A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem

[1]Mehdi Rezapoor Mirsaleh and [2]Mohammad Reza Meybodi

[1,2]Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

E-mail: (mrezapoorm, mmeybodi)@aut.ac.ir

Abstract

Vertex coloring problem is a combinatorial optimization problem in graph theory in which a color is assigned to each vertex of graph such that no two adjacent vertices have the same color. In this paper a new hybrid algorithm which is obtained from combination of cellular learning automata (CLA) and memetic algorithm (MA) is proposed for solving the vertex coloring problem. CLA is an effective probabilistic learning model combining cellular automata (CA) and learning automaton (LA). Irregular CLA (ICLA) is a generalization of CLA in which the restriction of rectangular grid structure in CLA is removed. The proposed algorithm is based on the irregular open CLA (IOCLA) that is an extension of ICLA in which the evolution of CLA is influenced by both local and global environments. Similar to other IOCLA-based algorithms, in the proposed algorithm, local environment is constituted by neighboring LAs of any cell and the global environment consists of a pool of memes in which each meme corresponds to a certain local search method. Each meme is represented by a set of LAs from which the history of the corresponding local search method can be extracted. To show the superiority of the proposed algorithm over some well-known algorithms, several computer experiments have been conducted. The results show that the proposed algorithm performs better than other methods in terms of running time of algorithm and the required number of colors.

Keywords*: Learning Automata (LA), Memetic Algorithm (MA), Vertex coloring problem*

1. Introduction

Vertex coloring problem is a well-known combinatorial optimization problem in graph theory which is used in many applications such as timetabling [1], air traffic flow management [2], register allocation [3] and scheduling [4]. Given graph $G = (\underline{V}, \underline{E})$ where $\underline{V}$ is the set of $|\underline{V}| = n$ vertices and $\underline{E}$ is the set of $|\underline{E}| = m$ edges, a k-coloring of graph G is a function $\Phi: \underline{V} \rightarrow \underline{\Gamma}$ where

$\underline{\Gamma} = \{1,2,\dots,k\}$ is the set of integers, each one represents one color. A coloring is feasible if no two endpoints of each edge have the same color. That is, for all $[u,v] \in E$ we have that $\Phi(u) \neq \Phi(v)$, otherwise coloring is infeasible. If the endpoints *u* and *v* of any edge have the same color, the vertices *u* and *v* are in conflict. The set of all vertices that are in conflict, form the conflict set $\underline{V^C}$. The minimum number of colors required for a feasible coloring is called the *chromatic number* $\chi(G)$ and a graph *G* is said *k-chromatic*, if $\chi(G) = k$. We can consider a feasible k-coloring as a partitioning of the set of vertices in *k* disjoint sets, named color classes and shown by $\underline{C} = \{C_1,\dots,C_k\}$. The k-coloring problem is NP-complete for general graph and chromatic number problem is also NP-hard [5]. The vertex coloring problem can be considered as a decision or an optimization problem. In decision version of the vertex coloring problem, the question to be answered is whether for some given *k*, a feasible k-coloring exists. The optimization version of vertex coloring problem intends to identify the smallest number *k* by which the graph can be feasibly colored.

LA is based on general schemes of reinforcement learning algorithms. LAs enable agents to learn their interaction with an environment. They select actions via a stochastic process and apply them on a random unknown environment. They can learn the best action by iteratively performing and receiving stochastic reinforcement signals from the unknown environment. These stochastic responses from the environment show the favorability of the selected actions, and the LAs change their action selecting mechanism in favor of the most promising actions according to responses from the environment [6, 7].

In this paper a new hybrid algorithm is proposed for solving the vertex coloring problem. This algorithm is obtained from combination of CLA and MA. A CLA is a CA in which one or more LAs are assigned to its every cell. The LA residing in a particular cell determines its action on the basis of its action probability vector. ICLA is a generalization of CLA in which the restriction of rectangular grid structure in CLA is removed. The proposed algorithm is based on the IOCLA which is an extension of ICLA in which the evolution of CLA is influenced by local and global environments. Similar to other IOCLA-based algorithms, in the proposed algorithm, local environment is constituted by neighboring LAs of any cell. On the other hand, the global environment consists of a pool of memes in which each meme corresponds to a certain local search method and is represented by a set of LAs from which the history of the corresponding local search can be extracted. In this algorithm, the input graph is modeled by an isomorphic

IOCLA in which each vertex of graph is associated with a cell of IOCLA, and then each cell is equipped with an LA whose actions constitute the set of colors by which the corresponding vertex can be colored. 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. At stage *t*, the LA of each cell chooses one of its actions (colors) according to its action probability vector. The selected action is applied to the local environment (neighboring LAs) as well as the global environment. Local environment produces a favorable or unfavorable response depending on the selected actions of neighboring LAs. That is, if the selected action (color) is also selected by the LAs of at least one of its neighboring cells or the number of selected actions (colors) by the cell and its neighbors is greater than the number of color used in the coloring found so far, then the local environment produces an unfavorable response. Otherwise, it produces a favorable response. Global environment consists of a pool of memes of size L and a solution which is initialized randomly and named *current solution*. Each meme corresponds to a certain local search method and is represented by a set of LAs using which the history of the corresponding local search method during the search process can be preserved and then used for computation of memetic fitness as will be described later. The action probability vectors of LAs used in a meme (the history) are updated according to a learning algorithm. If the color of a vertex in the current solution is the same before and after applying the local search method, the selected action of LA of corresponding vertex is rewarded. Otherwise, it is penalized. Global environment receives the selected actions of all the cells. A new solution is generated in global environment by combining all the selected actions (colors) of all the cells. The response of global environment is determined by comparing the fitness of new solution and the fitness of current solution which is improved by local search method. That is, if the fitness of new solution is better than the fitness of current solution, global environment produces a favorable response and the current solution is replaced by the new solution. Otherwise, it produces an unfavorable response and the current solution remains unchanged. In this paper, we compare the results of the proposed algorithm with the results of the best-known vertex coloring algorithms such as: TPA [8], AMACOL [9], ILS [10], CHECKCOL [11], GLS [12] and CLAVCA [13]. The obtained results show the superiority of the new method over the other algorithms in terms of running time of algorithm and required number of colors. This paper is organized as follows. In Section 2, an overview of related works on vertex graph coloring problem is represented. The theory of learning automata is described in

Section 3. The new proposed algorithm is described in Section 4. Sections 5 is including of implementation considerations, simulation results, and comparison with other algorithms to highlight the contributions of the new algorithm. Conclusions and future works are discussed in Section 6. In the appendix we provide a list of notations used in the proposed algorithm.

