Journal of Computational Science 24 (2018) 389–401

Contents lists available at ScienceDirect

Journal of Computational Science

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

A Michigan memetic algorithm for solving the vertex coloring problem

Mehdi Rezapoor Mirsaleh [a,*], Mohammad Reza Meybodi [b]

[a] Department of Computer Engineering and Information Technology, Payame Noor University (PNU), P.O. BOX 19395-3697, Tehran, Iran
[b] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 15 February 2017
Received in revised form 4 October 2017
Accepted 11 October 2017
Available online 17 October 2017

Keywords:
Learning automata (LA)
Memetic algorithm (MA)
Vertex coloring problem

ABSTRACT

Genetic algorithms (GAs) can be divided into two broad approaches, Michigan and Pittsburgh Approaches. In Pittsburg approach, one of the individuals (chromosomes) in the population becomes the solution of the problem being solved whereas in Michigan approach the whole population represents the solution. For example in the problem of vertex coloring, in Michigan approach, each chromosome encodes a color of a vertex and the set of all chromosomes in the population represents the solution which is a coloring for the graph whereas in the Pittsburgh approach each individual encodes a coloring for the whole graph. Combining a genetic algorithm (GA) with a local search produces a type of evolutionary algorithm (EA) known as a memetic algorithm (MA). MA uses the local search method to either accelerate the discovery of good solutions, for which evolution alone would take too long to discover, or to reach solutions that would otherwise be unreachable by evolution or a local search method alone. In this paper a Michigan memetic algorithm for solving vertex coloring problem is proposed. The initial population is a set of chromosomes each of which is associated to a vertex of the input graph. Each chromosome is a part of the solution and represents a color for its corresponding vertex. The proposed algorithm is a distributed algorithm in which each chromosome locally evolves by evolutionary operators and improves by a learning automata based local search. To evaluate the efficiency of the proposed algorithm several computer experimentations have been conducted. The results show the superiority of the proposed algorithm over existing algorithms in terms of running time and the required number of colors.

1. Introduction

Vertex coloring problem is a well-known combinatorial optimization problem in graph theory which is used in many applications such as timetabling, air traffic follow management, register allocation and scheduling [1]. Graph$G = (V, E)$ is given where V is the set of $|V| = n$ vertices and E is the set of $|E| = m$ edges. A k-coloring of graph G is a function $\Phi{:}V \rightarrow \Gamma$ where $\Gamma = \{1, 2, \ldots, k\}$ is the set of integers, each one represents one color. A coloring is valid if no two endpoints of each edge have the same colors. That is, for all $[u, v] \in E$ we have that $\Phi(u) \neq \Phi(v)$, otherwise coloring is invalid. 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, forms the conflict set V^C with size $|V^C|$. The minimum number of colors required for a valid coloring is called the chromatic number $\chi(G)$ and a graph G is said k-chromatic, if $\chi(G) = k$. We can consider a valid k-coloring as a partitioning of the set of vertices in k disjoint sets, named color classes and shown by $C = \{C_1, \ldots, C_k\}$. The k-coloring problem is formally NP-complete for general graph and chromatic number problem is NP-hard [2]. 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 valid k-coloring exists, and in the optimization version of the vertex coloring problem, the goal is to find the smallest number k by which the graph can be feasibly colored. 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 [3]. On the other hand, a near optimal coloring of graph can be used in many applications. Hence, different local search algorithms have been proposed for finding the near optimal solution for vertex coloring problem in the literature. Central to any local search algorithm is the neighborhood, which defines the set of solutions that can be reached from a current solution by one move, and the evaluation function used to rate solutions [4].

* Corresponding author.
E-mail addresses: mrezapoorm@aut.ac.ir (M. Rezapoor Mirsaleh), mmeybodi@aut.ac.ir (M.R. Meybodi).

https://doi.org/10.1016/j.jocs.2017.10.005
1877-7503/

In this paper a new Michigan memetic algorithm, called Michigan LA-MA, is proposed. The new algorithm is based on the memetic algorithm (MA) and learning automata (LA). Michigan LA-MA is composed of two parts, genetic section and memetic section. Evolution is performed in the genetic section and local search is performed in the memetic section. Genetic section consists of a population of chromosomes, each of which is associated to a vertex of graph. That is, each chromosome locally represents a color for its corresponding vertex and the whole population is used to coloring the graph. On the other hand, memetic section is equipped with a set of learning automata which are used to balance the process of exploration performed by evolution and the process of exploitation performed by local search. The proposed algorithm is a distributed algorithm in which each chromosome locally evolves and selects a color by evolutionary operators and improves its selection by a local search. In this paper, we compare the results of the proposed algorithm with the results of the best-known vertex coloring algorithms such as: TPA [5], AMACOL [6], ILS [7], CHECKCOL [8], GLS [9], CLAVCA [3] and CLAMACOL [10]. 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. Michigan approach and Pittsburg approach are briefly reviewed in Section 3. The theory of automata is described in Section 4. Michigan LA-MA is introduced in Section 5. The Michigan LA-MA based algorithm for vertex coloring problem is described in Section 6. Sections 7 is including of implementation considerations, simulation results, and comparison with other algorithms to highlight the contributions of the new algorithm. Finally, conclusions and future works are discussed in Section 8.

