Expert Systems with Applications 38 (2011) 9237–9247

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A cellular learning automata-based algorithm for solving the vertex coloring problem

Javad Akbari Torkestani [a,*], Mohammad Reza Meybodi [b,c]

[a] Department of Computer Engineering, Islamic Azad University, Arak Branch, Arak, Iran
[b] Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
[c] Institute for Studies in Theoretical Physics and Mathematics (IPM), School of Computer Science, Tehran, Iran

ARTICLE INFO

Keywords:
Vertex coloring problem
Learning automata
Cellular learning automata

ABSTRACT

Vertex coloring problem is a combinatorial optimization problem in which a color is assigned to each vertex of the graph such that no two adjacent vertices have the same color. Cellular learning automata (CLA) is an effective probabilistic learning model combining cellular automata and learning automata. Irregular cellular learning automata (ICLA) is a generalization of cellular learning automata in which the restriction of rectangular grid structure in traditional CLA is removed. In this paper, an ICLA-based algorithm is proposed for finding a near optimal solution of the vertex coloring problem. The proposed coloring algorithm is a fully distributed algorithm in which each vertex chooses its optimal color based solely on the colors selected by its adjacent vertices. The time complexity of the proposed algorithm is computed for finding a $\frac{1}{1-\epsilon}$ optimal solution of the vertex coloring problem in an arbitrary graph. To show the superiority of our proposed algorithm over the existing methods, simulation experiments have been conducted. The obtained results show that the proposed algorithm outperforms the others in terms of the required number of colors and running time of algorithm.

1. Introduction

Graph coloring problem is a classical combinatorial optimization problem in graph theory. Graph coloring show up in an incredible variety of forms, for example vertex coloring, multicoloring, bandwidth coloring, list coloring, set coloring, T-coloring, lambda coloring, alpha coloring, etc. Graph coloring is widely used in real life applications such as computer register allocation (Chaitin et al., 1981), air traffic flow management (Barnier & Brisset, 2002), timetabling (Carter, Laporte, & Lee, 1996; de Werra, 1985; Lewis & Paechter, 2007; Thompson & Dowsland, 1998), scheduling (Leighton, 1979), frequency assignment (Gamst, 1986), and light wavelengths assignment in optical networks (Zymolka, Koster, & Wessaly, 2003). Graph coloring is a promising approach for channel assignment in computer networks. The minimum coloring is a well-known NP-hard problem for general graphs (Karp, 1972).

Vertex coloring problem is a well-known coloring problem in which a color is assigned to each vertex of the graph. A legal vertex coloring of graph $G\ \langle V,E\rangle$, where $V(G)$ is the set of $|V| = n$ vertices and $E(G)$ is the edge set including $|E| = m$ edges, is to assign distinct colors to each vertex of the graph in such a way that no two endpoints of any edge are given the same colors. Vertex coloring problem can be modeled by a quadruple $\langle V, E, C, F\rangle$, where V denotes the vertex-set of graph G, E denotes the edge-set of graph G, C denotes the set of colors assigned to the vertices, and $F{:}V \rightarrow C$ is the coloring function that assigns a color to each vertex such that $F(v_i) \neq F(v_j)$ for every $e_{(i,j)} \in E$. 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 whether or not the graph is $\mathcal{P}$-colorable, and is called $\mathcal{P}$-coloring problem. Graph $G\ \langle V, E\rangle$ is $\mathcal{P}$-colorable, if it can be legally colored with at most $\mathcal{P}$ different colors. The chromatic number $\chi(G)$ is the minimum number of colors required for coloring the graph, and a graph is G said to be $\mathcal{P}$-chromatic, if $\chi(G) = \mathcal{P}$. A minimum coloring G of is a legal coloring in which the smallest number of colors (i.e., chromatic number) to be assigned to the vertices. The minimum coloring problem is formally an NP-hard problem for general graphs as determining the chromatic number is known to be NP-hard (Karp, 1972). It is shown in Garey and Johnson (1979) that the decision problem of graph $\mathcal{P}$-colorability is an NP-complete problem for $\mathcal{P} \geqslant 3$, and can be solved in polynomial time otherwise.

In this paper, we propose an irregular cellular learning automata-based algorithm for solving the vertex coloring problem. In this algorithm, the input graph is modeled by an ICLA in which

* Corresponding author.
E-mail addresses: j-akbari@iau-arak.ac.ir (J. Akbari Torkestani), mmeybodi@aut.ac.ir (M.R. Meybodi).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.01.098

the restriction of rectangular grid structure in traditional CLA is removed. To do this, each vertex of the graph is associated with a cell of cellular learning automata and then each cell is equipped with a learning automaton. Learning automata are independently activated and choose their colors. Due to the distribution of computation in cellular learning automata, the proposed vertex coloring algorithm is fully distributed and locally run at each cell in dependent of the other cells. The proposed algorithm is composed of a number of stages, and at each stage a coloring is locally found for each cell and its neighbors. The selected coloring is penalized, if the coloring is illegal or its number of colors is larger than that of the best coloring found so far. Otherwise, the selected coloring is rewarded. As the proposed algorithm proceeds, learning automata learn how to select the colors so that the graph can be thoroughly and legally colored with the minimum number of colors. In this paper, we compute the running time of the proposed algorithm for finding a $\frac{1}{1-\epsilon}$ optimal solution to the vertex coloring problem in an arbitrary graph. We also compare the results of the proposed algorithm with those of CHECKCOL (Caramia, Dellolmo, & Italiano, 2006), GLS (Voudouris & Tsang, 2003), ILS (Lourenco, Martin, & Stutzle, 2002), TPA (Caramia & Dell'Olmo, 2008) and AMACOL (Galinier, Hertz, & Zufferey, 2008) which are the best-known existing vertex coloring methods. The obtained results show the significant superiority of the proposed coloring algorithm over the other methods in terms of the number of colors and running time.

The structure of the rest of the paper is as follows. In the next section, an overview of the vertex coloring algorithms is presented in a nutshell. In Section 3, learning automata and cellular learning automata are briefly reviewed. The proposed cellular learning automata based algorithm is introduced in Section 4. Section 5 is devoted to the analysis of the time complexity of the proposed algorithm. The performance of the proposed vertex coloring algorithm is studied through the computer simulation in Section 6. In this section, the results of our proposed algorithm are also compared with those of the well-known coloring methods. Section 7 represents the concluding remarks.

2. Related work

Due to the NP-hardness of the vertex coloring problem for general graphs, the exact algorithms (Beigel & Eppstein, 2005; Byskov, 2004, 2005; Lawler, 1976) can only be applied on small graphs, while very large graphs often arise in a variety of applications. On the other hand, in realistic applications, it often suffices to find a near optimal coloring of the graph. Hence, a host of polynomial time approximation algorithms have been proposed for finding the near optimal solutions to the coloring problem. The approximation approaches reported in the literature can be classified as local search approaches (Blchliger & Zufferey, 2008; Caramia et al., 2006; Caramia & Dellolmo, 2008; Galinier & Hertz, 2006; Mabrouk, Hasni, & Mahjoub, 2008; Prestwich, 2008; Vredeveld & Lenstra, 2003; Hertz & Werra, 1987), genetic algorithms (Fleurent & Ferland, 1996), fuzzy-based optimizations (Asmuni, Burke, Garibaldi, McCollum, & Parkes, 2009; Munoz, Ortuno, Ramirez, & Yanez, 2005), evolutionary algorithms (Dobrev, Schröder, Sykora, & Vrto, 2000; Eiben, Vanderhauw, & Vanhemert, 1998; Galindier & HAO, 1999; Malaguti & Toth, 2008), simulated annealing methods (Chams, Hertz, & de Werra, 1987; Thompson & Dowsland, 1998), ant colony-based approaches (Bui, Nguyen, Patel, & Phan, 2008; Dowsland & Thompson, 2008), Markov chain approaches (Cooper, Dyer, & Frieze, 2001), neural network approaches (Talavan & Yanez, 2008) and so on. In the remaining of this section, the exact and approximation vertex coloring algorithms are briefly described.