2. Related work

There are wide varieties of approaches that have been reported for solving the vertex coloring problem in the literature. These approaches are classified as exact and approximation techniques. The first exact algorithm was proposed in [14] to compute the chromatic number of graph. This algorithm finds the optimal solution with complexity order $O(2.4423^n)$. Also, an $O(2.40231^n)$ algorithm was proposed to compute the chromatic number of graph in [15]. Faster algorithms are reported in the literature when the number of coloring is fixed. An $O(1.3289^n)$ algorithm for $k=3$ to solve k-coloring problem is proposed in [16]. Also, an $O(1.7504^n)$ algorithm is proposed in [17] for $k=4$. Finally, two algorithms with complexity order $O(2.1020^n)$ and $O(2.3289^n)$ were proposed for $k=5$ [15] and for $k=6$ [18], respectively. Due to NP-hardness of vertex coloring problem for large graphs, exact *techniques* can only be used for small graphs, while there are very large graphs in a variety of applications. On the other hand, a near optimal coloring of graph can be used in many applications. Hence, different approximation algorithms have been proposed for finding the near optimal solution for vertex coloring problem in the literature.

A two-phased local search algorithm is proposed in [8] for the vertex coloring problem. The algorithm generates candidate solutions by alternately executing two interacting functionalities, namely, a stochastic and a deterministic local search. In stochastic phase, which is based on biased random sampling, the feasible colorings are created. Then, in deterministic phase, each vertex is assigned to the color in such a way that the solution penalty is minimized.

An iterated local search algorithm (ILS), which is based on a random walk in the space of the local optima, is proposed in [10]. The algorithm generates candidate solutions in three steps: 1) a walk is built by iteratively perturbing a local optimal solution. 2) a new local optimal solution is obtained by applying a local search algorithm, and 3) an acceptance criterion is used for selecting solutions to continue the search.

A priority-based local search algorithm, called CHECKCOL, is proposed in [11]. CHECKCOL decreases the running time of the algorithm by avoiding unnecessary searches in large parts of the graph without making any progress in the solution. To do this, it introduces the checkpoint notion to force the algorithm to stop at certain steps, release all of its memory and then it starts a local search. In this algorithm, a priority is dynamically assigned to each vertex of the graph. The priority is used to define a new effective long term memory scheme which is integrated with the short term memory scheme implied by the checkpoints.

In [19] a tabu-search algorithm was proposed to solve vertex coloring problem. In this algorithm the vertices of a graph are partitioned into several blocks and a different color is assigned to each block at each iteration. This technique makes the solutions feasible or infeasible. In the next step, to create a feasible coloring, a set of neighbors are generated for each vertex, restricted by a tabu-list, and then a new solution is obtained by applying local search algorithm. Tabu-list prevents premature convergence of the algorithm.

A hybrid algorithm based on genetic algorithm and tabu-search method is proposed in [20] to solve vertex coloring problem. The hybrid algorithm consists of a population of solutions and a crossover operator by which each vertex in the child chromosome inherits its color from one of the parent chromosomes. The mutation operator is replaced by a tabu-search method. The hybrid algorithm is first improved by selecting k-independent sets of vertices and then hybrid algorithm is applied on the remaining vertices.

A genetic local search algorithm is proposed in [21] to solve vertex coloring problem. This algorithm introduces a new crossover operator based on the union of independent sets (UIS) which is combined with a tabu-search method. The algorithm has been tested on several data sets and the results are compared with the results obtained from the best known methods. The comparison shows the superiority of the proposed algorithm.

An adaptive memory algorithm for k-coloring problem called AMACOL, was proposed in [9]. The AMACOL is a hybrid evolutionary algorithm in which the color classes that derived from the colorings generated during the previous stages of the search are stored in a central memory. At each generation, the central memory is used to generate the new solutions which are then improved by applying a local search method. Finally, the central memory is updated using the obtained solutions.

A hybrid evolutionary algorithm called HEA, was proposed in [22] for vertex coloring problem. HEA uses the DSATUR construction heuristic to initialize the population of chromosomes. At each generation, new chromosomes are generated by first recombining two parent chromosomes that are improved by local search method. The greedy partition crossover (GPX) is used as the crossover operator in HEA. GPX generates a new chromosome by alternately selecting color classes of each parent. The new chromosome generated by GPX is then improved by tabu search method and replaces the worse parent in current population.

3. Theory of automata

In this section, learning automata (LA) is introduced in brief. Then cellular learning automata (CLA) which is a combination of cellular automata and learning automata; and irregular open cellular learning automata (IOCLA) are presented.

3.1. Learning Automaton

A learning automaton [6] is an adaptive decision-making unit. It can be described as determination of an optimal action from a set of actions through repeated interactions with an unknown random environment. It selects an action based on a probability distribution at each instant and applies it on a random environment. The environment sends a reinforcement signal to automaton after evaluating the input action. The learning automaton process the response of environment and update its action probability vector. By repeating this process, the automaton learns to choose the optimal action so that the average penalty obtained from the environment is minimized. The environment is represented by a triple $< \underline{\alpha}, \underline{\beta}, \underline{c} >$ where $\underline{\alpha} = \{\alpha_1, \dots, \alpha_r\}$ is the finite set of the inputs, $\underline{\beta} = \{\beta_1, \dots, \beta_m\}$ is the set of outputs that can be taken by the reinforcement signal, and $\underline{c} = \{c_1, \dots, c_r\}$ is the set of the penalty probabilities, where each element c_i of $\underline{c}$ is corresponds to one input action α_i. When the penalty probabilities are constant, the random environment is said a stationary random environment. It is called a non stationary environment, if they vary with time. Depending on the nature of the reinforcement signal, there are three types of environments: P-model, Q-model and S-model. The environments, in which the output can take only one of two values 0 or 1, are referred to as P-model environments. The

reinforcement signal in Q-model environment selects a finite number of the values in the interval $[a, b]$. When the output of environments is a continuous random variable in the interval $[a, b]$, it is referred to as S-model. The relationship between the learning automaton and the random environment is shown in Fig. 1.

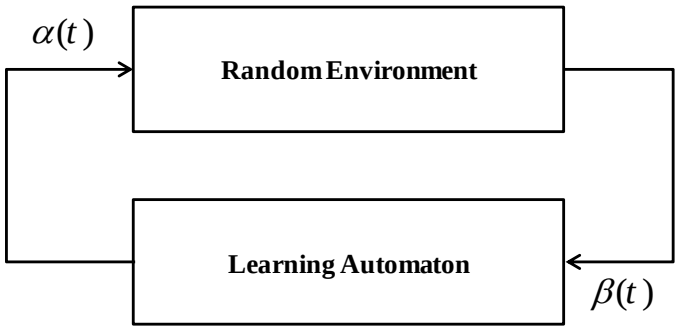

Fig. 1: The relationship between the learning automaton and random environment

There are two main families of Learning automata [7]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}, \underline{\alpha}, T >$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of output actions, and T is learning algorithm which is used to modify the action probability vector. Learning algorithm is the critical factor affecting the performance of variable structure learning automata. Suppose learning automaton selects action $\alpha_i(t) \in \underline{\alpha}$ according to action probability vector $\underline{p}(t)$ at instant *t*. The action probability vector $\underline{p}(t)$ is updated by the learning algorithm given in Eq. (1), if the selected action $\alpha_i(t)$ is rewarded by the random environment, and it is updated as given in Eq. (2), if the taken action is penalized. *a* and *b* denote the reward and penalty parameters and *r* is the number of actions that can be taken by learning automaton.

$$P_j(t+1) = \begin{cases} P_j(t) + a(1 - P_j(t)) & j = i \\ (1-a)P_j(t) & \forall j, j \neq i \end{cases} \quad (1)$$

$$P_j(t+1) = \begin{cases} (1-b)P_j(t) & j = i \\ b/(1-r) + (1-b)P_j(t) & \forall j, j \neq i \end{cases} \quad (2)$$

If $a = b$, the recurrence equations (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a \gg b$ the given equations are called linear reward-ϵ penalty ($L_{R\varepsilon P}$), and finally if $b = 0$ they are called linear reward-Inaction (L_{R-I}). In the latter case, the action probability vector remains unchanged when the taken action is penalized by the environment.