2. Related work

The wide variety approaches have been reported for solving the vertex coloring problem in the literature. These approaches classified as exact and approximation techniques. The first exact algorithm proposed in [11] to compute the chromatic number of graph. This algorithm finds the optimal solution with running time $O\left(2.4423^n\right)$. Also, an $O\left(2.40231^n\right)$ algorithm proposed to compute the chromatic number of graph in [12]. The faster algorithms reported in the literature when the number of coloring is fixed. An $O\left(1.3289^n\right)$ algorithm for k = 3 to solve k-coloring problem was provided in [13]. Also, an $O\left(1.7504^n\right)$ algorithm proposed in [14] for k = 4. Finally, two algorithms with running time $O\left(2.1020^n\right)$ and $O\left(2.3289^n\right)$ proposed for k = 5 [12] and for k = 6 [14] respectively. Due to the NP-hardness of the vertex coloring problem for large graphs, the 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 (TPA) is proposed in [5] for vertex coloring problem. The algorithm generates candidate solution 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 valid colorings is created. Then, in deterministic phase, each vertex is assigned to the color class 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 [7]. The algorithm generates candidate solution 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. 3) An acceptance criterion is used for selecting solutions to continue the search.

A priority-based local search algorithm, called CHECKCOL, is proposed in [8]. CHECKCOL decreases the running time of algorithm by avoiding unnecessary searches in large parts of the graph without making any progress in the solution. To do this, it introduce the checkpoint notion to force the algorithm to stop at certain steps and release all of its memory and then to start local search. In this algorithm, each vertex of the graph is dynamically assigned a priority. 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 [15] a tabu search algorithm was proposed to solve the vertex coloring problem. In this algorithm the vertices of the graph partition into several blocks and a different color is assigned to each block at each generation. This causes that the solutions be valid or invalid. In next step, to create a valid coloring, the set of neighbors generate for each vertex which is restricted by a tabu list and then a new solution is obtained by applying local search algorithm. Tabu list prevents premature convergence of algorithm.

A hybrid algorithm based on genetic algorithm and tabu search method is proposed in [16] to solve the 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 improved by selecting k-independent sets of vertices first and then applying hybrid algorithm on the remaining vertices.

A genetic local search algorithm is proposed In [17] to solve the 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 compared with the results obtained from best known methods. The comparison showed the superiority of the proposed algorithm.

An adaptive memory algorithm for k-coloring problem called AMACOL, was proposed in [6]. 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, 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 by using the obtained solutions.

A hybrid evolutionary algorithm called HEA, was proposed in [18] for vertex coloring problem. HEA uses the DSATUR construction heuristic to initialize the population of chromosomes. At each generation, new chromosomes are generated by recombining two parent chromosomes that are improved by local search method. The greedy partition crossover (GPX) is used as crossover operator in HEA. GPX generate a new chromosome by alternately selecting color classes of each parents. The new chromosome generated by GPX is then improved by tabu search method and replaced with the worse parent in current population.

A cellular learning automata (CLA) based memetic algorithm called CLAMACOL, was proposed for finding a near optimal solution for vertex coloring problem in [10]. CLAMACOL is based on irregular open CLA (IOCLA) which is an extension of irregular CLA (ICLA) in which the evolution of CLA is influenced by local and global environments. Similar to other IOCLA-based algorithms, in CLAMACOL, 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.

3. Michigan versus Pittsburgh approach

Genetic algorithms (GAs) can be divided into two broad approaches, Michigan and Pittsburgh Approaches. Pittsburg approach denotes any approach where each individual (chromosome) encodes the whole solution to the problem being solved and GA population converges to a single individual which is what usually happens in standard GAs. In Michigan approach the whole population represents the solution and each individual encodes a part of the solution to the problem. For example in the problem of vertex coloring, in Michigan approach, each chromosome encodes a color of a vertex and the set of all chromosomes in the population represents the solution which is a coloring for the graph whereas in the Pittsburgh approach each chromosome encodes a coloring for the whole graph. The Pittsburgh approach leads to syntactically longer individuals, which tends to make fitness computation more computationally expensive. Also, we may need to make some modifications to standard genetic operators to cope with relatively complex individuals. By contrast, in the Michigan approach the individuals are simpler and syntactically shorter. This tends to reduce the time taken to compute the fitness function and to simplify the design of genetic operators.

4. Theory of automata

A learning automaton [19] 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 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 [19,20]. Learning automata have been found efficiently operate in unknown environments where no priori information is available [21]. A learning automaton able to modify its behavior based solely on simple responses it receives from the environment. One of the strengths of the learning automata approach is that this learning model is supported by a powerful mathematical framework [22,23]. An attractive feature of learning automata approach is that it could be regarded as a simple unit from which complex systems such as hierarchies, games and feed forward networks could be constructed to handle the complicated learning problems [24,25]. Learning automaton 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 processes the response of environment and updates 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 $\langle \underline{\alpha}, \underline{\beta}, \underline{c} \rangle$ where $\underline{\alpha}=\{\alpha_1,\ldots,\alpha_r\}$is the finite set of the inputs, $\underline{\beta} = \{\beta_1,\ldots,\beta_m\}$ is the set of outputs that can be taken by the reinforcement signal, and $\underline{c}=\{c_1,\ldots,c_r\}$ is the set of the penalty probabilities, where each element c_i of $\underline{c}$ is associated with 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.

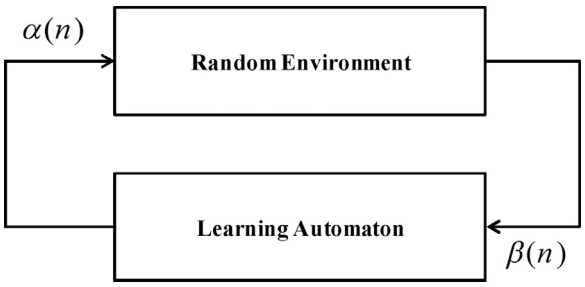

Fig. 1. The relationship between the learning automaton and random environment.