2.1. Exact algorithms

The exact vertex coloring algorithms aim at finding an exact solution to legally and thoroughly color the graph. Since the minimum vertex coloring is an NP-hard problem, exact algorithms proposed for vertex coloring are solely able to color the small instances with up to 100 vertices for random graphs. In these algorithms, finding an exact solution for coloring the hard-to-color graphs in a reasonable time is impossible. All known exponential-time algorithms to compute the chromatic number of a graph (and an optimal coloring) need exponential memory, more precisely $O(2n)$ memory, and they all are based on a dynamic programming approach and the use of maximal independent sets. The first one had been published by Lawler (1976). The running time of this algorithm $O(1+\sqrt[3]{3})^n = O(2.4423^n)$. This algorithm had not been improved for 25 years. Then the combination of improved upper bounds on the number of maximal independent sets of size at most k and changes in the way to fill the table of subsolutions in the dynamic programming algorithm led to better algorithms. Eppstein established an

$$O\left(\frac{4}{3}+\frac{3^{\frac{4}{3}}}{4}\right)^n = O(2.4151^n) \quad (1)$$

time algorithm. Finally Byskov provided an $O(2.40231^n)$ algorithm to compute the chromatic number of a graph. Faster algorithms exist when the number of colorings is fixed. For small fixed numbers of colors, faster algorithms are known. The currently best known bounds for c – coloring are: $O(1.3289^n)$ for $c = 3$ (Beigel & Eppstein (2005)), $O(1.7504^n)$ for $c = 4$ (Byskov (2004)), $O(2.1020^n)$ for $c = 5$ (Byskov and Eppstein, see (Byskov, 2005)), and $O(2.3289^n)$ for $c = 6$ (Byskov (2004)). Each of these algorithms uses polynomial memory. Byskov (2005) also give an algorithm using $O(2n)$ memory and $O(2.1809^n)$ time for 6-coloring.

2.2. Approximation algorithms

For large graphs, the exact solutions cannot be obtained in a reasonable time, so the exact algorithms can only be applied on small graphs. Due to the fact that in many applications it often suffices to find a near optimal coloring of the graph, the approximation algorithms seem to be feasible. Approximation algorithms find a near optimal solution in a reasonable time (polynomial time) for coloring the graph. Different approximation approaches have been proposed for vertex coloring in the literature. In the rest of this section, we briefly review some well-known approximation vertex coloring methods.

2.2.1. Local search

Local search is a metaheuristic for solving computationally hard optimization problems. Local search can be used on problems that can be formulated as finding a solution maximizing (or minimizing) a criterion among a number of candidate solutions. Vertex coloring is a famous hard-to-computation problem that can be solved by local searches. Caramia and Dell'Olmo (2008) also proposed a two-phased local search for vertex coloring. The algorithm alternately executes two closely interacting functionalities, namely, a stochastic and a deterministic local search. The stochastic phase is based on a biased random sampling in which the feasible colorings are iteratively constructed. In deterministic phase, each vertex is assigned to the color which causes the lowest increase of the solution penalty. In this algorithm, the objective function tries to minimize the penalty function. Caramia et al. (2006) proposed a priority-based local search algorithm, called CHECKCOL, in which the running time of algorithm decreases by avoiding unnecessary searches in large portions of the graph without making any

progress in the solution. To do this, they introduced the notion of checkpoint, and forced the algorithm to stop at certain steps, to release all of its memory, and to start a new local search. Furthermore, in this algorithm, each vertex of the graph is dynamically assigned a priority. These priorities are used to define a new and more effective long term memory scheme, which is integrated with the short term memory scheme implied by the checkpoints. The concept of checkpoint along with the priority has a significant impact on the solution quality, cache consciousness, and running time of algorithm. An iterated local search algorithm (ILS) was proposed by Lourenco et al. (2002) to solve the graph coloring problem. The proposed algorithm is based on a randomized walk in the space of the local optima. This walk is built by iteratively perturbing a locally optimal solution, next applying a local search algorithm to obtain a new locally optimal solution, and finally using an acceptance criterion for deciding from which of these solutions to continue the search. Voudouris and Tsang (2003) proposed a Guided Local Search (GLS), which is a meta-heuristic search method, to solve the combinatorial optimization problems. A GLS is a stochastic local search method in which the evaluation function is modified so as to escape from the local optima and plateaus. Chiarandin and Stutzle (2007) applied GLS to solve the graph coloring problem. In this method, the objective function is modified using a specific scheme, when the local search algorithm settles in a local optimum.

2.2.2. Tabu search

Tabu search is a mathematical search method by which the combinatorial optimization problems can be solved. Tabu search uses a neighborhood search procedure to iteratively move from a solution to a neighbor solution, until some stopping criterion is satisfied. Several tabu search algorithms have been proposed for graph coloring in the literature. In Hertz and Werra (1987), Hertz and Werra proposed a tabu search algorithm in which a partition of the vertices of the graph is maintained at each iteration. In this algorithm, a different color is assigned to each block of the partition, which is not always guaranteed to be an independent set. Therefore, this algorithm works with the solutions which are not necessarily feasible. At each iteration, a sample of neighbors of each given configuration is generated. The set of neighbors generated for each vertex is restricted by a tabu list which prevents the algorithm from getting stuck in local optima. Caramia and Dell'Olmo (1999) proposed a local search algorithm, called HCD, based on tabu search. The basic idea behind HCD was to make use of tabu concepts without explicitly representing tabu lists. Instead, a dynamic assignment of priorities to the vertices in the graph performed the same task, avoiding repetitions in subsequent moves of the algorithm. They showed in Caramia and Dell'Olmo (1999) the experimental gain of HCD over tabu search.

2.2.3. Evolutionary algorithms

An evolutionary algorithm is a generic population-based meta-heuristic optimization method. Evolutionary algorithms use the mechanisms inspired by the biological evolution. In evolutionary algorithms, the solutions with high fitness are selected to produce the next generations. Galindier and HAO (1999) proposed a hybrid evolutionary algorithm called HEA for graph coloring. HEA starts with a population P of candidate solutions, which is initialized by using the DSATUR (Brelaz, 1979) construction heuristic restricted to k-colors, and then iteratively generates new candidate solutions by first recombining two members of the current population that are improved by local search. For the recombination, the greedy partition crossover (GPX) (Galindier & HAO, 1999) is used. Starting with two candidate partitionings (parents), GPX generates a candidate solution (offspring) by alternately selecting color classes of each parent. The new candidate partitioning returned by GPX is then improved by tabu search, and is inserted in the population replacing the worse parent. The population is re-initialized if the average distance between colorings in the population falls below a threshold of 20. An adaptive algorithm, called AMACOL, was proposed by Galinier et al. (2008) for the solution of the graph coloring problem. The adaptive memory algorithm is a hybrid evolutionary heuristic in which a central memory is used to store the stable sets that originate from the colorings generated during the previous stages of the search. On each generation, a randomized greedy set covering heuristic is used to find a set of color classes that covers the set of vertices of the graph. This covering is transformed into a coloring in a straightforward way. Then, an iterative neighborhood technique is applied to the coloring. Eventually, the central memory is updated by using the color classes of the new obtained coloring.

Genetic algorithms are a particular class of evolutionary algorithms in which the techniques inspired by the evolutionary biology such as inheritance, mutation, selection, and crossover are used. The fitness function is defined over the genetic representation and measures the quality of the represented solution. Genetic algorithms randomly initialize a population of the solutions and then improve it through repetitive application of biologic operators. Fleurent and Ferland (1996) were the first to experiment a genetic local search algorithm for coloring the graphs. Like the other genetic algorithms, the genetic coloring algorithm proposed by Fleurent and Ferland uses a population of solutions and a crossover operator, but the random mutation operator of the genetic algorithm is replaced by a local search operator here. Indeed, they hybridize a genetic algorithm with tabu search. They use a union crossover where each vertex in the child inherits its color from one of the parents, according to which causes the smallest increase in nearby conflicting edges. Fleurent and Ferland improved their solutions by identifying k-independent sets first and then using their procedure on the remaining vertices. Dorne and Hao (1998) presented a new genetic local search algorithm for graph coloring problem. The proposed algorithm introduces an original crossover called UIS crossover based on the notion of the union of independent sets. The new crossover is combined with a powerful local search operator (tabu search). The resulting hybrid algorithm allows us to improve on the best known results for some large benchmarks. Glass and Prügel-Bennett (2003) proposed a genetic graph coloring algorithm based on an exponential permutation neighborhood in which a subgraph of the graph vertices is selected and the colors within the sub-graph are permuted in such a way as to minimize the number of clashes. They improved the performance of the graph coloring algorithm proposed by Galinier and Hao's algorithm (Galinier & Hao, 1997) using a steepest descent method. In the proposed method, as the problem of finding the optimal point in the neighborhood can be modeled as a linear assignment problem, the neighborhood can be searched in polynomial time.

3. Theory of automata

In this section, cellular automata (CA) and learning automata (LA) are first introduced in brief. Then cellular learning automata (CLA) is presented as a combination of cellular automata and learning automata. Finally, irregular cellular learning automata (ICLA) in which the restriction of the rectangular grid structure in traditional cellular learning automata is removed is presented.