Learning automata have a vast variety of applications in combinatorial optimization problems [23-26], computer networks [25, 27-30], queuing theory [31, 32], signal processing [33, 34], information retrieval [35, 36], adaptive control [37-39], neural networks engineering [40, 41] and pattern recognition [42-44].

3.2. Cellular learning automata (CLA)

Cellular learning automata (CLA), which is a combination of cellular automata (CA) and learning automata (LA), is a mathematical model for solving problems. Because of the ability of CLA to learn, this model is superior to CA and also because of the equipping CLA with a collection of LAs which can interact with each other, this model is superior to LA. The basic idea of CLA is to use LA to adjust the state transition probability of CA. the operation of CLA can be described as follows: at the first step, the state of every cells is determined on the basis of action probability vector of the LA residing in that cell. The initial value of this state may be chosen randomly or on the basis of the past experience. In the second step, the rule of CA determines the reinforcement signal to each LA residing in that cell. Finally, each LA 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.

A d-dimensional cellular learning automata is a structure $\mathcal{A} = \left(\mathrm{Z}^{\mathrm{d}}, \underline{\Phi}, \underline{\mathrm{A}}, \underline{\mathrm{N}}, \mathcal{F}\right)$, where Z^d is a lattice of d-tuples of integer numbers. $\underline{\Phi}$ is a finite set of states. $\underline{A}$ is the set of learning automata each of which is assigned to each cell of the CA. $\underline{N} = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{\bar{m}}\}$ is a finite subset of Z^d called neighborhood vector, where $\bar{x}_i \in Z^d$. $\mathcal{F}: \underline{\Phi}^{\bar{m}} \to \underline{\beta}$ 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 reinforcement signal to each learning automaton based on the current actions selected by its neighboring learning automata.

3.3. Irregular Open Cellular Learning Automata (IOCLA)

An irregular cellular learning automaton (ICLA) is a CLA in which the restriction of rectangular grid structure in CLA is removed. There are many applications such as wireless networks, graph related applications, etc. that cannot be adequately modeled with CLA. An ICLA is defined as an undirected graph in which, each node represents a cell which is equipped with a learning automaton (LA). The LA residing in each cell selects its action on the basis of its action probability vector. The local rule and the actions selected by any particular LA and its neighboring learning automata determine the reinforcement signal to the LA residing in each cell (The local rule in ICLA is the same as the local rule in CLA).

An irregular open cellular learning automaton (IOCLA) is a new class of ICLA in which the evolution of CLA is influenced by local and external environments. In IOCLA, the neighboring learning automata of any cell constitute its local environment. Two types of external environments can be considered in the IOCLA: global environment and exclusive environment. Each IOCLA has one global environment that influences all cells and an exclusive environment for each particular cell influencing the evolution of that cell. The interconnection of a typical cell c_i and its various types of environments are shown in Fig. 2.

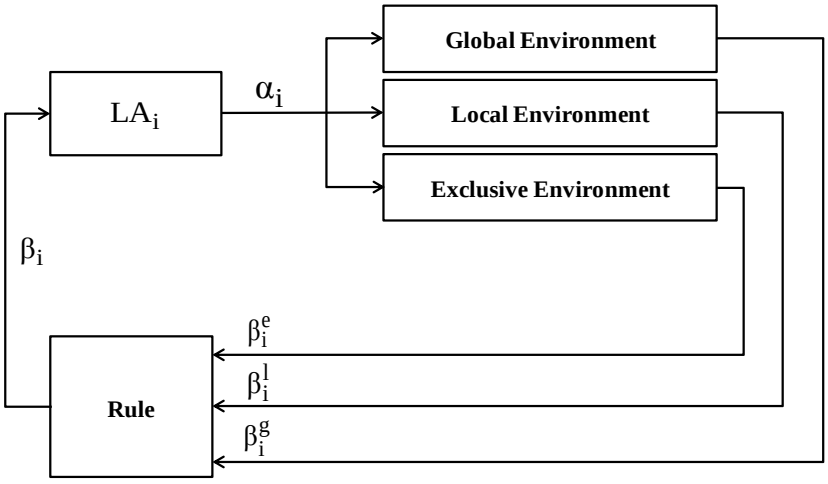

Fig. 2 - The interconnection of a typical cell in IOCLA with its various environments.

The operation of IOCLA could be described as follows: At iteration *t*, each learning automaton chooses one of its actions. Let α_i be the action chosen by learning automaton LA_i. The actions of

all learning automata are applied to their corresponding local environments (neighboring learning automata) as well as the global environment and their corresponding exclusive environments. Each environment produces a signal, which is used by the local rule to generate a reinforcement signal to the learning automaton residing in every cell. Let β_i^l, β_i^e and β_i^g be the reinforcement signals generated by local, exclusive and global environments of cell c_i, respectively. The local rule of IOCLA determines the reinforcement signal to each learning automaton LA_i using β_i^l, β_i^e and β_i^g. Finally, all learning automata update their action probability vectors in the basis of supplied reinforcement signal by local rule (β_i). This process continues until the desired result is obtained.

4. The proposed algorithm for vertex coloring problem

In this section, we propose a CLA-based memetic algorithm, called CLAMACOL, for solving the vertex coloring problem. In the proposed algorithm, an IOCLA isomorphic to input graph is generated. For this purpose, each vertex v_i of the graph is associated with cell c_i of IOCLA where the cell is equipped with learning automaton LA_i. The set of actions of LA_i constitutes the set of colors by which the vertex v_i can be colored. The resulting IOCLA can be model by a duple $< \underline{A}, \underline{a} >$ where $\underline{A} = \{LA_1, LA_2, \dots, LA_n\}$ is the set of LAs corresponding to vertices of the graph and $\underline{a} = \{\underline{a_1}, \underline{a_2}, \dots, \underline{a_n}\}$ denotes the action-set of LAs in which $\underline{a_i} = \{a_i^1, a_i^2, \dots, a_i^{\Delta_i+1}\}$ is the set of actions (colors) that can be taken by LA_i (the set of colors by which the vertex v_i can be colored). Also, n is the number of vertices and Δ_i is the degree of vertex v_i. It has been shown in [45] that an arbitrary graph can be colored with at most $\Delta + 1$ colors, where Δ denotes the graph degree. Therefore, it can be concluded that vertex v_i and its neighboring vertices can be colored with at most $\Delta_i + 1$ colors in the worst case. That is why, in the proposed algorithm, the action-set of LA_i (corresponding to color-set of vertex v_i) is composed of $\Delta_i + 1$ actions (or colors). The operation of the proposed algorithm which composed of a number of stages can be described as follows. At stage t, the LA at cell $c_i(LA_i)$ chooses one of its actions (colors) randomly, based on its action probability vector which is initially set to $\frac{1}{\Delta_i+1}$. The selected action ($\alpha_i \in \underline{a_i}$) is applied to the local environment of cell c_i (neighboring LAs) as well as the global environment and its corresponding exclusive environment. Local environment produces a signal