There are two main families of Learning automata [26]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $\langle \underline{\beta}, \underline{\alpha}, \underline{T} \rangle$, 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 $\alpha_i(k) \in \underline{\alpha}$ according to action probability vector $\underline{p}(k)$ at instant k. The action probability vector $\underline{p}(k)$ is updated by the learning algorithm given in Eq. (1), if the selected action $\alpha_i(k)$ 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.

$$P_j(n+1) = \begin{cases} P_j(n) + \alpha(1 - P_j(n)) j = i) \\ (1-\alpha)P_j(n) \forall_j, j \neq i \end{cases} \quad (1)$$

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

If $a = b$, the recurrence Eqs. (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a \gg b$ the given equations are called linear reward-€ 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 cognitive radio network [27], grid computing [28], adaptive Petri net [29–31], stochastic graphs [32,33], data mining [34], network sampling [35], peer-to-peer networks [36], social networks [37], recommender system [38], complex networks [39,40], image processing [41] and machine vision [42].

Several optimization methods such as CLA–EC which is obtained from the combination of cellular learning automata (CLA) and evolutionary computing (EC) [43] and CLA–DE which is obtained from the combination of CLA and a differential evolution algorithm (DE) [44] are reported in the literature. Both methods, with the aid of CLA, try to create a balance between exploration and exploitation.

In the CLA–EC model, each chromosome of the population is assigned to one cell of the CLA and each cell in the CLA is equipped with a set of learning automata, each of which corresponds to a gene of the assigned chromosome. The operation of the CLA–EC can be described as follows: At first step, all LAs in the CLA–EC choose their actions synchronously. The set of actions chosen by the learning automaton of a cell determines the chromosome for that cell. Based on a local rule, a vector of reinforcement signals is generated and given to the LAs residing in the cell. All LAs in the CLA–EC update their action probability vectors based on the reinforcement signal received using a learning algorithm. The process of action selection and updating the internal structures of LAs is repeated until a predetermined criterion is met.

In CLA–DE model each cell of the CLA is composed of two parts called the candidate solution (CS) and the solution model (SM). The SM represents a model for the desirability of different regions of the problem domain. The CS of a cell is the temporary, fittest solution found by the cell. The SM is a learning system including a set of LA in which each automaton is restricted to one dimension of the problem search space, and learns the promising regions of that dimension. The SM governs the evolution of its related CS. On the other hand, CSs also guide the learning process of their SMs through reinforcement signals provided for the SMs. Unlike DE or GA methods in which each entity represents a single point in the search space, each SM represents the entire search space, and thus prevents trapping at the local optima. The SMs interact with each other through a set of local rules. So they can impact the learning process of each other.

A new variant of Differential Evolution (DE), called ADE-Grid, is proposed in [45] which aims at controlling the diversity by adapting the mutation strategy, crossover rate and scale factor during the run. In ADE-Grid, learning automata are employed to determine the proper value of these parameters, and the suitable strategy for the construction of a mutant vector for each individual, adaptively. For this purpose, the individuals of the population are placed on distinct cells of a mesh grid. ADE-Grid is able to maintain the diversity among the individuals and encourage them to move toward several promising areas of the search space as well as the best found position.

An orthogonal evolutionary algorithm with LA (OELA) which consists of a new fitness function based on the decomposition of the objective space and a quantization orthogonal crossover (QOX) with LA is proposed in [46]. The goal of OELA is to try to find an optimal solution for each sub objective space. For this purpose, the fitness function which uses the weight vector and aggregate function is proposed to evenly decompose the objective space so that the diversity and the coverage of the obtained solutions to the true Pareto front (PF) can be well maintained. In addition, a QOX with learning automaton which improves the search efficiency by using the past information is also utilized in the algorithm, where the variables are purposely grouped by using the past information so as to quickly find optimal solutions.

In [47], two different classes of learning automata based differential evolution for adaptive selection of crossover probability and mutation strategy in differential evolution are proposed. In the first class, i.e. GLADE, genomes of the population use the same mutation strategy and crossover probability. In the second class, i.e. ILADE, each genome of the population acts in a separate manner and adjusts its own mutation strategy and crossover probability based on its success and failure.

In [40] a Michigan memetic algorithm; called MLAMA-Net; is proposed for solving the community detection problem. The MLAMA-Net algorithm is an evolutionary algorithm in which each chromosome represents a part of the solution and the whole population represents the solution. In the MLAMA-Net algorithm, the population of chromosomes is a network of chromosomes which is isomorphic to the input network. Each node has a chromosome and a learning automaton. The chromosome represents the community of corresponding node and saves the histories of exploration. The learning automaton represents a meme and saves the histories of the exploitation. The MLAMA-Net algorithm is a distributed algorithm in which each chromosome locally evolves by evolutionary operators and improves by a local search. By interacting with both the evolutionary operators and local search, this algorithm effectively detects the community structure in complex networks and solves the resolution limit problem of modularity optimization.

5. Michigan LA-MA

Michigan LA-MA is obtained from combination of learning automata (LA) and memetic algorithm (MA). This algorithm is composed of two parts, genetic section and memetic section. Evolution is performed in genetic section and local search is performed in memetic section. Memetic section is equipped with a set of learning automata which are used to balance the process of exploration performed by evolution and the process of exploitation performed by local search. In what follows genetic and memetic sections are described.

5.1. Genetic section

The genetic section is represented by a triple $\langle P(t), A, GF\rangle$ where

- $P(t) = \{CR^1, CR^2, \ldots, CR^N\}$ is the population of chromosomes at generation t.
- A is a binary matrix where indicates the adjacency of chromosomes. If chromosome CR^i is adjacent to chromosome CR^j then $A[i,j] = 1$, otherwise $A[i,j] = 0$.
- GF is fitness function which is used to evaluate the fitness of a chromosome based on its genotype and genotypes of its adjacent chromosomes. The fitness of chromosome CR^i at generation t; which is referred to as genetic fitness; is denoted by $GF^i(t)$. The fitness function is a maximizing function and the genetic fitness is rescaled to a number in [0, 1].