3.1. Cellular automata

Cellular automata are mathematical models for systems consisting of large number of simple identical components with local

interactions. CA is a non-linear dynamical system in which space and time are discrete. It is called cellular because it is made up of cells like points in a lattice or like squares of checker boards, and it is called automata because it follows a simple rule (Fredkin, 1990). The simple components act together to produce complicated patterns of behavior. Cellular automata perform complex computations with a high degree of efficiency and robustness. They are especially suitable for modeling natural systems that can be described as massive collections of simple objects interacting locally with each other (Mitchell, 1996; Packard & Wolfram, 1985). Informally, a d-dimensional CA consists of an infinite d-dimensional lattice of identical cells. Each cell can assume a state from a finite set of states. The cells update their states synchronously on discrete steps according to a local rule. The new state of each cell depends on the previous states of a set of cells, including the cell itself, and constitutes its neighborhood (Kari, 1990). The state of all cells in the lattice is described by a configuration. A configuration can be described as the state of the whole lattice. The rule and the initial configuration of the CA specify the evolution of CA that tells how each configuration is changed in one step.

3.2. Learning automata

A learning automaton (Narendra & Thathachar, 1989; Thathachar & Sastry, 1997) is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action-set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of the inputs $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$, denotes the set of the values that can be taken by the reinforcement signal $c \equiv \{c_1, c_2, \ldots, c_r\}$, and denotes the set of the penalty probabilities, where the element is associated with the given action α_i. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non stationary environment. The environments depending on the nature of the reinforcement signal $\underline{\beta}$ can be classified into P -model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval [0,1] can be taken by the reinforcement signal. Such an environment is referred to as P-model environment. In S-model environments, the reinforcement signal lies in the interval [0,1]. The relationship between the learning automaton and its random environment has been shown in Fig. 1.

Learning automaton has shown to perform well in systems where incomplete information about the environment exists. Learning automaton is also proved to be useful in complex, dynamic and random environments with a large amount of uncertainties. Learning automata have a wide variety of applications in combinatorial optimization problems (Akbari Torkestani & Meybodi, in press-b, 2010a, 2010c) and computer networks (Akbari Torkestani & Meybodi, in press-a, in press-c, in press-d, 2010b, 2010d, 2010e, 2010f). Learning automata can be classified into two main families (Narendra & Thathachar, 1989): fixed structure learning automata and variable structure learning automata. Variable structure

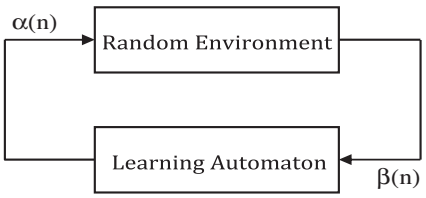

Fig. 1. The relationship between the learning automaton and its random environment.

learning automata are represented by a triple $\langle \underline{\beta}, \underline{\alpha}, T \rangle$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha_i(k) \in \underline{\alpha}$ and $\underline{p(k)}$ denote the action selected by learning automaton and the probability vector defined over the action set at instant k, respectively. Let a and b denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. Let r be the number of actions that can be taken by learning automaton. At each instant, the action probability vector $\underline{p(k)}$ is updated by the linear learning algorithm given in Eq. (2), if the selected action $\alpha_i(k)$ is rewarded by the random environment, and it is updated as given in Eq. (3) if the taken action is penalized.

$$p_j(k+1) = \begin{cases} p_j(k) + a[1 - p_j(k)] & j = i \\ (1-a)p_j(k) & \forall j \neq i \end{cases} \tag{2}$$

$$p_j(k+1) = \begin{cases} (1-b)p_j(k) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(k) & \forall j \neq i \end{cases} \tag{3}$$

If $a = b$, the recurrence Eqs. (2) and (3) are called linear reward-penalty (L_{R-P}) algorithm, if $a \gg b$ the given equations are called linear reward-ϵ penalty ($L_{R-\epsilon P}$), and finally if $b = 0$ they are called linear reward-Inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

3.3. Cellular learning automata

Cellular learning automata, which is a combination of cellular automata and learning automata, is a powerful mathematical model for many decentralized problems and phenomena. The basic idea of CLA, which is a subclass of stochastic CA, is to use learning automata to adjust the state transition probability of stochastic CA. Cellular learning automata is a mathematical model for dynamical complex systems that consists of a large number of simple components. These components, which have the learning capability, cooperate to produce complicated behavioral patterns. A CLA is a CA in which one or more learning automata are assigned to its every cell. The learning automaton residing in a particular cell determines its state (action) on the basis of its action probability vector. Like CA, there is a rule that CLA operate under it. The rule of CLA and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in that cell. In CLA, the neighboring learning automata of any particular learning automaton constitute its local environment. The action probability vector of the neighboring learning automata varies during the evolution of the CLA, and so the local environment is nonstationary. CLA has been found to perform well in many applications such as image processing (Meybodi, Beigy, & Taherkhani, 2000), rumor diffusion (Meybodi & Taherkhani, 2001), modeling of commerce networks (Meybodi & Khojasteh, 2001), channel assignment in cellular networks (Beigy & Meybodi, 2003), clustering the wireless sensor networks, VLSI Placement

(Meybodi & Mehdipour, 2003), and solving NP-hard problems (Enami Eraghi, Akbari Torkestani, & Meybodi, 2009a, 2009b), to name just a few.

The operation of cellular learning automata could be described as follows: At the first step, the internal state of every cell is specified. The state of every cell is determined on the basis of action probability vectors of the learning automata residing in that cell. The initial value of this state may be chosen on the basis of past experience or at random. In the second step, the rule of cellular automata determines the reinforcement signal to each learning automaton residing in that cell. Finally, each learning automaton updates its action probability vector on the basis of supplied reinforcement signal and the chosen action. This process continues until the desired result is obtained. Formally a d-dimensional CLA is given below.

CELLULAR LEARNING AUTOMATA A d-dimensional cellular learning automata is a structure $\mathcal{A} = (Z^d, \Phi, A, N, f)$, where.

1. Z^d is a lattice of d-tuples of integer numbers.
2. $\underline{\Phi}$ is a finite set of states.
3. A is the set of LAs each of which is assigned to each cell of the CA.
4. $N = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m\}$ is a finite subset of Z^d called neighborhood vector, where $\bar{x}_i \in Z^d$.
5. $f : \underline{\Phi}^m \rightarrow \underline{\beta}$ is the local rule of the cellular learning automata, where $\underline{\beta}$ is the set of values that the reinforcement signal can take. It gives the reward (reinforcement) signal to each LA from the current actions selected by its neighboring LAs.

Cellular learning automata can be classified into synchronous and asynchronous models. In synchronous CLA, all cells are synchronized with a global clock and executed at the same time. That is, in synchronous CLA, all learning automata (in different cells) are activated at the same time in parallel. In Meybodi and Beigy (2004), a mathematical methodology is introduced to study the behavior of the synchronous CLA and its convergence properties. It is shown that the synchronous CLA converges to a globally stable state for a class of rules called commutative rules. A CLA is called asynchronous CLA (ACLA) if at a given time only some LAs are activated independently from each other, rather than all together in parallel. In some applications such as dynamic channel assignment in cellular networks (Beigy & Meybodi, 2003), a type of cellular learning automata in which the action of each cell in next stage of its evolution not only depends on the local environment (actions of its neighbors) but it also depends on the external environments. Such a cellular learning automata is referred to as open synchronous CLA (Beigy & Meybodi, 2007).

3.4. Irregular cellular learning automata

An Irregular cellular learning automata (ICLA) (Enami Eraghi et al., 2009a, Enami Eraghi, Akbari Torkestani, & Meybodi, 2009b; Esnaashari & Meybodi, 2008) is a cellular learning automata (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 non-stationary because the action probability vectors of the neighboring LAs vary during the evolution of the ICLA.