(β_i^l). That is, if the selected action (color) is also selected by the LAs of at least one of its neighboring cells, then β_i^l is set to 1. Otherwise β_i^l is set to 0. The cardinality of the set of colors which are selected by the LA_i at cell c_i and its neighboring cells are named color-degree and denoted as D_i. D_i is computed in the exclusive environment of cell c_i and β_i^e is generate based on it. That is, if D_i is greater than its own dynamic threshold T_i then β_i^e is set to 1. Otherwise β_i^e is set to 0. Dynamic threshold T_i which retains the minimum value of D_i, that has been seen so far for cell c_i, can be initially set to $\Delta_i + 1$. Global environment receives all of the selected actions (colors) of LAs. The actions (colors) chosen by the LAs generate a new solution for coloring of graph. The value of β_i^g is determined by comparing the genetic fitness of new solution and the genetic fitness of current solution which is improved by local search. That is, if the genetic fitness of new solution is better than the genetic fitness of current solution, β_i^g is set to 0 and the current solution is replaced by the new solution. Otherwise, β_i^g is set to 1 and the current solution remains unchanged. The genetic fitness function counts the number of edges that their end points have the same color class. Formally, the genetic fitness of current solution at stage *t* can be described as $GF(t) = 1/(1 + \sum_{j=1}^{k} |\underline{E_j}|)$, where $|\underline{E_j}|$ is the cardinality of set of edges with both end points in color class C_j. LA_i rewards its selected action, if all three local, exclusive and global environments produce favorable responses ($\beta_i^l = 0, \beta_i^e = 0$ and $\beta_i^g = 0$). Otherwise it penalizes its selected action. Global environment consists of a pool of memes of size *L*. The size of meme pool is equal to the number of local search methods. Each local search method corresponds to one of the memes in the meme pool. Each meme saves the history of the effect of its corresponding local search method. Each meme is composed of *n* LAs that each of them corresponds to a vertex of the current solution. Specifically, the i_{th} automaton corresponds to the i_{th} vertex of the current solution. Each LA has $\Delta_i + 1$ actions corresponding to $\Delta_i + 1$ possible colors that can be taken by vertex v_i. The history of the effect of γ_{th} local search method at stage *t* is represented by the action probability vectors of LAs in the γ_{th} meme as given Eq. (3).

$$\underline{M^{\gamma}(t)} = [\underline{M_1^{\gamma}(t)}, \underline{M_2^{\gamma}(t)}, \dots, \underline{M_n^{\gamma}(t)}] \quad (3)$$

where

$$\underline{M_j^{\gamma}(t)} = [M_{j1}^{\gamma}(t), M_{j2}^{\gamma}(t), \dots, M_{j(\Delta_j+1)}^{\gamma}(t)]', 1 \le j \le n, \sum_{p=1}^{\Delta_j+1} M_{jp}^{\gamma}(t) = 1 \; \forall \gamma, j.$$

$M_{jp}^{\gamma}(t)$ denotes the probability that action p of j_{th} leaning automaton in meme γ (corresponding to the color p for vertex *j*) is selected. The probability of selecting an action (color) by j_{th} LA is initially set to $\frac{1}{\Delta_j+1}$. The action probability vectors of a meme (the history) are updated according to a learning algorithm based on the reinforcement signal received after applying the corresponding local search method on current solution. The aim of the set of LAs of a meme is to find those colors of the current solution that result in a better solution. $MF_{\gamma}(t) = \prod_{j=1}^{n} M_{jp}^{\gamma}(t)$ where p is the action (color) selected by automaton j in meme γ is the probability that the current solution is generated by applying local search method γ at stage *t*. $MF_{\gamma}(t)$ is referred to as memetic fitness of the current solution after applying meme γ at stage *t*. Memetic fitness changes when the action probability vectors of LAs in the meme is updated. Updating is performed on the basis of the result of applying corresponding local search method on current solution. If the values of vertex j of current solution are the same before and after applying local search method γ on current solution, the action of j_{th} LA of meme γ which corresponds to the value of j_{th} vertex is rewarded, or penalized, otherwise. It has been shown in [46] that applying a local search method on current solution, for example local search method γ, is beneficial if $\frac{(1-MF_{\gamma}(t))^2}{MF_{\gamma}(t)} > \frac{(1-GF(t))}{GF(t)}$ where $GF(t)$ is the genetic fitness of current solution at stage *t*. The learning process continues for each LA_i in IOCLA, if the choice probability of an action (color) exceeds a pre-specified threshold, e.g., π_i. Different local search methods have been proposed for vertex coloring problem in the literature[12]. Next, we briefly review five local search methods used in proposed algorithm.

- *1-exchange local search*

The most frequently used local search is the 1-exchange local search in which one vertex v_i is selected randomly and moved from its current color class i.e., C_j, into a different color class i.e., C_l where $l \neq j$ and $l, j \in \{1, \dots, \Delta_i + 1\}$.

- *Swap local search*

In the swap local search exactly one vertex $v \in \underline{V^C}$ exchange the color class with another vertex $v \in V$.

- *Restricted 1-exchange local search*

The 1-exchange local search, which is restricted to change only the color class of vertices that are in conflict, since only these modifications can lead to a increase of the genetic fitness function; called restricted 1-exchange local search.

- *Cyclic exchange local search*

An extension of the 1-exchange and swap local search is the cyclic exchange local search. The cyclic exchange local search is a sequence of 1-exchanges local search. A cyclic exchange of length m acts on a sequence of distinctly colored vertices $(u_1, \ldots, u_m)$. For simplicity, we will denote the color class of any u_i, $i = 1, \ldots, m$, by C_i. The cyclic exchange local search moves any vertex u_i, $i = 1, \ldots, m$, from C_i into C_{i+1}. We use the convention $C_{m+1} = C_1$. A cyclic exchange local search does not change the cardinality of the color classes involved in the move. Fig. 3 (a) gives an example of cycle exchange local search method.

- *Path exchange local search*

The path exchange local search is the same as cycle exchange local search expect that the u_m remains in C_m. Hence the sequence of exchanges is not closed and the cardinality of C_1 and C_m is modified. Fig. 3 (b) gives an example of a path exchange local search method.