In proposed algorithm we use the Michigan approach for chromosome representation. Each chromosome as a part of solution, is composed of n genes and each gene can take one of m possible values from set $V = \{v_1, v_2, \ldots, v_m\}$. Chromosome i is denoted by $CR^i = \left[CR^i_1, CR^i_2, \ldots, CR^i_n\right]$ where CR^i_k is the value of k_{th} gene, $1 \le i \le N$, $1 \le k \le n$ and $CR^i_k \in V$. Initial chromosomes are created randomly and then changed by mutation operator with rate r_m at next generations.

5.2. Memetic section

The memetic section is represented by a triple $\langle M(t), LS, MF\rangle$ where

- $M(t) = \{M^1(t), M^2(t), \ldots, M^N(t)\}$ is the population of memes at generation t and $M^i(t)$ is a meme which save the effect (history) of the local search on the chromosome CR^i at generation t.
- LS is a local search method where is applied on a chromosome and its adjacent chromosomes.
- MF is a function which is used to evaluate the effect (history) of local search method on a chromosome. The effect (history) of the local search method on chromosome CR^i at generation t; which is referred to as memetic fitness; is denoted by $MF^i(t)$.

For each chromosome, there is a meme which saves the effect (history) of local search method on its corresponding chromosome. Each meme is composed of n learning automata each of which corresponds to a gene of its corresponding chromosome. Specifically, the j_{th} automaton in meme $M^i(t)$ corresponds to the j_{th} gene of the chromosome CR^i. Each learning automaton has m actions corresponding to m possible values that each gene can take from set V. The effect (history) of the local search method on chromosome CR^i at generation t is represented by the action probability vectors of learning automata in the meme $M^i(t)$ as given by Eq. (3).

$$M^i(t) = \left[M^i_1(t), M^i_2(t), \ldots, M^i_n(t)\right], \quad 1 \le i \le N \tag{3}$$

where

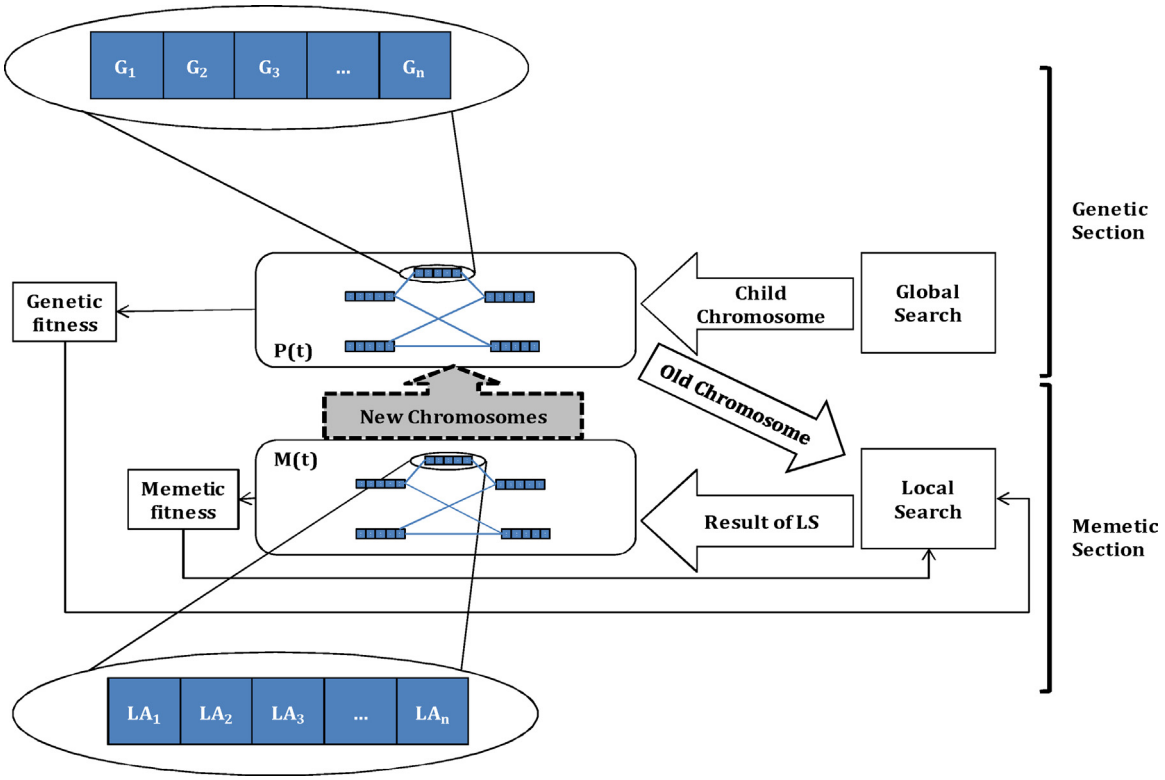

Fig. 2. The relationship between the genetic section and memetic section in Michigan LA-MA.

$M_j^i(t) = \left[M_{j1}^i(t), M_{j2}^i(t), \ldots, M_{jm}^i(t)\right]'$, $1 \le i \le N$, $1 \le j \le n$ and $\forall i, j, \sum_{k=1}^{m} M_{jk}^i(t) = 1$.

