. In all experiments presented in this paper, the reinforcement scheme by which the action probability vectors are updated is L_{R-I}, and the reward and penalty parameters (i.e., *a* and *b*) are 0.15 and 0. In these experiments, threshold *P* is set to 0.95. So, the proposed algorithm stops if all the activated learning automata choose their action with a probability higher than 0.95 or when they achieve maximum iteration. The results are summarized in Tables 4, 5, and 6. In these tables, the first column includes the graph name, the chromatic number is in the second column, and the next columns include the number of colors required for coloring the graph (CN) and the running time of each algorithm (RT). The programming language used for experiences is C#. Table 1 represents the simulation parameters in this study.

Table1. Simulation Parameters

Simulator name	C# & NS2
Node Placement	Random
Application Type	Event Driven
Frequency band	2.4Ghz
Link layer transmission rate	250kbps
Simulation Time	400 sec
Area Size	100m *100m
Network Architecture	Homogeneous, Flat
Number of nodes	Variable
Average Node degree	Variable
Communication Radius	5m
Packet Size	25byte
Buffer Size	10 Packets
Learning automata type	L_{R-I}
Penalty parameter	0
Reward parameter	0.15

In the second part of this section, we used the NS2 network simulator to simulate power control and channel assignment in sensor network topology and evaluate the cardinality of channel assignment for a wireless sensor network with different numbers of radios. The result of our method is compared with Thomas' method [1], and the tables summarize the LAVCA, GLS, and AMACOL algorithms simulated in sensor networks.

Table2. The characteristics of Fullln, Insertion and Queen graphs

Graph	Number of Nodes	Number of Edges	Density	Chromatic Number
1-Fullln-3	30	100	0.23	4
2-Fullln-3	52	201	0.15	5
1-Insertions-4	67	232	0.10	5
2-Insertions-3	37	72	0.11	4
3-Insertions-3	56	110	0.07	3
Queen7-7	49	476	0.40	7
Queen6-6	36	290	0.46	7
Queen5-5	25	160	0.53	5

Fullln and Leighton benchmark graphs are the first and second graphs on which the proposed algorithm is tested. The characteristics (i.e., density and number of vertices and edges) of Fullln, Insertion, and Queen graphs are in Table 2, and the specifications of Leighton benchmark graphs are given in Table 3. The number of nodes in Leighton is 450 and denotes as le450_xy, that **x** represents the optimal number of colors for coloring (chromatic number), and **y** is the density (the probability of Connecting every pair of nodes in the graph) of the edges. Table 3 includes 12 graph families with density 0.17, 0.1, 0.08, 0.06.

Table3. The characteristics of Leighton graphs

Graph	Number of Nodes	Number of Edges	Density
Le450_5a	450	5714	.06
Le450_5b	450	5734	.06
Le450_5c	450	9803	.10
Le450_5d	450	9757	.10
Le450_15a	450	8168	.08
Le450_15b	450	8169	.08
Le450_15c	450	16680	.17
Le450_15d	450	16750	.17
Le450_25a	450	8260	.08
Le450_25b	450	8263	.08
Le450_25c	450	17343	.17
Le450_25d	450	17425	.17

The third class of benchmarks we consider includes Apographs, which are used in the design of transparent optical networks. In this class, graphs have many vertices between 905 and 5231, and all instances have a clique of size 40. The characteristics (i.e., density, and the number of vertices and edges) of Wap benchmark graphs are given in Table 4.

Table4. The characteristics of Waographs

Graph	Number of Nodes	Number of Edges	Density	Chromatic Number
Wap01a	2368	110871	.04	42
Wap02a	2464	111742	.04	41
Wap03a	4730	286722	.03	44
Wap04a	5231	294902	.02	43
Wap05a	905	43081	.11	41
Wap06a	947	43571	.10	41
Wap07a	1809	103368	.06	42
Wap08a	1870	104176	.06	42

Comparing the results of algorithms, we can see that when the number of nodes is low in all cases, all algorithms achieve the chromatic number (the minimum color required for coloring). Comparing the results of the number of iterations in all algorithms, we find that in all cases (when the density is high or low), the proposed algorithm converges to an optimal solution faster than other algorithms. For graphs with high density (*i.e.*, Queen Graphs), GLS, LAVCA, and AMACOL have the next places respectively. Still, for graphs with low density (i.e., Fulllns, Insertions), AMACOL converges to an optimal solution faster than LAVCA. However, the GLS algorithm has the second place in convergence speed (Table 5).