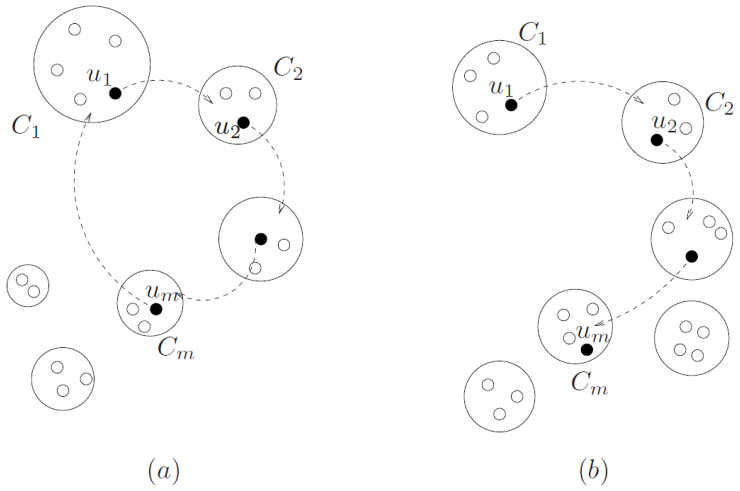

Fig. 3 - An example of cycle exchange local search (a) and path exchange local search (b)

Pseudo code for proposed algorithm demonstrated in Fig. 4:

Proposed algorithm for vertex coloring algorithm

Input: Cell c_i, threshold π_i
Output: *Current Solution[i]*
Begin Algorithm
  Vertex v_i is associated with cell c_i;
  Cell c_i is equipped with automaton LA_i;
  Automaton LA_i forms its action-set;
  $T_i \leftarrow \Delta_i + 1$; *t=0*;
  Repeat
    New Solution[i] = Automaton LA_i choose one of its color randomly;
    Automaton LA_i computes its color-degree D_i;
    //---------- Local Environment ----------
    If (the color selected by automaton LA_i is chosen by at least one of its adjacent automata) **Then**
      $\beta_i^l \leftarrow 1$;
    Else
      $\beta_i^l \leftarrow 0$;
    End If
    //---------- Exclusive Environment ----------
    If ($D_i \leq T_i$) **Then**
      $\beta_i^e \leftarrow 0$;
      If (β_i^l of automaton LA_i and its adjacent automata are zero) **Then**
        $T_i \leftarrow D_i$;
      End If
    Else
      $\beta_i^e \leftarrow 1$;
    End If
    //---------- Global Environment ----------
    If (genetic fitness of new solution > genetic fitness of current solution) **Then**
      $\beta_i^g \leftarrow 0$;
      Current Solution ← *New Solution*.
    Else
      $\beta_i^g \leftarrow 1$;
    End If
    //----------
    If ($\beta_i^g = 0$ **AND** $\beta_i^e = 0$ **AND** $\beta_i^l = 0$) **Then**
      Automaton LA_i rewards its selected action (color);
    Else
      Automaton LA_i penalizes its selected action (color);
    End If
    For each local search γ
      If ($\frac{(1-MF_\gamma(t))^2}{MF_\gamma(t)} > \frac{(1-GF(t))}{GF(t)}$) **Then**
        Apply γ_{th} local search on current solution;
        Update the action probability vectors of meme $M^\gamma(t)$ according to learning algorithm;
      End If
    End For
    t=t+1;
  Until(The probability with which automaton LA_i choose an action (color) > π_i)
End Algorithm

Fig. 4. Pseudo code for CLAMACOL

5. Experimental Results

In this section several experiments have been conducted to show the efficiency of the proposed algorithm. The results of the proposed algorithm are compared with the results of the best-known vertex coloring algorithms such as: TPA [8], AMACOL [9], ILS [10], CHECKCOL [11], GLS [12] and CLAVCA [13]. The experiments are implemented by MATLAB 2009a running on a PC with a 2.83 GHz processor and 4 GB memory. In these experiments, learning rate of all leaning automata is set to 0.1, and the algorithm terminates when the learning automaton in each cell of IOCLA has an action with probability at least 0.95. For this purpose, we have used a database of hard-to-color benchmarks DSJ graphs [47], Leighton graphs [4] and WAP graphs [8]. The characteristics of the used benchmark graphs are given in Table 1, Table 3 and Table 5, respectively. Table 2, Table 4 and Table 6 show the comparison of obtained results for different algorithms. In these tables, for all of the algorithms, the first column (color) represents the number of colors required for coloring the graph and the second column (time) shows the running time of each algorithm in seconds.

The first class of the benchmark graphs on which the proposed algorithm is tested is DSJ graphs. This class comprises graphs of a variable number vertices, *n*, where each of the *n(n-1)/2* possible edges is generated independently at random with probability *p*. DSJ graphs are denoted as *DSJCn.p* where $n \in \{125,250,500,1000\}$ and $p \in \{0.1, 0.5, 0.9\}$. The characteristics of DSJ benchmark graphs are given in Table 1.

Table 1 - The characteristics of DSJ benchmark graphs

Graph Name	Number of vertices	Number of edges	Density
DSJC125.1	125	736	0.1
DSJC125.5	125	3891	0.5
DSJC125.9	125	6961	0.9
DSJC250.1	250	3218	0.1
DSJC250.5	250	15668	0.5
DSJC250.9	250	27897	0.9
DSJC500.1	500	12458	0.1
DSJC500.5	500	62624	0.5
DSJC500.9	500	112437	0.9
DSJC1000.1	1000	49629	0.1
DSJC1000.5	1000	249826	0.5
DSJC1000.9	1000	449449	0.9

Table 2 shows the results of simulation experiments conducted on DSJ benchmark graphs. Comparing the results reported in Table 2, we find that, in most cases, GLS outperforms the others in terms of running time. It can be seen that AMACOL always select the smallest color-set for coloring the graphs. Comparing the CLAMACOL with GLS, it can be seen that the size of the color-set, in the CLAMACOL, is significantly smaller as compared with GLS in most cases. On the other hand, comparing the result of proposed algorithm and AMACOL, we find that, in most cases, the color-sets chosen by CLAMACOL are as small as those AMACOL, while the running time of the CLAMACOL is considerably shorter than AMACOL.