$M_{jk}^i(t)$ denotes the probability that action k of j_{th} leaning automaton in the meme $M^i(t)$ (corresponding to the value v_k of gene j of chromosome CR^i) is selected in the exploitation process. In other words, $M_{jk}^i(t)$ is the probability that value v_k is selected by local search method for gene j. $M_{jk}^i(0)$ where$1 \le i \le N, 1 \le j \le nand1 \le k \le m$ is initially set to $1/m$. After the local search has been applied on chromosome CR^i, the action probability vectors of the meme $M^i(t)$ are updated according to a learning algorithm based on the reinforcement signal received from the genetic section. $MF^i(t) = \prod_{j=1}^{n} M_{jk}^i(t)$, where k is the action of automaton j in the meme $M^i(t)$ which corresponds to the value of gene j in chromosome CR^i (that is $v_k = CR_j^i$), is the probability of locating chromosome CR^i in the basin of attraction containing the best solution found so far by local search method at generation t. The $MF^i(t)$ is only depended on the action probability vectors of learning automata in the meme $M^i(t)$ and the value of gene j in chromosome CR^i. Memetic fitness of a chromosome changes when the action probability vectors of learning automata in its corresponding meme is updated. Updating is performed on the basis of the result of applying the local search method on a chromosome. If the values of gene j of chromosome i (CR_j^i) are the same before and after applying local search method on chromosome CR^i, the action of j_{th} learning automaton of the meme $M^i(t)$ which corresponds to the value CR_j^i is rewarded and penalized, otherwise. It is worth noting that, local search changes only the action probability vectors of the meme not values of the genes. That is, local search only changes the memetic fitness not the genetic fitness. Local search method is applied on chromosome CR^i based on the genetic information (genotype and genetic fitness) and memetic information (action probability vectors and memetic fitness) of chromosome CR^i and the genetic and memetic information of its adjacent chromosomes. The adjacent chromosomes of chromosome CR^i is a finite subset of $P(t)$. For example, in vertex graph coloring problem, local search is applied on chromosome CR^i associated to vertex i based on the information of chromosome CR^i and the information of chromosomes associated to neighboring vertexes of vertex i. The relationship between genetic section and memetic section for Michigan LA-MA is shown in Fig. 2.

The proposed algorithm is a fully distributed algorithm in which each chromosome locally evolves based on its adjacent chromosomes and independent of the other chromosomes. The operation of the proposed algorithm can be described as follow. Initial population is created randomly and the probability of selecting an action for all learning automata is set to $1/m$. The proposed algorithm is progressed in a number of generations as long as the termination criteria are not satisfied. First, mutation operator is applied on chromosome CR^i with rate r_m. In the next step, local search method is applied to chromosome CR^i based on a criterion described in next section, and then the action probability vectors of the meme $M^i(t)$ (the history) is updated according to a learning algorithm based on the reinforcement signal received from the genetic section. In this step chromosome CR^i and its genetic fitness will not change. Fig. 3 demonstrates the pseudo code for Michigan LA-MA.

The steps of procedure Michigan LA-MA are described below.

Step 1: Initialize the action probability vectors of the meme $M^i(t)$.

Step 2: Create the chromosome CR^i randomly.

Step 3: Until the termination condition is satisfied, Michigan LA-MA is running in the while loop.

Step 4: Mutation operator is applied on chromosome CR^i with rate r_m.

Step 5: Calculate the genetic fitness for chromosome CR^i.

Step 6: Calculate the memetic fitness for chromosome CR^i.

Procedure: Michigan LA-MA
Begin
$t=0$;
1: Initialize the probability vectors of the meme $M^i(t) = \left[M_1^i(t), M_2^i(t), \ldots, M_n^i(t)\right]$;
2: Initialize chromosome CR^i;
3: **While**(termination criteria are not satisfied)
$t=t+1$;
If(random() < r_m)
4: Apply Mutation operator on chromosome CR^i;
End If
5: Calculate $GF^i(t)$;
6: Calculate $MF^i(t)$;
7: **If** (applying local search on chromosome CR^i is beneficial)
8: Apply the local search method on CR^i and update the action probability vectors of meme $M^i(t)$ according to the learning algorithm;
End If
9: Generate a new chromosome based on the action probability vectors of the meme ($M^i(t)$)
10: **If** (Genetic fitness of new chromosome is higher than genetic fitness of chromosome CR^i)
11: Replace chromosome CR^i with new generated chromosome;
End If
End While
End Procedure Michigan LA-MA

Fig. 3. Pseudo code for Michigan LA-MA.

Step 7: If the local search leads to greater benefits than global search then the local search is applied to chromosome CR^i. We shown in [48], if inequality $\left(1 - MF^i(t)\right)^2 / MF^i(t) \geq \left(1 - GF^i(t)\right) / GF^i(t)$ is satisfied then applying local search method on chromosome CR^i is beneficial.

Step 8: Update the action probability vectors of the meme $M^i(t)$ according to a learning algorithm based on the reinforcement signal received from the genetic section after applying the local search method on chromosome CR^i. If the values of a gene (for example j_{th} gene) of chromosome CR^i are the same before and after applying the local search method on the chromosome CR^i, the corresponding action of j_{th} learning automaton of the meme $M^i(t)$ is rewarded and penalized, otherwise.

Step 9: Every learning automaton of the meme $M^i(t)$ randomly chooses one of its actions (the value of gene) and as a result a new chromosome is generated.

Steps 10, 11: If the genetic fitness of new chromosome be higher than genetic fitness of chromosome CR^i, the newly generated chromosome replaces chromosome CR^i.

6. The michigan LA-MA based algorithm for vertex coloring problem

In this section, we propose a Michigan LA-MA based algorithm called MLAMACOL for solving the vertex coloring problem.