4. The proposed vertex coloring algorithm

In this section, we propose a cellular learning automat-based approximation algorithm called CLAVCA for solving the minimum vertex coloring problem. In the proposed algorithm, before the coloring process starts, an asynchronous irregular cellular learning automata isomorphic to the input graph is created. To construct such an irregular cellular learning automata, each graph node is associated with a cell of cellular learning automata, and then a learning automaton is assigned to each cell. Learning automata are independently activated and choose their colors. The proposed coloring algorithm is a fully distributed algorithm which is locally and independently run at each cell of the cellular learning automata. In fact, each cell locally selects its color independent of the other cells. Each cell $\mathcal{C}_i$ of the cellular learning automata (or each vertex v_i) is equipped with a learning automaton A_i whose actions constitute the set of colors by which the cell $\mathcal{C}_i$ can be colored. The resulting irregular cellular learning automata can be modeled by a duple $\langle \underline{A}, \underline{a} \rangle$, where $\underline{A} = \{A_1, A_2, \ldots, A_n\}$ denotes the set of learning automata corresponding to the graph vertex-set, and $\underline{a} = \{\underline{a_1}, \underline{a_2}, \ldots, \underline{a_n}\}$ denotes the action-set in which $\underline{a_i} = \{\underline{a_i^1}, \underline{a_i^2}, \ldots, \underline{a_i^{r_i}}\}$ (for each $\underline{a_i} \in \underline{a}$) defines the set of actions that can be taken by learning automaton A_i (or the set of colors by which vertex v_i can be colored).

Note that, since each cell $\mathcal{C}_i$ is associated with vertex v_i, hereafter (in some cases) vertex v_i may be referred to as cell $\mathcal{C}_i$ and vice versa. It has been shown in reference (Galinier & Hertz, 2006) that an arbitrary graph can be colored with at most $\Delta + 1$ colors, where Δ denotes the maximum vertex degree in the graph (or the graph degree). Therefore, it can be concluded that cell $\mathcal{C}_i$ (vertex v_i) and its neighboring cells can be colored with at most $\Delta_i + 1$ colors in the worst case, where Δ_i is the degree of vertex v_i. That is why, in our proposed vertex coloring algorithm, the action-set of learning automaton (corresponding to color set of cell $\mathcal{C}_i$) is composed of $\Delta_i + 1$ actions (or colors). In the proposed vertex coloring algorithm, the color selected by each cell (or the action chosen by its corresponding learning automaton) is rewarded or penalized based solely on the colors chosen by its neighboring cells. That is, the proposed algorithm can be fully localized at each cell.

The operation of the proposed vertex coloring algorithm can be described as follows. The proposed algorithm is composed of a number of stages and at each stage the learning automaton of each cell randomly chooses one of its actions (colors) based on its action probability vector. For each learning automaton A_i, all the actions have the same initial choice probability each of $\frac{1}{\Delta_i+1}$. We call the set of colors which are selected by the learning automaton of cell $\mathcal{C}_i$ and its neighboring cells as the local color-set of cell $\mathcal{C}_i$. The cardinality of the local color-set of a given cell is referred to as the color-degree of the cell and denoted as $\mathcal{D}_i$. The local rule of the cellular learning automata under which the automata update their state is defined as follows. At each stage k of the proposed algorithm, the color which is selected by the learning automaton of cell $\mathcal{C}_i$ is penalized, if this color is also chosen by the automaton of at least one of its neighboring cells or the color-degree of the cell in this stage, i.e., $\mathcal{D}_i$, is greater than its own dynamic threshold, i.e., $\mathcal{J}_i$. Dynamic threshold $\mathcal{J}_i$ retains the minimum color-degree (which belongs to the best local coloring) that has been seen so far (see Line 16). Dynamic threshold $\mathcal{J}_i$ can be initially set to a value which is greater than or equal to the maximum $\mathcal{D}_i$. For example, in our algorithm $\mathcal{J}_i$ is set to $\Delta_i + 1$ (see Line 6). Otherwise, the selected color is

rewarded. In other words, the selected color is rewarded, if it is selected by none of the neighboring cells and $\mathcal{D}_i \leqslant \mathcal{J}_i$.

The proposed algorithm can be locally executed at each cell independent of the other cells. As the proposed algorithm proceeds, each learning automaton learns how to select a color based solely on the colors selected by its adjacent automata (two learning automata are adjacent, if they belongs to the neighboring cells) so that the mentioned color is selected by none of its adjacent learning automata. Since the local color-set which is formed by each cell and its neighboring cells must be compared with the best local color-set which it has seen so far, the proposed algorithm guarantees the optimality of the final color-set for each cell. This means that, as the proposed algorithm approaches to the end the size of the selected color-set converges to the optimal size. For each learning automaton A_i, the learning process stops if the choice probability of a color exceeds a pre-specified threshold, e.g., π_i (see Line 23). π_i's can be selected equal or different (discriminatory). Fig. 2 shows the proposed vertex coloring algorithm which is run at cell $\mathcal{C}_i$.

5. Complexity analysis

In this section, we analyze the time complexity of the proposed cellular learning automata-based vertex coloring algorithm. To compute the running time of algorithm, we first estimate an upper bound (lemma 1) and a lower bound (lemma 2) on the number of iterations of the proposed algorithm for finding a $\frac{1}{1-\epsilon}$ optimal local coloring in the neighborhood of each vertex. Then, we show that the time required for finding a $\frac{1}{1-\epsilon}$ optimal coloring (for each vertex) is confined between these two estimated bounds, and the running time of the proposed algorithm is restricted to the required number of iterations for coloring the vertex with the maximum degree.

Theorem 1. *Let $|opt_i|$ denotes the cardinality of the optimal local color-set of vertex v_i (say $\mathbb{C}_i^*$), and the action probability vector of vertex v_i (i.e., $\underline{p}_i$) is updated according to the proposed vertex coloring algorithm. The time required for finding a $\frac{1}{1-\epsilon}|opt_i|$ size local color-set is*

$$\mathcal{F}]\left(\frac{1}{\Delta_i+1}\right) \leqslant T_i(k) \leqslant \mathcal{F}\left(\frac{1}{\Delta_i+1}(1-a)^{\mathcal{M}_i-1}\right) \tag{4}$$

where

$$\mathcal{F}(x) = \frac{2}{1+x-\frac{\epsilon}{\Delta_i}}\log_{1-a}^{\frac{\epsilon}{\Delta_i(1-x)}}, \tag{5}$$

$\epsilon \in (0,1)$ is the error parameter of the proposed algorithm, a denotes the learning rate of algorithm, $\mathcal{M}_i$ is the number of local colorings which legally color vertex v_i and its neighbors, and Δ_i is the degree of vertex v_i.

Proof. Let $\underline{\mathbb{C}}_i^* = \{\mathbb{C}_i^1, \mathbb{C}_i^2, \ldots, \mathbb{C}_i^{\mathcal{M}_i}\}$ denotes the set of all possible local color-sets (colorings) by which vertex v_i and its neighbors can be legally colored, where $\mathcal{M}_i$ is the number of local colorings which legally color vertex v_i and its neighbors. Let $\mathbb{C}_i^* = \{C_j^* | (v_i, v_j) \in E \text{ or } i = j\}$ denotes the optimal local color-set in the neighborhood of vertex v_i, where C_j^* is the optimal color with which vertex v_j can be colored. Before stating the proof of this theorem, we prove the following two lemmas. □

Lemma 1. *If $\underline{p}_i$ is updated according to the proposed vertex coloring algorithm, the time required for finding a $\frac{1}{1-\epsilon}|opt_i|$ size local color-set in the worst case is*

$$\frac{2}{1+x-\frac{\epsilon}{\Delta_i}}\log_{1-a}^{\frac{\epsilon}{\Delta_i(1-x)}}$$

where $x \geqslant p_i^* \cdot (1-a)^{\mathcal{M}_i-1}$.

Proof. Lemma 1 aims at computing the worst case running time of the proposed algorithm. The worst case occurs if all the other color-sets (color-sets with greater color-degree in the neighborhood of vertex v_i) are chosen before the optimal one (i.e., $\mathbb{C}_i^*$). In such a case, the learning process can be divided into two distinct phases. In the first phase, called *shrinking phase*, it is assumed that all the other local color-sets, from the largest one to the smallest one, are chosen before $\mathbb{C}_i^*$ and so rewarded. Therefore, in the worst case, the probability of coloring vertex v_i with the optimal color (i.e., color c_i^*) at the end of the shrinking phase is computed as

$$p_i^*(\mathcal{M}_i - 1) \geqslant p_i^*(\mathcal{M}_i - 2) \cdot (1-a) \tag{6}$$

Cellular Learning Automata-based Vertex Coloring Algorithm (CLAVCA)

```
01: Input: Cell C_i, Threshold π_i
02: Begin Algorithm
03: Vertex v_i is associated with cell C_i
04: Cell C_i is equipped with automaton A_i
05: Automaton A_i forms its action-set
06: J_i ← Δ_i + 1
07: Repeat
08:   Automaton A_i chooses one of its colors at random
09:   If the color selected by automaton A_i is chosen by (at least one of) its adjacent automata Then
10:       Automaton A_i penalizes its selected color
11:   Else
12:       Automaton A_i computes its color-degree D_i
13:       If  D_i ≤ J_i  Then
14:           Automaton A_i rewards its selected color
15:           If automaton A_i and its adjacent automata are rewarded Then
16:               J_i ← D_i
17:           End If
18:       Else
19:           Automaton A_i penalizes its selected color
20:       End If
21:   End If
22:   Increment stage number k                    {k ← k + 1}
23: Until the probability with which automaton A_i chooses a color is greater than π_i
24: End Algorithm
```