Table 2 - A performance comparison of the proposed algorithm and the most effective coloring algorithms on DSJ benchmarks graphs

Graph	TPA		AMACOL		ILS		CHECKCOL		GLS		CLAVCA		CLAMACOL	
Name	color	time	color	time	color	time	color	time	color	time	color	time	color	time
DSJC125.1	**5**	**0**	**5**	**0**	**5**	**0**	**5**	**0**	**5**	**0**	**5**	**0**	**5**	**0.002**
DSJC125.5	19	289	17	125	**17**	**2**	17	110	18	0	17	12	17	8.06
DSJC125.9	44	5	44	57	**44**	**0**	44	4	**44**	**0**	44	19	44	12.35
DSJC250.1	8	10	8	12	**8**	**0**	8	28	9	0	8	7	9	4.35
DSJC250.5	30	3282	28	64	28	34	28	557	30	1	28	27	**28**	**18.36**
DSJC250.9	72	155	72	2604	**72**	**6**	72	182	73	6	73	32	72	21.85
DSJC500.1	**12**	**0**	12	9	13	0	12	4	13	0	12	20	13	10.63
DSJC500.5	48	124	48	326	50	106	48	1789	52	81	48	42	**48**	**28.68**
DSJC500.9	127	1268	126	1710	128	82	126	2045	130	154	126	70	**126**	**71.36**
DSJC1000.1	21	28	**20**	**969**	21	6	21	142	22	1	21	39	22	18.34
DSJC1000.5	84	2386	84	9235	91	303	84	7025	93	546	**84**	**88**	85	86.75
DSJC1000.9	226	3422	224	4937	228	2245	226	12545	234	1621	224	119	**224**	**76.14**

Leighton benchmark graphs are the second class which the proposed algorithm is tested on. Leighton graphs are random graphs of density below 0.25, which are constructed by first partitioning vertices into k distinct sets representing the color classes and then assigning edges only between vertices that belong to different sets. The chromatic number of these graphs is guaranteed to be k by setting cliques of sizes ranging from 2 to k into the graph.

The Leighton benchmark graphs are denoted as *Le_kx*, where k is the chromatic number of the graph and $x \in \{a, b, c, d\}$ is a letter used to distinguish different graphs with the same characteristics, with c and d graphs having higher edge density than the a and b ones. The characteristics of Leighton benchmark graphs are given in Table 3.

Table 3 - The characteristics of Leighton benchmark graphs

Graph Name	Number of vertices	Number of edges	Density
Le_5a	450	5714	0.06
Le_5b	450	5734	0.06
Le_5c	450	9803	0.1
Le_5d	450	9757	0.1
Le_15a	450	8168	0.08
Le_15b	450	8169	0.08
Le_15c	450	16680	0.17
Le_15d	450	16750	0.17
Le_25a	450	8260	0.08
Le_25b	450	8263	0.08
Le_25c	450	17343	0.17
Le_25d	450	17425	0.17

Table 4 shows the results of the conducted experiments on Leighton benchmark graphs. Comparing the results reported in Table 4, we observe that, in most cases, ILS outperforms the other algorithms in terms of running time. It can also be seen that CHECKCOL, in most cases, selects the smallest color-set for coloring the benchmark graphs, while its running time is the worst. Comparing the result of CLAMACOL with the minimum number of required colors for coloring benchmark graphs, we observe that, in most cases, the size of the color-set obtained by the proposed algorithm is equal to the minimum number of required colors for coloring the benchmark graphs. It can also be seen that, the proposed algorithm significantly outperforms CHECKCOL in terms of running time.

Table 4 - A performance comparison of the proposed algorithm and the most effective coloring algorithms on Leighton benchmarks graphs

Graph	TPA		AMACOL		ILS		CHECKCOL		GLS		CLAVCA		CLAMACOL	
Name	color	time	color	time	color	time	color	time	color	time	color	time	color	time
Le_5a	5	125	5	127	**5**	**0**	5	108	5	3	5	9	5	2.28
Le_5b	5	96	5	86	**5**	**0**	5	110	5	2	5	11	5	1.09
Le_5c	5	39	5	7	**5**	**0**	5	2	5	6	5	8	5	1.49
Le_5d	5	18	5	3	**5**	**0**	5	2	5	9	5	5	5	1.37
Le_15a	15	1444	15	345	**15**	**0**	15	2145	15	2	15	23	15	12.36
Le_15b	15	1655	15	345	**15**	**0**	15	2756	**15**	**0**	15	18	15	12.34
Le_15c	15	82	**15**	**2**	15	19	15	4534	15	6	15	30	15	10.36
Le_15d	15	34	**15**	**4**	15	20	15	4576	15	8	15	32	15	13.67
Le_25a	26	2451	25	1365	25	49	**25**	**3**	25	34	26	28	25	4.68
Le_25b	26	1253	25	865	25	36	**25**	**3**	27	68	26	23	25	5.68
Le_25c	26	44	26	93	26	2	**25**	**3477**	26	18	26	30	27	8.05
Le_25d	26	22	26	10	26	1	25	4524	26	2	**25**	**46**	26	16.75

The third class of benchmark graphs on which the proposed algorithm is tested on is WAP graphs which arise in the design of transparent optical networks where each vertex corresponds to a light path in the network and the edges correspond to intersecting paths. These graphs are denoted by *WAP0m*, where $m \in \{1,2,\dots,8\}$. They have between 905 and 5231 vertices and all instances have a clique of size 40. The characteristics of WAP benchmark graphs are given in Table 5.

Table 5 - The characteristics of Wap benchmark graphs

Graph Name	Number of vertices	Number of edges	Density
WAP01a	2368	110871	0.04
WAP02a	2464	111742	0.04
WAP03a	4730	286722	0.03
WAP04a	5231	294902	0.02
WAP05a	905	43081	0.11
WAP06a	947	43571	0.1
WAP07a	1809	103368	0.06
WAP08a	1870	104176	0.06

Table 6 shows the results of the conducted experiments on WAP benchmark graphs. Comparing the results given in Table 6, we find that, TPA and GLS outperform the other algorithms in terms of the required number of colors for coloring the WAP benchmark graphs. It also can be seen that, in most cases, ILS outperforms others in terms of the running time. The obtained results reported in Table 6 show that, in most cases, both the running time and the size of color-sets created by the proposed algorithm is smaller than that of the best reported results.

Table 6 - A performance comparison of the proposed algorithm and the most effective coloring algorithms on WAP benchmarks graphs

Graph	TPA		AMACOL		ILS		CHECKCOL		GLS		CLAVCA		CLAMACOL	
Name	color	time	color	time	color	time	color	time	color	time	color	time	color	time
Wap01a	42	245	45	345	44	2	44	568	42	55	42	16	**42**	**10.05**
Wap02a	41	1618	44	802	43	251	43	486	41	160	41	28	**41**	**14.36**
Wap03a	44	17	53	245	46	365	46	689	44	782	**43**	**51**	44	29.65
Wap04a	43	95	48	45	44	484	44	23	43	834	43	47	**43**	**32.65**
Wap05a	50	257	50	263	51	125	51	79	50	58	51	69	**50**	**16.35**
Wap06a	41	348	44	545	42	1	42	25	41	8	**40**	**3**	42	1.86
Wap07a	42	541	45	89	44	1	44	182	42	215	**40**	**14**	44	12.59
Wap08a	42	200	45	446	43	56	44	22	42	41	**42**	**26**	44	14.67

Table 7 shows the ranking of comparing algorithms on different benchmarks graphs. The ranking is calculated based on the number of colors required for coloring the graph and the running time

of each algorithm in seconds. From the results reported in Table 7, we observe that, the average rank of CLAMACOL (3.25) is smaller than the average rank of other algorithms.