6.1. Genetic section for vertex coloring problem

Genetic section includes a population of chromosomes with size N = n where n is the number of vertices of given graph. For this purpose, each vertex v_i of graph is associated with the chromosome CR^i which represents the color of vertex v_i by an integer number. Let $\underline{a_i} = \{a_i^1, a_i^2, \ldots, a_i^{d_i+1}\}$ be the set of values (colors) that can be represented by chromosome CR^i (the set of colors by which the vertex v_i can be colored) where d_i is the degree of vertex v_i. It has been shown in [49] that an arbitrary graph can be colored with at most $D+1$ colors, where D denotes the graph degree. Therefore, it can be concluded that vertex v_i and its neighboring vertices can be colored with at most d_i+1 colors in the worst case. That is why the set of values (colors) that can be represented by chromosome CR^i (corresponding to color-set of vertex v_i) is composed of d_i+1 values (or colors). Initial chromosome CR^i is created randomly by selecting a random integer number from color-set $\underline{a_i} = \{a_i^1, a_i^2, \ldots, a_i^{d_i+1}\}$. At the beginning of each generation, mutation operator is applied on chromosome CR^i with rate r_m in which the value of chromosome CR^i is exchanged by other value of its color-set $\underline{a_i} = \{a_i^1, a_i^2, \ldots, a_i^{d_i+1}\}$.

6.2. Memetic section for vertex coloring problem

Memetic section in MLAMACOL consists of n memes where n is the number of vertices of given graph. The effect (history) of the local search method on chromosome CR^i at generation t is represented by meme i ($M^i(t)$) where is equipped with a learning automaton in which $\underline{a_i} = \{a_i^1, a_i^2, \ldots, a_i^{d_i+1}\}$ (defined in genetic section) is the set of actions (colors) that can be taken by learning automaton (the set of colors by which the vertex v_i can be colored). Updating of the action probability vector of associated learning automaton to meme $M^i(t)$ (Memetic fitness of chromosome CR^i) is performed on the basis of the result of applying the local search method on the chromosome CR^i as described in next paragraph.

6.3. Local search for vertex coloring problem

In vertex coloring problem the local search method is applied on chromosome CR^i based on the genetic and memetic information of chromosome CR^i and the genetic and memetic information of chromosomes associated to neighbors of vertex i. Let α_i be the color of vertex v_i which is represented by chromosome CR^i, $N_i = \{u | [u, v_i] \in E\}$ be the set of neighbors of vertex v_i, $N_i(c) = \left\{j \in N_i : \alpha_j = c\right\}$ be the set of neighbors of vertex v_i with color c and $|N_i(c)|$ be the number of neighbors of vertex v_i with color c. At generation t, the action α_i of learning automaton associated to meme $M^i(t)$ is rewarded, if color of vertex v_i does not conflict with the colors represented by its neighboring chromosomes or it has the highest priority amongst its neighbors. Otherwise, it will be penalized.

The priority of vertex v_i can be described as follows:

$$priority(v_i) = \begin{cases} 1 & if(d_i > d_j) \forall j \in N_i(\alpha_i) \\ 1 & if(d_i = d_j) and \left(MF^i(t) > MF^j(t)\right) \forall j \in N_i(\alpha_i) \\ 0 & otherwise \end{cases} \quad (4)$$

where $MF^i(t)$ and $MF^j(t)$ are the memetic fitness of chromosomes CR^i and CR^j at generation t, respectively. In the last step, learning automaton associated to meme $M^i(t)$ randomly chooses one

Table 1
The characteristics of DSJ benchmark graphs.

Graph Name	Number of vertices	Number of edges	Density
DSJC125.1	125	736	0.09
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

of its actions (the value of gene) and as a result a new chromosome is generated. If the genetic fitness of new chromosome be higher than genetic fitness of chromosome CR^i, the newly generated chromosome replaces the chromosome CR^i. The genetic fitness of chromosome CR^i is calculated based on the number the neighbors of vertex i which are in conflict with vertex i as follows:

$$GF^i(t) = \frac{1}{1 + |N_i(\alpha_i)|} \tag{5}$$

where α_i is the selected color for vertex v_i and $|N_i(\alpha_i)|$ is the number of neighbors of vertex v_i with color α_i. $GF^i(t)$ counts the number of neighbors of vertex v_i that have same color with color of vertex v_i. The coloring process continues (in parallel) for each vertex v_i, if the probability of an action (color) of meme i exceeds a pre-specified threshold, e.g., π_i.

7. Experimental result

In this section several experiments are described to show the efficiency of proposed algorithm. The results of proposed algorithm are compared with the results of the best-known vertex coloring algorithms such as: TPA [5], AMACOL [6], ILS [7], CHECKCOL [8], GLS [9], CLAVCA [3] and CLAMACOL [10]. The obtained results show the superiority of the new algorithm over the other algorithms in terms of running time of algorithm and number of colors required for coloring the benchmarks. The experiments are implemented by MATLAB 2009a running on a PC with a 2.83 GHz processor and 4 GB memory. For all experiments the mutation rate is 0.05, the output of environment can take only one of two values 0 or 1 (P-model environment), the action probability vectors of learning automata are updated according to L_{R-I} learning algorithm with a = 0.5 and algorithm is terminated when the probability of chosen at least an action (color) of learning automaton in each meme was 0.95 or greater. It also should be noted that these parameters are obtained based on empirical experiment. We used a database of hard-to-color benchmarks like DSJ graphs [50], Leighton graphs [51] and WAP graphs [5]. The characteristics of the used benchmark graphs are given in Tables 1–6 show the comparison of obtained results for different algorithms. In these tables, for all algorithms, the first column (color) includes the number of colors required for coloring the graph, the second column (FE) includes the number of fitness evaluations of each algorithm and the third column (time) includes the running time of each algorithm in seconds. Also, to find the impact of learning automata on MLAMACOL performance, two experiments have been conducted.