Table 5. Comparing the efficiency of proposed algorithm and other coloring algorithms in the Fulllns, Insertions and Queen Graphs

		AMACOL	AMACOL	GLS	GLS	LAVCA	LAVCA	LAGC	LAGC
Graph	Best	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations
1-Fulllns-3	4	4	158	4	148	4	169	**4**	**141**
2-Fulllns-3	5	5	170	5	158	5	175	**5**	**146**
1-Fulllns-4	5	5	186	5	174	5	191	**5**	**147**
2-Insertions-3	4	4	159	4	148	4	166	**4**	**136**
3-Insertions-3	4	4	183	4	171	4	196	**4**	**165**
Queen7-7	7	7	173	7	156	7	181	**7**	**155**
Queen6-6	7	7	169	7	154	7	162	**7**	**152**
Queen5-5	5	5	163	5	151	5	157	**5**	**151**

The next class of benchmark graphs on which the proposed algorithm is tested is Wap, a set of large random graphs. When the number of nodes is large, the proposed algorithm always does not converge to a chromatic number, which is its weakness (Table 6).

Table 6. Comparing the efficiency of proposed algorithm and other coloring algorithms in Wap Graphs

Graph	Best	AMACOL	AMACOL	GLS	GLS	LAVCA	LAVCA	LAGC	LAGC
		Number of Colors	Number of Iterations	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations
Wap01a	42	46.5	1500	42	1410	44.5	1500	**46**	**1500**
Wap02a	41	46.5	1500	41	1417	43	1500	**46.5**	**1500**
Wap06a	41	45	1500	41	1391	42.5	1500	**44.5**	**1500**

Table 7 shows the results reported for our algorithm compared with other algorithms in two instances of Leighton graphs. We find that, in almost all cases, LAGC outperforms others in terms of the number of iterations and the number of colors. It is necessary to mention that these graphs have many nodes, but the proposed algorithm is better than other algorithms because of its low density.

Table 7. Comparing the efficiency of proposed algorithm and other coloring algorithms in two instances of Leighton Graphs

		AMACOL	AMACOL	GLS	GLS	LAVCA	LAVCA	LAGC	LAGC
Graph	Best	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations	Number of Colors	Number of Iterations
Le450_15a	15	17.5	900	15	373	19.5	900	**15**	**315**
Le450_15b	15	18	900	15	389	20	900	**15**	**323**

Comparing the results reported in Table 8, we find that our algorithm converges to the optimal solution with many iterations in Leighton graphs with high density. GLS and LAVCA algorithms can find the optimal solution with lower iterations.

Table 8. Comparing the efficiency of proposed algorithm and other coloring algorithms in two another instances of Leighton Graphs

Graph	Best	AMACOL Number of Colors	AMACOL Number of Iterations	GLS Number of Colors	GLS Number of Iterations	LAVCA Number of Colors	LAVCA Number of Iterations	LAGC Number of Colors	LAGC Number of Iterations
Le450_15c	15	18	900	15	421	15	573	**15**	**662**
Le450_15d	15	18.5	900	15	438	15	595	**15**	**687**

As noted above, we used the NS network simulator to simulate power control and then channel assignment in topology and evaluate the cardinality of channel assignment for a wireless sensor network with different numbers of radios. The result of our method compares with Thomas' method [1], and the LAVCA, AMACOL, and GLS algorithms are summarized in Tables 9 and 10.

Table 8 shows the results reported for our algorithm compared with others in the case of an average maximum number of channel assignments in 100 iterations. The sensor network is distributed randomly with 5, 25,45,65,85, and 105 radios. For the networks with 5, 25, 45, and 65, the LAGC algorithm always assigns the smallest channels to radios, but when the number of radios increases, the performance of the proposed algorithm decreases gradually.

Table 9. Comparing the efficiency of the proposed algorithm and other coloring algorithms with different radios in the sensor network

Average Number of Assigned Channels for 100 Iteration					Number of Radios
AMACOL	GLS	LAVCA	LAGC	DIA	
4.7	3.15	3.16	**3.13**	3.15	5
6.51	5.32	5.46	**5.31**	5.58	25
8.33	6.46	6.50	**6.47**	6.90	45
9.53	7.70	8.12	**7.65**	7.84	65
5.37	5.9	5.94	**6.00**	6.21	85
8.34	6.28	5.10	**6.3**	6.44	105

Table 10 shows that when connectivity is low, the LAGC algorithm is the best method for determining the average maximum number of channels. However, when the percentage of connectivity increases, our methods require more than 100 iterations to converge to the optimal solution. As the percentage of connectivity increases, the action set also increases, which requires an increase in the number of iterations.