Fig. 2. Pseudo code of the proposed vertex coloring algorithm.

where $\mathcal{M}_i$ is the number of possible local colorings in the neighborhood of vertex v_i, $a$denotes the learning rate of CLAVCA, and $p_i^*(\mathcal{M}_i - 1)$ denotes the probability with which vertex v_i is colored with optimal color c_i^* at the end of the shrinking phase. By repeatedly substituting recurrence function $p_i^*(\cdot)$ on the right hand side of inequality (6), we obtain

$$p_i^*(\mathcal{M}_i - 1) \geqslant p_i^* \cdot (1-a)^{\mathcal{M}_i - 1}$$

where p_i^* denotes the initial probability of coloring vertex v_i with its optimal color (i.e., color c_i^*). For the sake of simplicity in notation, $p_i^*(\mathcal{M}_i - 1)$ is temporarily substituted by q_i^*.

The second phase called *growing phase* begins when the optimal color-set, $\mathbb{C}_i^*$, is chosen for the first time. According to the proposed algorithm, during the growing phase, the probability of penalizing the optimal color-set is zero (at vertex v_i). Furthermore, since the reinforcement scheme by which the proposed algorithm updates the probability vectors is L_{R-1}, the conditional expectation of $q_i^*(k)$ (i.e., the probability of choosing color c_i^* at stage k of the growing phase), remains unchanged when the other color-sets are selected, and increases only when color-set $\mathbb{C}_i^*$ is chosen. That is, during the growing phase, the changes in the conditional expectation of $q_i^*(k)$ is always non-negative and calculated as

$$\begin{aligned}
&q_i^*(1) = q_i^* + a \cdot (1 - q_i^*)\\
&q_i^*(2) = q_i^*(1) + a \cdot (1 - q_i^*(1)) = q_i^*(1) \cdot (1-a) + a\\
&\vdots\\
&q_i^*(k-1) = q_i^*(k-2) + a \cdot (1 - q_i^*(k-2)) = q_i^*(k-2) \cdot (1-a) + a\\
&q_i^*(k) = q_i^*(k-1) + a \cdot (1 - q_i^*(k-1)) = q_i^*(k-1) \cdot (1-a) + a
\end{aligned} \tag{7}$$

where k denotes the number of times color c_i^* must be chosen until the following condition is met.

$$q_i^*(k) = 1 - \frac{\epsilon}{\Delta_i} \tag{8}$$

where $1 - \frac{\epsilon}{\Delta_i} = \pi_i$. It should be noted that the growing phase continues until the probability of choosing color c_i^* approaches $1 - \frac{\epsilon}{\Delta_i}$. By substituting recurrence function $q_i^*(k)$ and after some simplifications we have

$$\begin{aligned}
q_i^*(k) &= q_i^*(k-1) \cdot (1-a) + a\\
&= [q_i^*(k-2) \cdot (1-a) + a] \cdot (1-a) + a = q_i^*(k-2) \cdot (1-a)^2 + a \cdot (1-a) + a\\
&= [q_i^*(k-3) \cdot (1-a) + a] \cdot (1-a)^2 + a \cdot (1-a) + a = q_i^*(k-2) \cdot (1-a)^3\\
&\quad + a \cdot (1-a)^2 + a \cdot (1-a) + a\\
&\vdots\\
&= q_i^*(1) \cdot (1-a)^{k-1} + a \cdot (1-a)^{k-2} + \cdots + 1 \cdot (1-a) + a\\
&= q_i^* \cdot (1-a)^k + a \cdot (1-a)^{k-1} + \cdots + a.(1-a) + a
\end{aligned}$$

Hence, we have

$$q_i^*(k) = q_i^* \cdot (1-a)^k + a \cdot (1-a)^{k-1} + \cdots + a \cdot (1-a) + a \tag{9}$$

After some algebraic simplifications, we have

$$q_i^*(k) = q_i^* \cdot (1-a)^k + a \cdot (1 + (1-a) + (1-a)^2 + \cdots + (1-a)^{k-1})$$

and so

$$q_i^*(k) = q_i^* \cdot (1-a)^k + a \cdot \sum_{i=0}^{k-1} (1-a)^i \tag{10}$$

The second term on the right hand side of Eq. (10) is a geometric series that sums up to $a \cdot \left(\frac{1-(1-a)^k}{1-(1-a)}\right)$, where $|1-a| < 1$. Since the learning rate $a \in (0,1)$, we have

$$q_i^*(k) = q_i^* \cdot (1-a)^k + a \cdot \left(\frac{1-(1-a)^k}{1-(1-a)}\right) \tag{11}$$

and

$$q_i^*(k) = q_i^* \cdot (1-a)^k + 1 - (1-a)^k \tag{12}$$

From Eqs. (8) and (12) we have

$$q_i^*(1-a)^k + 1 - (1-a)^k = 1 - \frac{\epsilon}{\Delta_i} \tag{13}$$

and

$$(1-a)^k = \frac{\epsilon}{\Delta_i(1-q_i^*)} \tag{14}$$

Taking $\log_{1-a}$ of both sides of Eq. (14), we derive

$$k = \log_{1-a}^{\frac{\epsilon}{\Delta_i(1-q_i^*)}} \tag{15}$$

Since during the growing phase, q_i^* remains unchanged when the other colors are penalized, k does not reflect the number of times the other colors are chosen and this should be separately calculated based on k. For this purpose, let q_i^* be the probability of choosing optimal color c_i^* at the beginning of the growing phase, and reaches $1 - \frac{\epsilon}{\Delta_i}$ after k iterations. On the other hand, the probability of choosing all the other colors is initially $1 - q_i^*$, and reaches $\frac{\epsilon}{\Delta_i}$ after the same number of iterations. Thus, the number of times the other colors are chosen (before the condition given in Eq. (8) is met) is obtained as

$$\frac{1 - q_i^* + \frac{\epsilon}{\Delta_i}}{1 + q_i^* - \frac{\epsilon}{\Delta_i}} \cdot k \tag{16}$$

Let $\mathbb{K}$ denotes the total number of iterations required to satisfy the condition given in Eq. (8). From Eq. (16) we have

$$\mathbb{K} = \frac{2}{1 + q_i^* - \frac{\epsilon}{\Delta_i}} \cdot k$$

By substituting from Eq. (15) we have

$$\mathbb{K} = \frac{2}{1 + q_i^* - \frac{\epsilon}{\Delta_i}} \cdot \log_{1-a}^{\frac{\epsilon}{\Delta_i(1-q_i^*)}} \tag{17}$$

From inequality (7) and Eq. (17), we conclude that the time complexity of CLAVCA for finding a $\frac{1}{1-\epsilon}|opt_i|$ size local color-set is less than

$$\frac{2}{1 + q_i^* - \frac{\epsilon}{\Delta_i}} \cdot \log_{1-a}^{\frac{\epsilon}{\Delta_i(1-q_i^*)}} \tag{18}$$

where $q_i^* \geqslant p_i^* \cdot (1-a)^{\mathcal{M}_i - 1}$, and hence the proof of Lemma 1. □

Lemma 2. *If p_i is updated according to the proposed vertex coloring algorithm, the running time of the proposed algorithm for finding a $\frac{1}{1-\epsilon}|opt_i|$ size local color-set is greater than*

$$\frac{2}{1 + q_i^* - \frac{\epsilon}{\Delta_i}} \cdot \log_{1-a}^{\frac{\epsilon}{\Delta_i(1-p_i^*)}}$$