7.1. Experiment 1

The first class of the benchmark graphs on which the proposed algorithm is tested is DSJ graphs. This class consists of the graphs with a variable number vertices n, where each of the n(n-1)/2 pos-

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

Graph Name	TPA			AMACOL			ILS			CHECKCOL			GLS			CLAVCA			CLAMACOL			MLAMACOL		
	color	FE	time	color	FE	time	color	FE	time	color	FE	time	color	FE	time	Color	FE	time	Color	FE	time	color	FE	time
DSJC125.1	5	8063	0	5	8501	0	5	8288	0	5	7750	0	5	8126	0	5	7513	0	5	7154	0	**5**	**6269**	5E-04
DSJC125.5	19	2489739	289	17	1138034	125	17	22101	2	17	913467	110	18	4334	0	17	100167	12	17	65056	8.06	**17**	**18236**	2.23
DSJC125.9	44	12995	5	44	143245	57	44	1214	0	44	10220	4	**44**	**1191**	0	44	42929	19	44	26830	12.4	44	14278	7.28
DSJC250.1	8	70548	10	8	88555	12	**8**	**3453**	0	8	184063	28	9	3386	0	8	46953	7	9	28797	4.35	8	14821	2.34
DSJC250.5	30	8895989	3282	28	184313	64	28	96110	34	28	1452311	557	30	4097	1	28	69443	27	28	45169	18.4	**28**	**15472**	6.85
DSJC250.9	72	277837	155	72	4906705	2604	**72**	**11938**	6	72	313435	182	73	11705	6	73	54107	32	72	35289	21.9	72	12671	8.63
DSJC500.1	**12**	**1250**	0	12	25042	9	13	1285	0	12	10814	4	13	1260	0	12	47754	20	13	24590	10.6	12	15380	7.42
DSJC500.5	48	50190	124	48	138783	326	50	44132	106	48	693431	1789	52	33114	81	48	15964	42	48	10395	28.7	**48**	**9137**	28.68
DSJC500.9	127	629381	1268	126	894855	1710	128	42076	82	126	975546	2045	130	77262	154	126	32593	70	126	31508	71.4	**126**	**22036**	56.68
DSJC1000.1	21	25532	28	**20**	**915774**	969	21	5986	6	21	122708	142	22	1354	1	21	32972	39	22	14915	18.3	21	19386	27.36
DSJC1000.5	84	549756	2386	84	2243237	9235	91	71867	303	84	1555646	7025	93	126886	546	84	18996	88	85	17762	86.8	**84**	**10152**	56.24
DSJC1000.9	226	580933	3422	224	883683	4937	228	391792	2245	226	2046897	12545	234	277405	1621	224	18900	119	**224**	**11496**	76.1	224	11534	86.98

*Best results are highlighted in bold.

Table 3
The characteristics of Leighton benchmark graphs.

Graph Name	Number of vertices	Number of edges	Density
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_25c	450	17343	0.17
Le_25d	450	17425	0.17

sible edges is generated independently at random with probability p. DSJ graphs are denoted as DSJCn.p where n ∈ {125, 250, 500, 1000} and p ∈ {0.1, 0.5, 0.9}. The characteristics of DSJ benchmark graphs are given in Table 1.

Table 2 shows the results of simulation experiments conducted on DSJ benchmark graphs. Comparing the results reported in Table 2, we find that, in 8 cases, GLS outperforms the others in terms of running time. It can be seen that AMACOL and CHECKCOL always select the smallest color-set for coloring the graphs, while their running time are the worst. It also can be seen that GLS uses the most number of colors, and AMACOL uses the smallest number of colors for coloring the graph. Comparing the MLAMACOL with GLS, it can be seen that the running time of the MLAMACOL is as close to GLS as possible, while the size of the color-set, in the MLAMACOL, is smaller as compared with GLS. On the other hand, comparing the result of proposed algorithm and AMACOL, we find that, in 10 cases, the color-sets chosen by MLAMACOL are as small as those AMACOL, while the running time of the MLAMACOL is considerably shorter than AMACOL.

7.2. Experiment 2

Leighton benchmark graphs are the second class by which the proposed algorithm is tested. 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 le450_kx, where 450 is the number of vertices, 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 Tables 3 and 4 shows the results of the conducted experiments on Leighton benchmark graphs. Comparing the results reported in Table 4, we observe that, in 4 cases, ILS outperforms the other algorithms in terms of running time. It also can be seen CHECKCOL, in all cases, select the smallest color-set for coloring the benchmark graphs, while its running time is the worst. Comparing the result of MLAMACOL with the minimum number of required colors for coloring benchmark graphs, we observe that the size of color-set conducted by the proposed algorithm is equal to the minimum number of required colors for coloring benchmark graphs. It also can be seen that, the proposed algorithm is significantly outperforms CHECKCOL in terms of running time.

7.3. Experiment 3

The third class of benchmark graphs by which the proposed algorithm is tested includes WAP graphs which arise in the design of transparent optical networks. In WAP graphs, each vertex corresponds to a light path in the network and edges correspond to intersecting paths. These graphs are denoted by WAP0ma, where m ∈ {1,2,. . .,8}. They have between 905 and 5231 vertices and

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

Graph Name	TPA			AMACOL			ILS			CHECKCOL			GLS			CLAVCA			CLAMACOL			MLAMACOL		
	color	FE	time	color	FE	time	color	FE	time	color	FE	time	color	FE	time	Color	FE	time	Color	FE	time	color	FE	time
Le_15a	15	4658513	1444	15	1174873	345	**15**	**1653**	0	15	6652123	2145	15	8125	2	15	70618	23	15	36651	12.4	15	21869	8.23
Le_15b	15	5338988	1655	15	1175681	345	15	1677	0	15	8546528	2756	**15**	**1625**	0	15	55593	18	15	36594	12.3	15	32891	12.6
Le_15c	15	177375	82	**15**	**5668**	2	15	42965	19	15	9372812	4534	15	14083	6	15	61102	30	15	20634	10.4	15	36323	21.3
Le_15d	15	74175	34	**15**	**10202**	4	15	45168	20	15	9459626	4576	15	18417	8	15	65108	32	15	26923	13.7	15	38327	22.5
Le_25c	26	95675	44	26	211965	93	26	5508	2	**25**	**7187993**	3477	26	40083	18	26	61102	30	27	16245	8.05	26	15281	8.65
Le_25d	26	48375	22	26	23804	10	26	3305	1	25	9352142	4524	26	5417	2	25	93155	46	26	32775	16.8	**25**	**26486**	15.4

*Best results are highlighted in bold.