Table 10. Comparing the efficiency of proposed algorithm and other coloring algorithms with 100 random radios in the environment

Average Number of Assigned Channels for 100 Iteration and 100 Radios				Density Percentage
AMACOL	GLS	LAVCA	LAGC	
8.11	31.6	8.15	**6.29**	5%
8.12	6.44	7.74	**6.33**	10%
8.12	6.81	7.36	**5.70**	15%
8.18	6.85	6.90	**6.84**	25%
8.20	6.59	6.98	**6.61**	35%
8.22	6.86	7.84	**7.3**	55%

6. Conclusion

One of the main challenges already faced by communication networks is the efficient management of increasing complexity. The recently proposed concept of cognitive networks has appeared as a candidate for addressing this issue. This paper proposes a new method for modifying the spectral efficiency of topology control in cognitive networks using a new coloring algorithm in Thomas cognitive networks. The coloring problem is a combinatorial optimization problem in which a color is assigned to each graph node so that no two adjacent nodes have the same color. The minimum coloring is an NP-hard problem, and most algorithms have been proposed to solve it. This paper proposed an approximation algorithm for solving the minimum coloring problem based on learning automata. The proposed algorithm iteratively finds different possible colorings of the graph. The proposed algorithm guarantees that the graph is legally colored at each iteration. As the proposed algorithm approaches the end, each node learns to select one of its neighborhoods and colors. Therefore, as the proposed algorithm proceeds, the required number of colors tends to be the chromatic number of the graph. To show the performance of the proposed algorithm, we compared it with several vertex coloring algorithms in terms of the time and the number of colors required for coloring the graphs. The results show that the proposed algorithm is superior to the others. Given the large application of the graphing algorithm, use it in other complex issues will be some of the future works.

References

[1] Thomas, R. W. "Cognitive Networks, Ph. D. Dissertation." Computer Engineering, Virginia Polytechnic Institute and State University (2007).

[2] Clark, David D., et al. "A knowledge plane for the internet." Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. ACM, 2003.

[3] R.W. Thomas, L.A. DaSilva, A.B. MacKenzie, "Cognitive networks", in: Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, Baltimore, MD, USA, November 8–11, 2005.

[4] Gholipour, M., A. T. Haghighat, and M. R. Meybodi. "Congestion avoidance in cognitive wireless sensor networks using TOPSIS and response surface methodology." Telecommunication Systems (2017): 1-19.

[5] Liang, C., and F. Richard Yu. "Wireless network virtualization: A survey, some research issues and challenges." IEEE Communications Surveys & Tutorials 17.1 (2015): 358-380.

[6] Gholipour, M., and Meybodi, M.R., "LA-Mobicast: A learning automata based distributed protocol for mobicast in sensor networks." Proceedings of CSICC2007 (2007): 1154-1161.

[7] Gholipour, M., Toroghi Haghighat, A., and Mohammad Reza Meybodi. "Hop by Hop Congestion Avoidance in wireless sensor networks based on genetic support vector machine." Neurocomputing 223 (2017): 63-76.

[8] Wu, K.-D., and Liao, W., "On constructing low interference topology in multihop wireless networks," in Proc. of IEEE ICC 2006, 2006.

[9] Santi, P., "Topology control in wireless ad hoc and sensor networks," ACM Computing Surveys (CSUR), vol. 37, pp. 164 – 194, March 2005.

[10] Elhassan, A., "Graph-coloring for course scheduling—A comparative analysis based on course selection order." e-Technologies and Networks for Development (ICeND), 2014 Third International Conference on. IEEE, 2014.

[11] Golovach, Petr A., et al. "A survey on the computational complexity of coloring graphs with forbidden subgraphs." Journal of Graph Theory 84.4 (2017): 331-363.

[12] Torkestani, J.A., Mohammad Reza Meybodi, "A New Vertex Coloring Algorithm Based on Variable Action-set Learning Automata", Journal of Computing and informatics, Vol. 29, pp. 447–466, 2010.

[13] Galinier, P., A. Hertz, and N. Zufferey, "An Adaptive Memory Algorithm for the K-coloring Problem," Discrete Applied Mathematics, Vol. 156, pp. 267–279, 2008.

[14] Voudouris, C., and E. P. K. Tsang, "Guided local search," In Handbook of Metaheuristics, F. Glover (Eds.), Kluwer Academic Publishers, USA, pp. 185-218, 2003.

[15] Thathachar, M.A.L., and B. R. Harita, "Learning automata with changing number of actions,"IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMG17, pp. 1095-1100, 1987.

[16] Website address: https://sites.google.com/site/graphcoloring.

[17] Bergman, D., J. N. Hooker. "Finite-Domain Cuts for Graph Coloring" NSF grant in School of Business, Carnegie Mellon University,2012.

[18] Leighton, F.T., "A graph coloring algorithm for large scheduling problems," Journal of Research of the National Bureau of Standards, Vol. 84, pp. 489–505, 1979.

[19] Caramia, M., and P. Dell'Olmo, "Embedding a novel objective function in a two-phased local search for robust vertex coloring," European Journal of Operational Research, Vol. 189, pp. 1358–1380, 2008.