Proof. Lemma 2 considers the running time of the proposed algorithm in the best case. In CLAVCA, the best case occurs when the optimal coloring $\mathbb{C}_i^*$ is chosen as the first coloring in the neighborhood of vertex v_i. In this case, the learning process does not include the shrinking phase, and so at the beginning of the growing phase the probability of coloring vertex v_i with optimal color c_i^* is equal to the initial probability p_i^*. Therefore, similar to the proof of Lemma 1, it can be easily proved that the minimum number of colorings required for satisfying the condition given in Eq. (8) is

$$\frac{2}{1+q_i^*-\frac{\epsilon}{\Delta_i}}\cdot\log_{1-a}^{\frac{\epsilon}{\Delta_i(1-q_i^*)}} \tag{19}$$

where $q_i^*=p_i^*$, which completes the proof of Lemma 2. □

From inequalities (18) and (19), we conclude that

$$\frac{2}{1+q_i^*-\frac{\epsilon}{\Delta_i}}\cdot\log_{1-a}^{\frac{\epsilon}{\Delta_i(1-q_i^*)}}\leqslant T_i(k)\leqslant\frac{2}{1+q_i^*-\frac{\epsilon}{\Delta_i}}\cdot\log_{1-a}^{\frac{\epsilon}{\Delta_i(1-q_i^*)}}$$

where $q_i^*\geqslant p_i^*\cdot(1-a)^{\mathcal{M}_i-1}$. As described in Section 3, the action-set of learning automaton A_i (corresponding to vertex v_i) comprises Δ_i+1 actions (or colors) each of initial probability $\frac{1}{\Delta_i+1}$. Therefore, the initial probability $p_i^*=\frac{1}{\Delta_i+1}$, and so we have

$$\mathcal{F}\left(\frac{1}{\Delta_i+1}\right)\leqslant T_i(k)\leqslant\mathcal{F}\left(\frac{1}{\Delta_i+1}(1-a)^{\mathcal{M}_i-1}\right)$$

where

$$\mathcal{F}(x)=\frac{2}{1+x-\frac{\epsilon}{\Delta_i}}\log_{1-a}^{\frac{\epsilon}{\Delta_i(1-x)}},$$

which completes the proof of the theorem. □

Theorem 2. *Time complexity of the proposed vertex coloring algorithm for finding a $\frac{1}{1-\epsilon}$ optimal coloring for graph $G\langle V,E\rangle$ is*

$$\mathcal{F}\left(\frac{1}{\Delta+1}\right)\leqslant T(k)\leqslant\mathcal{F}\left(\frac{1}{\Delta+1}(1-a)^{\mathcal{M}-1}\right)$$

where Δ is the graph degree (i.e., maximum vertex degree), $\mathcal{M}$ is the number of local colorings of the vertex with maximum degree Δ, and

$$\mathcal{F}(x)=\frac{2}{1+x-\frac{\epsilon}{\Delta}}\log_{1-a}^{\frac{\epsilon}{\Delta(1-x)}}$$

Proof. As stated earlier in the proposed algorithm, each learning automaton is randomly activated and colors its corresponding vertex independent of the other vertices. Therefore, the maximum number of iterations of the proposed algorithm is taken for finding a $\frac{1}{1-\epsilon}$ optimal local coloring for vertex v_j, where $\Delta_j=\max_{\forall v_i\in V}\Delta_i$ (maximum degree Δ_j is referred to as graph degree and denoted by Δ). That is, the maximum running time belongs to the vertex with the maximum degree Δ. On the other hand, as calculated in Lemmas 1 and 2, the running time of the proposed algorithm for finding a $\frac{1}{1-\epsilon}$ optimal coloring of graph $G\langle V,E\rangle$ is limited by the upper bound and lower bound on the running time of algorithm for the vertex with degree. Therefore, it is concluded that the time taken by the proposed algorithm for f finding a optimal coloring of graph is $G\langle V,E\rangle$

$$\mathcal{F}\left(\frac{1}{\Delta+1}\right)\leqslant T(k)\leqslant\mathcal{F}\left(\frac{1}{\Delta+1}(1-a)^{\mathcal{M}-1}\right)$$

where $\mathcal{F}(x)=\frac{2}{1+x-\frac{\epsilon}{\Delta}}\log_{1-a}^{\frac{\epsilon}{\Delta(1-x)}}$, and the proof of Theorem 2 is completed. □

6. Numerical results

To show the efficiency of the proposed vertex coloring algorithm, we have conducted several computer simulations. In these experiments, the proposed vertex coloring algorithm is tested on a subset of hard-to-color benchmarks like Leighton (Leighton, 1979), DSJ (Johnson, Aragon, McGeoch, & Schevon, 1991) and Wap (Caramia & Dell'Olmo, 2008). The performance of the

Table 1
The characteristics of DSJ benchmark graphs.

Graph name	Class	Number of vertices	Number of edges	Density
DSJC125.1	DSJ	125	736	0.09
DSJC125.5	DSJ	125	3891	0.50
DSJC125.9	DSJ	125	6961	0.90
DSJC250.1	DSJ	250	3218	0.10
DSJC250.5	DSJ	250	15668	0.50
DSJC250.9	DSJ	250	27897	0.90
DSJC500.1	DSJ	500	12458	0.10
DSJC500.5	DSJ	500	62624	0.50
DSJC500.9	DSJ	500	112437	0.90
DSJR500.1	DSJ	500	3555	0.03
DSJR500.1c	DSJ	500	121275	0.97
DSJR500.5	DSJ	500	58862	0.47
DSJC1000.1	DSJ	1000	49629	0.10
DSJC1000.5	DSJ	1000	249826	0.50
DSJC1000.9	DSJ	1000	449449	0.90

Table 3
The characteristics of Leighton benchmark graphs.

Graph name	Class	Number of vertices	Number of edges	Density
Le450_5a	LEI	450	5714	0.06
Le450_5b	LEI	450	5734	0.06
Le450_5c	LEI	450	9803	0.10
Le450_5d	LEI	450	9757	0.10
Le450_15a	LEI	450	8168	0.08
Le450_15b	LEI	450	8169	0.08
Le450_15c	LEI	450	16680	0.17
Le450_15d	LEI	450	16750	0.17
Le450_25a	LEI	450	8260	0.08
Le450_25b	LEI	450	8263	0.08
Le450_25c	LEI	450	17343	0.17
Le450_25d	LEI	450	17425	0.17

Table 2
A performance comparison of the coloring algorithms on hard-to-color DSJ benchmark graphs.

Graph	Best	TPA		AMACOL		ILS		CHECKCOL		GLS		CLAVCA	
		CN	RT	CN	RT	CN	RT	CN	RT	CN	RT	CN	RT
DSJC125.1	5	5	0	5	0	5	0	5	0	5	0	5	0.0090
DSJC125.5	17	19	289	17	125	17	2	17	110	18	0	17	11.898
DSJC125.9	44	44	5	44	57	44	0	44	4	44	0	44	19.463
DSJC250.1	8	8	10	8	12	8	0	8	28	9	0	8	07.234
DSJC250.5	28	30	3282	28	64	28	34	28	557	30	1	28	27.124
DSJC250.9	72	72	155	72	2604	72	6	72	182	73	6	73	31.684
DSJC500.1	12	12	0	12	9	13	0	12	4	13	0	12	20.390
DSJC500.5	48	48	124	48	326	50	106	48	1789	52	81	48	42.298
DSJC500.9	126	127	1268	126	1710	128	82	126	2045	130	154	126	70.465
DSJC1000.1	20	21	28	20	969	21	6	21	142	22	1	21	38.670
DSJC1000.5	84	84	2386	84	9235	91	303	84	7025	93	546	84	87.731
DSJC1000.9	224	226	3422	224	4937	228	2245	226	12545	234	1621	224	119.10

Table 4
A performance comparison of the coloring algorithms on hard-to-color Leighton benchmark graphs.

Graph	Best	TPA		AMACOL		ILS		CHECKCOL		GLS		CLAVCA	
		CN	RT	CN	RT	CN	RT	CN	RT	CN	RT	CN	RT
Le450_15a	15	15	1444	15	345	15	0	15	2145	15	2	15	22.785
Le450_15b	15	15	1655	15	345	15	0	15	2756	15	0	15	17.560
Le450_15c	15	15	82	15	2	15	19	15	4534	15	6	15	29.780
Le450_15d	15	15	34	15	4	15	20	15	4576	15	8	15	31.567
Le450_25c	25	26	44	26	93	26	2	25	3477	26	18	25	30.187
Le450_25d	25	26	22	26	10	26	1	25	4524	26	2	25	45.676

Table 5
The characteristics of Wap benchmark graphs.