Table 7 – The ranking of comparing algorithms (TPA, AMACOL, ILS, GLS, CLAVCA and CLAMACOL) on different benchmarks graphs

Graph Name	TPA	AMACOL	ILS	CHECKCOL	GLS	CLAVCA	CLAMACOL
DSJC125.1	3.5	3.5	3.5	3.5	3.5	3.5	7
DSJC125.5	7	5	**1**	4	6	3	2
DSJC125.9	4	7	1.5	3	1.5	6	5
DSJC250.1	3	4	**1**	5	6	2	7
DSJC250.5	7	4	3	5	6	2	**1**
DSJC250.9	4	5	**1**	3	6	7	2
DSJC500.1	**1**	3	5.5	2	5.5	4	7
DSJC500.5	3	4	6	5	7	2	**1**
DSJC500.9	5	3	6	4	7	2	**1**
DSJC1000.1	3	**1**	2	5	6	4	7
DSJC1000.5	2	4	6	3	7	**1**	5
DSJC1000.9	4	3	6	5	7	2	**1**
Le_5a	6	7	**1**	5	3	4	2
Le_5b	6	5	**1**	7	3	4	2
Le_5c	7	5	**1**	3	4	6	2
Le_5d	7	4	**1**	3	6	5	2
Le_15a	6	5	**1**	7	2	4	3
Le_15b	6	5	1.5	7	1.5	4	3
Le_15c	6	**1**	4	7	3	5	2
Le_15d	6	**1**	4	7	2	5	3
Le_25a	7	5	4	**1**	3	6	2
Le_25b	6	4	3	**1**	7	5	2
Le_25c	5	6	2	**1**	3	4	7
Le_25d	7	5	3	2	4	**1**	6
Wap01a	4	7	5	6	3	2	**1**
Wap02a	4	7	5	6	3	2	**1**
Wap03a	2	7	5	6	4	**1**	3
Wap04a	3	7	6	5	4	2	**1**
Wap05a	3	4	7	6	2	5	**1**
Wap06a	3	7	4	6	2	**1**	5
Wap07a	3	7	4	6	2	**1**	5
Wap08a	3	7	4	6	2	**1**	5
Average rank	4.578125	4.765625	3.40625	4.546875	4.125	3.328125	**3.25**

We performed a non-parametric test (wilcoxon rank sum test) at the 95% significance level to provide statistical confidence. Table 8 shows the p-values of the two-tailed wilcoxon rank sum test. From Table 8, the difference between the rank of the CLAMACOL algorithm and the rank of most other algorithms (TPA, AMACOL, CHECKOL and GLS algorithms) are statistically significant (p-value<0.05).

Table 8 – The results of statistical tests for CLAMACOL algorithm and other algorithms

CLAMACOL vs.	TPA	AMACOL	ILS	CHECKCOL	GLS	CLAVCA
p-values	**6.035E-03**	**5.913E-03**	7.524E-01	**1.537E-02**	**4.471E-02**	7.321E-01

6. Conclusion

In this paper, a new CLA based memetic algorithm is proposed for finding a near optimal solution for vertex coloring problem, denoted as CLAMACOL. The proposed algorithm is based on the IOCLA which is an extension of ICLA in which the evolution of CLA is influenced by local and global environments. Similar to other IOCLA-based algorithms, the local environment of proposed algorithm is constituted by the neighboring LAs of any cell. On the other hand, the global environment consists of a pool of memes in which each meme corresponds to a certain local search method. Each meme is represented by a set of LAs where the history of the corresponding local search method can be extracted. Each environment produces a signal, which is used by the local rule to generate a reinforcement signal to the learning automaton residing in every cell. All learning automata update their action probability vectors in the basis of supplied reinforcement signal by local rule. This process continues until the desired result is obtained. To show the efficiency of the proposed algorithm, several computer simulations are conducted on hard-to-color benchmark graphs. The results show that the proposed algorithm performs better than the well-known algorithms both in terms of the required number of colors and the running time of algorithm. This line of research could be extended in several directions, such as in applying the proposed algorithm in solving other forms of vertex coloring problem such as multi-coloring, bandwidth coloring, and bandwidth multi-coloring problems. Also, the proposed algorithm could be applied to real-world dynamic problems and its efficiency can be evaluated. Another direction that may be pursued is the improving the proposed algorithms by designing new genetic fitness or memetic fitness functions.

Appendix

The notations used in the proposed algorithm are listed in Table 9.

Table 9 – Notations used in the proposed algorithm

G	Input graph
$\underline{V}$	Set of vertices of graph G
$\underline{E}$	Set of edges of graph G
$\underline{V^C}$	Set of vertices that are in conflict
$\underline{C}$	Color classes
C_j	Set of vertices with color j
$\lvert\underline{E_j}\rvert$	Cardinality of set of edges with both end points in color j
n	Number of vertices
v_i	i_{th} vertex of graph G
c_i	i_{th} cell of IOCLA associated to vertex v_i
$\underline{A}$	Set of learning automata corresponding to vertices of the graph
$\underline{a}$	Action-set of learning automata $\underline{A}$
LA_i	Learning automaton corresponding to vertex v_i
$\underline{a_i}$	Set of actions (colors) that can be taken by LA_i
Δ_i	Degree of vertex v_i
Δ	Graph degree
β_i^l	Reinforcement signals generated by local environment of cell c_i
β_i^e	Reinforcement signals generated by exclusive environment of cell c_i
β_i^g	Reinforcement signals generated by global environment of cell c_i
D_i	Cardinality of set of colors which are selected by the LA_i at cell c_i and its neighboring cells (color-degree)
$\underline{M^\gamma(t)}$	History of the effect of γ_{th} local search method at stage t (the information of meme γ at stage t)
$M_{jp}^\gamma(t)$	Probability of action p of j_{th} leaning automaton in meme γ
$MF_\gamma(t)$	Memetic fitness of the current solution after applying meme γ at stage t
$GF(t)$	Genetic fitness of current solution at stage t
π_i	Pre-specified threshold for probability of an action of LA_i
T_i	Dynamic threshold for color-degree of vertex v_i

References

[1] R. Lewis and B. Paechter, "Finding Feasible Timetables Using Group-Based Operators," *Evolutionary Computation, IEEE Transactions on,* vol. 11, pp. 397-413, 2007.

[2] N. Barnier and P. Brisset, "Graph coloring for air traffic flow management," *Annals of Operations Research,* vol. 130, pp. 163-178, 2004.

[3] G. J. Chaitin, M. A. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins, and P. W. Markstein, "Register allocation via coloring," *Computer Languages,* vol. 6, pp. 47-57, 1981.

[4] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of research of the national bureau of standards,* vol. 84, pp. 489-506, 1979.

[5] R. M. Karp, "Reducibility among combinatorial problems. RE Miller and JW Thatcher editors, Complexity of Computer Computations, 85-103," ed: Plenum Press, 1972.

[6] K. S. Narendra and M. A. L. Thathachar, *Learning automata: an introduction*: Prentice-Hall, Inc., 1989.

[7] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 32, pp. 711-722, 2002.

[8] M. Caramia 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.

[9] P. Galinier, A. Hertz, and N. Zufferey, "An adaptive memory algorithm for the k-coloring problem," *Discrete Applied Mathematics,* vol. 156, pp. 267-279, 2008.

[10] H. R. Lourenço, O. Martin, T. Stutzle, F. Glover, and G. Kochenberger, "Iterated Local Search," *Handbook of Metaheuristics,* pp. 321-353, 2002.

[11] M. Caramia, P. Dell'Olmo, and G. F. Italiano, "CHECKCOL: Improved local search for graph coloring," *Journal of Discrete Algorithms,* vol. 4, pp. 277-298, 2006.

[12] M. Chiarandini, I. Dumitrescu, and T. Stützle, "Stochastic local search algorithms for the graph colouring problem," *Handbook of approximation algorithms and metaheuristics,* pp. 1-63, 2007.

[13] J. Akbari Torkestani and M. R. Meybodi, "A cellular learning automata-based algorithm for solving the vertex coloring problem," *Expert Systems with Applications,* vol. 38, pp. 9237-9247, 2011.

[14] E. L. Lawler, "A note on the complexity of the chromatic number problem," *Information Processing Letters,* vol. 5, pp. 66-67, 1976.

[15] B. Jesper Makholm, "Exact algorithms for graph colouring and exact satisfiability," *Operations Research Letters,* vol. 32, pp. 547-556, 2004.

[16] R. Beigel and D. Eppstein, "3-coloring in time O (n1. 3289)," *Journal of Algorithms,* vol. 54, pp. 168-204, 2005.

[17] J. M. Byskov, "Enumerating maximal independent sets with applications to graph colouring," *Operations Research Letters,* vol. 32, pp. 547-556, 2004.

[18] M. Rezapoor M. and M. R. Meybodi, "LA-MA: A new memetic model based on learning automata," in *Proceeding of 18th National Conference of Computer Society of Iran,* Tehran, Iran, 2013, pp. 1-16.

[19] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring," *Computing,* vol. 39, pp. 345-351, 1987.

[20] C. Fleurent and J. A. Ferland, "Genetic and hybrid algorithms for graph coloring," *Annals of Operations Research,* vol. 63, pp. 437-461, 1996.
[21] R. el Dorne and J. Hao, "A new genetic local search algorithm for graph coloring," 1998.
[22] P. Galinier and J.-K. Hao, "Hybrid evolutionary algorithms for graph coloring," *Journal of combinatorial optimization,* vol. 3, pp. 379-397, 1999.
[23] B. Oommen and E. Hansen, "The asymptotic optimality of discretized linear reward-inaction learning automata," *IEEE Transactions on Systems, Man and Cybernetics,* vol. 14, pp. 542-545, 1984.
[24] B. Johnoommen, "Absorbing and ergodic discretized two-action learning automata," *IEEE Transactions on Systems, Man and Cybernetics,* vol. 16, pp. 282-293, 1986.
[25] J. Akbari Torkestani and M. R. Meybodi, "Learning automata-based algorithms for finding minimum weakly connected dominating set in stochastic graphs," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 18, pp. 721-758, 2010.
[26] J. Akbari Torkestani, "An adaptive focused Web crawling algorithm based on learning automata," *Applied Intelligence,* vol. 37, pp. 586-601, 2012.
[27] J. Akbari Torkestani and M. R. Meybodi, "An efficient cluster-based CDMA/TDMA scheme for wireless mobile ad-hoc networks: A learning automata approach," *Journal of Network and Computer applications,* vol. 33, pp. 477-490, 2010.
[28] J. Akbari Torkestani and M. R. Meybodi, "Mobility-based multicast routing algorithm for wireless mobile Ad-hoc networks: A learning automata approach," *Computer Communications,* vol. 33, pp. 721-735, 2010.
[29] J. Akbari Torkestani and M. R. Meybodi, "An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata," *Computer Networks,* vol. 54, pp. 826-843, 2010.
[30] M. Jahanshahi, M. Dehghan, and M. Meybodi, "LAMR: learning automata based multicast routing protocol for multi-channel multi-radio wireless mesh networks," *Applied Intelligence,* vol. 38, pp. 58-77, 2013.
[31] M. R. Meybodi, "Learning automata and its application to priority assignment in a queueing system with unknown characteristics," Ph.D. thesis, Departement of Electrical Engineering and Computer Science, University of Oklahoma, Norman, Oklahoma, USA, 1983.
[32] M. L. Tsetlin, *Automaton theory and modeling of biological systems* vol. 102: Academic Press New York, 1973.
[33] A. Hashim, S. Amir, and P. Mars, "Application of learning automata to data compression," *Adaptive and learning systems,* pp. 229-234, 1986.
[34] B. Manjunath and R. Chellappa, "Stochastic learning networks for texture segmentation," in *Twenty-Second Asilomar Conference on Signals, Systems and Computers*, 1988, pp. 511-516.
[35] B. J. Oommen and E. Hansen, "List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations," *SIAM Journal on Computing,* vol. 16, pp. 705-716, 1987.
[36] B. J. Oommen and D. C. Y. Ma, "Deterministic learning automata solutions to the equipartitioning problem," *IEEE Transactions on Computers,* vol. 37, pp. 2-13, 1988.

[37] G. P. Frost, "Stochastic optimisation of vehicle suspension control systems via learning automata," Ph.D. Thesis, Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, 1998.
[38] M. Howell, G. Frost, T. Gordon, and Q. Wu, "Continuous action reinforcement learning applied to vehicle suspension control," *Mechatronics,* vol. 7, pp. 263-276, 1997.
[39] C. Unsal, P. Kachroo, and J. S. Bay, "Multiple stochastic learning automata for vehicle path control in an automated highway system," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 29, pp. 120-128, 1999.
[40] H. Beigy and M. R. Meybodi, "A learning automata-based algorithm for determination of the number of hidden units for three-layer neural networks," *International Journal of Systems Science,* vol. 40, pp. 101-118, 2009.
[41] M. R. Meybodi and H. Beigy, "Neural network engineering using learning automata: determining of desired size of three layer feed forward neural networks," *Journal of Faculty of Engineering,* vol. 34, pp. 1-26, 2001.
[42] B. J. Oommen and D. S. Croix, "String taxonomy using learning automata," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 27, pp. 354-365, 1997.
[43] A. G. Barto and M. I. Jordan, "Gradient following without back-propagation in layered networks," in *1st Int. Conference Neural Nets, San Diego,* 1987.
[44] M. Thathachar and V. V. Phansalkar, "Learning the global maximum with parameterized learning automata," *IEEE Transactions on Neural Networks,* vol. 6, pp. 398-406, 1995.
[45] P. Galinier and A. Hertz, "A survey of local search methods for graph coloring," *Computers & Operations Research,* vol. 33, pp. 2547-2562, 2006.
[46] M. Rezapoor M. and M. R. Meybodi, "A New Criteria for Creating Balance Between Local and Global Search in Memetic Algorithms," *Iranian Journal of Electrical and Computer Engineering (IJECE),* vol. 12, pp. 31-37, 2014.
[47] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning," *Operations research,* vol. 39, pp. 378-406, 1991.