Graph name	Class	Number of vertices	Number of edges	Density
Wap01a	KOS	2368	110871	0.04
Wap02a	KOS	2464	111742	0.04
Wap03a	KOS	4730	286722	0.03
Wap04a	KOS	5231	294902	0.02
Wap05a	KOS	905	43081	0.11
Wap06a	KOS	947	43571	0.10
Wap07a	KOS	1809	103368	0.06
Wap08a	KOS	1870	104176	0.06

proposed algorithm is measured both in terms of the time and the number of colors required for coloring the benchmarks, and compared with those of CHECKCOL (Caramia et al., 2006), GLS (Voudouris & Tsang, 2003), ILS (Lourenco et al., 2002), TPA (Caramia & Dell'Olmo, 2008) and AMACOL (Galinier et al., 2008). In these experiments, learning rate of the proposed algorithm is set to 0.1, and algorithm is terminated (for all cells) when the probability of the chosen color-set is 0.95 or greater. The obtained results are summarized in Tables 2, 4, and 6. In these tables, the first column includes the number of colors required for coloring the graph (CN), and the second column includes the running time of each algorithm in seconds (RT).

DSJ (Johnson et al., 1991) benchmark graphs are the first class on which Algorithm 4 is tested. This class comprises a set of uniform (n,p) random graphs which are denoted as *DSJCn.p*, where $n = \{125, 250, 500, 1000\}$ is the number of vertices, and $p = \{0.1, 0.5, 0.9\}$ denotes the probability of connecting every pair of nodes in the graph. graphs are geometric graphs were introduced by Johnson et al. (1991) to provide benchmarks for heuristics. The number of nodes in these graphs changes from 125 to 1000, and so, it is not easy to find good solutions, especially with a density of 0.5 and 0.9, and with a number of nodes greater than 125. The characteristics (i.e., density and number of vertices and edges) of DSJ benchmark graphs are given in Table 1.

Table 2 shows the results of the conducted experiments on DSJ benchmark graphs. Comparing the results of the other algorithms, we observe that, in most cases, GLS has the shortest running time, and CHECKCOL and AMACOL have the worst running time. It also can be seen that, in almost all cases, AMACOL picks the smallest number of colors to color the graphs, and GLS uses the most number of colors. Comparing the results of the CLAVCA with AMACOL, we find that the color-sets chosen by the proposed algorithm are as small as those of AMACOL, while the running time of the proposed algorithm is considerably shorter than AMACOL. On the other hand, comparing the proposed algorithm with GLS, it can be seen that the running time of the proposed algorithm is as close to GLS as possible, while the size of the color-set, in the proposed algorithm, is significantly smaller as compared with GLS.

The second class of the benchmark graphs on which Algorithm 4 is tested is Leighton (Akbari Torkestani & Meybodi, in press-a) which is a set of large random graphs, with 450 nodes, and denoted as *Le*450_*xy*, where $x = \{5,15,25\}$ is the chromatic number of the graph and equal to the maximum clique size, and extension $y = \{a,b,c,d\}$ denotes the edge density of the graph. Finding a solution to the graphs with extensions *c* and *d* is, in general, more difficult than that with extensions *c* and *b*. The characteristics (i.e., density and number of vertices and edges) of Leighton benchmark graphs are given in Table 3.

Table 4 shows the results of the simulation experiments conducted on Leighton benchmark graphs. Comparing the results reported in Table 4, we find that, in almost all cases, ILS outperforms the others in terms of running time. It can be seen that CHECKCOL always selects the smallest color-sets for coloring the graphs, while its running time is the worst. Comparing the results of the proposed algorithm with the chromatic number of the benchmarks, we observe that the size of the color-sets constructed by the proposed algorithm is equal to the chromatic number. It can be seen that, CHECKCOL is also capable of finding the optimal solution (Best results). However, comparing the running time of our proposed algorithm with that of CHECKCOL, we find that the proposed algorithm significantly outperforms CHECKCOL in terms of the running time.

The third class of the benchmarks we consider includes Wap graphs which arise in the design of transparent optical networks and are denoted by *Wap*0*ma*, where $m = \{1,2,\ldots,8\}$. In this class, the graphs have a large number of 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 5.

Table 6
A performance comparison of the coloring algorithms on hard-to-color Wap benchmark graphs.

Graph	Best	TPA		AMACOL		ILS		CHECKCOL		GLS		CLAVCA	
		CN	RT	CN	RT	CN	RT	CN	RT	CN	RT	CN	RT
Wap01a	42	42	245	45	345	44	2	44	568	42	55	42	15.785
Wap02a	41	41	1618	44	802	43	251	43	486	41	160	41	27.902
Wap03a	44	44	17	53	245	46	365	46	689	44	782	43	50.563
Wap04a	43	43	95	48	45	44	484	44	23	43	834	43	46.576
Wap06a	41	41	348	44	545	42	1	42	25	41	8	40	2.6150
Wap07a	42	42	541	45	89	44	1	44	182	42	215	40	14.398
Wap08a	42	42	200	45	446	43	56	44	22	42	41	42	26.471

Table 6 shows the results of the simulation experiments conducted on Wap benchmark graphs. Comparing the results reported in Table 6, we observe that, GLS and TPA outperform the others in terms of the required number of colors, and ILS in terms of the time required for coloring the Wap benchmark graphs. The obtained results given in Table 6 show that not only the size of the color-sets constructed by our proposed algorithm is smaller than that of the best reported results, but also the running time of the proposed algorithm is considerably shorter as compared with the other coloring methods.

7. Conclusion

In this paper, an irregular cellular learning automata-based algorithm was proposed for finding a near optimal solution to the vertex coloring problem. Irregular cellular learning automata is a generalization of cellular learning automata in which the restriction of rectangular grid structure in traditional CLA is removed. The proposed coloring algorithm is a fully distributed algorithm in which each vertex chooses its optimal color based solely on the colors selected by its adjacent vertices. We computed the time complexity of the proposed algorithm for finding a near optimal solution of the vertex coloring problem in an arbitrary graph. Several computer simulations were conducted on hard-to-color benchmark graphs to show the efficiency of the proposed algorithm. The obtained results showed that our proposed algorithm outperforms the existing methods both in terms of the running time of algorithm and the required number of colors.

References

Akbari Torkestani, J., & Meybodi, M. R. (in press-a). A learning automata-based cognitive radio for clustered wireless ad-hoc networks. *Journal of Network and Systems Management*.

Akbari Torkestani, J., & Meybodi, M. R. (in press-b). A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs. *Journal of Supercomputing*.

Akbari Torkestani, J., & Meybodi, M. R. (in press-c). Weighted steiner connected dominating set and its application to multicast routing in wireless MANETs. *Wireless Personal Communications*.

Akbari Torkestani, J., & Meybodi, M. R. (in press-d). A mobility-based cluster formation algorithm for wireless mobile ad Hoc networks. *Journal of Cluster Computing*.

Akbari Torkestani, J., & Meybodi, M. R. (2010a). Learning automata-based algorithms for finding minimum weakly connected dominating set in stochastic graphs. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, 18*(6), 721–758.

Akbari Torkestani, J., & Meybodi, M. R. (2010b). Mobility-based multicast routing algorithm in wireless mobile ad hoc networks: A learning automata approach. *Journal of Computer Communications, 33*, 721–735.

Akbari Torkestani, J., & Meybodi, M. R. (2010c). A new vertex coloring algorithm based on variable action-set learning automata. *Journal of Computing and Informatics, 29*(3), 1001–1020.

Akbari Torkestani, J., & Meybodi, M. R. (2010d). An efficient cluster-based CDMA/TDMA scheme for wireless mobile ad-hoc networks: A learning automata approach. *Journal of Network and Computer applications, 33*, 477–490.

Akbari Torkestani, J., & Meybodi, M. R. (2010e). Clustering the wireless ad-hoc networks: A distributed learning automata approach. *Journal of Parallel and Distributed Computing, 70*, 394–405.

Akbari Torkestani, J., & Meybodi, M. R. (2010f). An intelligent backbone formation algorithm in wireless ad hoc networks based on distributed learning automata. *Journal of Computer Networks, 54*, 826–843.

Asmuni, H., Burke, E. K., Garibaldi, J. M., McCollum, B., & Parkes, A. J. (2009). An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables. *Computers & Operations Research, 36*, 981–1001.

Barnier, N., & Brisset, P., (2002). Graph coloring for air traffic flow management. In *Proceedings of the fourth international workshop on integration of AI and OR techniques* (pp. 133–147). France: Le Croisic.

Beigel, R., & Eppstein, D. (2005). 3-coloring in time. *Journal of Algorithms, 54*, 168–204.

Beigy, H., & Meybodi, M. R. (2003). A self-organizing channel assignment algorithm: A cellular learning automata approach. *Springer-verlag lecture notes in computer science* (Vol. 2690, pp. 119–126). Springer-Verlag.

Beigy, H., & Meybodi, M. R. (2007). Open synchronous cellular learning automata. *Advances in Complex Systems, 10*(4), 527–556.

Blchliger, I., & Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research, 35*, 960–975.

Brelaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM, 22*(4), 252–256.

Bui, T. N., Nguyen, T. H., Patel, C. M., & Phan, K. T. (2008). An ant-based algorithm for coloring graph. *Discrete Applied Mathematics, 156*, 190–200.

Byskov, J. M. (2004). Enumerating maximal independent sets with applications to graph coloring. *Operations Research Letters, 32*, 547–556.

Byskov, J. M., (2005). Exact algorithms for graph coloring and exact satisfiability, PhD thesis, University of Aarhus, Aarhus, Denmark.

Caramia, M., & Dell'Olmo, P. (1999). A fast and simple local search for graph coloring. *Lecture Notes in Computer Science, Algorithm Engineering, 1618*, 319–333.

Caramia, M., & Dell'Olmo, P. (2008). Embedding a novel objective function in a two-phased local search for robust vertex coloring. *European Journal of Operational Research, 189*, 1358–1380.

Caramia, M., & Dellolmo, P. (2008). Coloring graphs by iterated local search traversing feasible and infeasible solutions. *Discrete Applied Mathematics, 156*, 201–217.

Caramia, M., Dellolmo, P., & Italiano, G. F. (2006). CHECKCOL: Improved local search for graph coloring. *Journal of Discrete Algorithms, 4*, 277–298.

Carter, M., Laporte, G., & Lee, S. (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society, 47*, 373–383.

Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E., & Markstein, P. W. (1981). Register allocation via coloring. *Computer Languages, 6*, 47–57.

Chams, M., Hertz, A., & de Werra, D. (1987). Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research, 32*, 260–266.

Chiarandin, M., & Stutzle, T. (2007). Stochastic local search algorithms for graph set T-coloring and frequency assignment. *Constraints, 12*(3), 371–403.

Cooper, C., Dyer, M., & Frieze, A. (2001). On Markov chains for randomly H-coloring a graph. *Journal of Algorithms, 39*, 117–134.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research, 19*, 151–162.

Dobrev, S., Schröder, H., Sykora, O., & Vrto, I. (2000). Evolutionary graph coloring. *Information Processing Letters, 76*, 91–94.

Dorne, R., & Hao, J. K., (1998). A new genetic local search algorithm for graph coloring. Lecture Notes in Computer Science (Vol. 1498). Germany, pp. 745–754.

Dowsland, K. A., & Thompson, J. M. (2008). An improved ant colony optimization heuristic for graph coloring. *Discrete Applied Mathematics, 156*, 313–324.

Eiben, A. E., Vanderhauw, J. K., & Vanhemert, J. I. (1998). Graph coloring with adaptive evolutionary algorithms. *Journal of Heuristics, 4*, 25–46.

Enami Eraghi, A., Akbari Torkestani, J., & Meybodi, M. R., (2009). Solving the bandwidth multicoloring problem: A cellular learning automata approach. In *International IACSIT conference on machine learning and computing (IACSIT ICMLC 2009)*. Perth, Australia.

Enami Eraghi, A., Akbari Torkestani, J., & Meybodi, M. R., (2009). Cellular learning automata-based graph coloring problem. In *International IACSIT conference on machine learning and computing (IACSIT ICMLC 2009)*. Perth, Australia.

Esnaashari, M., & Meybodi, M. R. (2008). A cellular learning automata based clustering algorithm for wireless sensor networks. *Sensor Letters, 6*(5), 723–735.

Fleurent, C., & Ferland, J. A. (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research, 63*, 437–461.

Fredkin, E. (1990). Digital machine: A informational process based on reversible cellular automata. *Physica, D45*, 245–270.

Galindier, P., & HAO, J. K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization, 3*, 379–397.

Galinier, P., & Hao, J. K., (1997). Tabu search for maximal constraint satisfaction problems. In *Proceedings of the third international conference on principles and practice of constraint programming (CP'97)* (pp. 196–208). Lecture Notes in Computer Science, Vol. 1330. Germany.

Galinier, P., & Hertz, A. (2006). A survey of local search methods for graph coloring. *Computers & Operations Research, 33*, 2547–2562.

Galinier, P., Hertz, A., & Zufferey, N. (2008). An adaptive memory algorithm for the K-coloring problem. *Discrete Applied Mathematics, 156*, 267–279.

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

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: W.H. Freeman and Co..

Glass, C., & Prügel-Bennett, A. (2003). Genetic algorithm for graph coloring: Exploration of Galinier and Hao's Algorithm. *Journal of Combinatorial Optimization, 7*, 229–236.

Hertz, A., & Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing, 39*, 345–351.

Johnson, D. R., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation, part II, graph coloring and number partitioning. *Operations Research, 39*, 378–406.

Kari, J. (1990). Reversibility of 2D cellular automata is undecidable. *Physica, D45*, 379–385.

Karp, R. M. (1972). *Reducibility among combinatorial problems complexity of computer computations*. USA: Plenum Press. pp. 85–103.

Lawler, E. (1976). A note on the complexity of the chromatic number problem. *Information Processing Letters, 5*, 66–67.

Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards, 84*(6), 489–506.

Lewis, R., & Paechter, B. (2007). Finding feasible timetables using group based operators. *IEEE Transactions on Evolutionary Computation, 11*(3), 397–413.

Lourenco, H. R., Martin, O., & Stutzle, T. (2002). Iterated local search. In F. Glover & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 321–353). USA: Kluwer Academic Publishers.

Mabrouk, B. B., Hasni, H., & Mahjoub, Z. (2008). On a parallel genetic–tabu search based algorithm for solving the graph coloring problem. *European Journal of Operational Research*. doi:10.1016/j.ejor.2008.03.050.

Malaguti, E., & Toth, P. (2008). An evolutionary approach for bandwidth multi coloring problems. *European Journal of Operational Research, 189*, 638–651.

Meybodi, M. R., & Khojasteh, M. R., (2001). Application of cellular learning automata in modelling of commerce networks. In *Proceedings of 6th annual international computer society of iran computer conference CSICC-2001* (pp. 284–295). Isfehan, Iran.

Meybodi, M. R., & Mehdipour, F., (2003). VLSI placement using cellular learning automata. In *Proceedings of 8th annual international computer society of iran computer conference CSICC-2001* (pp. 195–203). Mashad, Iran.

Meybodi, M. R., & Taherkhani, M., (2001). Application of cellular learning automata in modeling of rumor diffusion. In *Proceedings of 9th conference on electrical engineering, power and water institute of technology* (pp. 102–110), Tehran, Iran.

Meybodi, M. R., & Beigy, H. (2004). A mathematical framework for cellular learning automata. *Journal of Advances in Complex Systems, 7*(3-4), 295–320.

Meybodi, M. R., Beigy, H., & Taherkhani, M., (2000). Application of cellular learning automata to image processing. In *Proceedings of first conference in mathematics and communication, iranian telecommunication research center* (pp. 23.1–23.10). Tehran, Iran.

Mitchell, M., (1996). Computation in cellular automata: A selected review, Technical report, Santa Fe Institute, Santa Fe, NM, USA.

Munoz, S., Ortuno, M. T., Ramirez, J., & Yanez, J. (2005). Coloring fuzzy graphs. *Omega, 33*, 211–221.

Narendra, K. S., & Thathachar, K. S. (1989). *Learning automata: An introduction*. New York: Printice-Hall.

Packard, N. H., & Wolfram, S. (1985). Two dimensional cellular automata. *Journal of Statistical Physics, 38*, 901–946.

Prestwich, S. (2008). Generalized graph coloring by a hybrid of local search and constraint programming. *Discrete Applied Mathematics, 156*, 148–158.

Talavan, P. M., & Yanez, J. (2008). The graph coloring problem: A neuronal network approach. *European Journal of Operational Research, 191*, 100–111.

Thathachar, M. A. L., & Sastry, P. S. (1997). A hierarchical system of learning automata that can learn the globally optimal path. *Information Science, 42*, 743–766.

Thompson, J., & Dowsland, K. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research, 25*, 637–648.

Voudouris, C., & Tsang, E. P. K. (2003). Guided local search. In F. Glover (Ed.), *Handbook of metaheuristics* (pp. 185–218). USA: Kluwer Academic Publishers.

Vredeveld, T., & Lenstra, J. K. (2003). On local search for the generalized graph coloring problem. *Operations Research Letters, 31*, 28–34.

Zymolka, A. M. C. A., Koster, & Wessaly, R., (2003). Transparent optical network design with sparse wavelength conversion. In *Proceedings of the 7th IFIP working conference on optical network design and modelling* (pp. 61–80). Budapest, Hungary.