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
WAP06a	947	43571	0.1
WAP07a	1809	103368	0.06
WAP08a	1870	104176	0.06

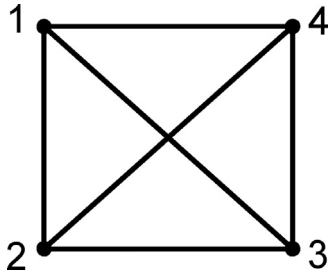

Fig. 4. Sample graph G, to study the search progress of the proposed algorithm.

all instances have a clique of size 40. The characteristics of WAP benchmark graphs are given in Table 5.

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, ILS outperforms others in terms of the running time. The obtained results reported in Table 6 show that, in all 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 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 number of fitness evaluations of each algorithm. From the results reported in Error! Reference source not found., we observe that, the both average rank of MLAMACOL (2.12) and median rank of MLAMACOL (1) are smaller than the average rank and median rank of other algorithms.

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 MLAMACOL algorithm and the rank of all other algorithms (TPA, AMACOL, ILS, CHECKOL, GLS, CLAVCA and CLAMACOL) are statistically significant (p-value <0.05).

7.4. Experiment 4

Experiments 4 and 5 aimed to find the impact of learning automata on MLAMACOL performance. For this purpose we compared the proposed algorithm (MLAMACOL) with the proposed algorithm in which the learning automata residing in each meme are replaced by pure chance automata (MLAMACOL-PC), in terms of speed of convergence. In pure chance automaton the actions are always chosen with equal probabilities. The goal of experiment 4 is to study the search progress of the proposed algorithm. Radar charts of Figs. 5 and 6 show the search space and search progress of the MLAMACOL algorithm and MLAMACOL algorithm with pure chance algorithm for sample graph G shown in Fig. 4, respectively.

Figs. 5 and 6 show the all $4\hat{4}=256$ possible solutions for coloring the given graph G and the number of vertexes which are in

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

Graph Name	TPA			AMACOL			ILS			CHECKCOL			GLS			CLAVCA			CLAMACOL			MLAMACOL		
	color	FE	time	color	FE	time	color	FE	time	color	FE	time	color	FE	time	Color	FE	time	Color	FE	time	color	FE	time
Wap01a	42	92956	245	45	138000	345	44	780	2	44	207153	568	42	21029	55	42	5647	16	42	3370	10.1	**42**	**2447**	8.32
Wap02a	41	226872	1618	44	118557	802	43	36177	251	43	65504	486	41	22609	160	41	3652	28	41	1779	14.4	**41**	**1341**	12.3
Wap03a	44	2333	17	53	35447	245	46	51488	365	46	90889	689	44	108149	782	43	6511	51	44	3596	29.7	**43**	**1956**	18.4
Wap04a	43	14589	95	48	7286	45	44	76403	484	44	3395	23	43	129071	834	43	6714	47	43	4431	32.7	**43**	**1962**	16.5
Wap06a	41	320657	348	44	529429	545	42	947	1	42	22143	25	41	7429	8	40	2571	3	42	1515	1.86	**40**	**2186**	3.06
Wap07a	42	232630	541	45	40347	89	44	442	1	44	75227	182	42	93167	215	40	5600	14	44	4784	12.6	**40**	**2340**	7.02
Wap08a	42	80625	200	45	189550	446	43	23205	56	44	8525	22	42	16656	41	42	9750	26	44	5226	14.7	**42**	**4275**	13.7

*Best results are highlighted in bold.

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

Graph Name	TPA	AMACOL	ILS	CHECKCOL	GLS	CLAVCA	CLAMACOL	MLAMACOL
DSJC125.1	5	8	7	4	6	3	2	**1**
DSJC125.5	8	6	2	5	7	4	3	**1**
DSJC125.9	4	8	2	3	**1**	7	6	5
DSJC250.1	3	4	1	5	6	2	8	7
DSJC250.5	8	5	4	6	7	3	2	**1**
DSJC250.9	5	6	1	4	7	8	3	2
DSJC500.1	**1**	4	3	3	2	5	8	2
DSJC500.5	4	5	7	6	8	3	2	**1**
DSJC500.9	6	4	7	5	8	3	2	**1**
DSJC1000.1	4	1	2	6	7	5	8	3
DSJC1000.5	3	5	7	4	8	2	6	**1**
DSJC1000.9	5	4	7	6	8	3	**1**	2
Le_15a	7	6	**1**	8	2	5	4	3
Le_15b	7	6	2	8	1	5	4	3
Le_15c	7	**1**	5	8	2	6	3	4
Le_15d	7	**1**	4	8	2	6	3	5
Le_25c	6	7	2	**1**	4	5	8	3
Le_25d	8	6	4	3	5	2	7	**1**
Wap01a	5	8	6	7	4	3	2	**1**
Wap02a	5	8	6	7	4	3	2	**1**
Wap03a	3	8	6	7	5	2	4	**1**
Wap04a	4	8	7	6	5	3	2	**1**
Wap06a	4	8	5	7	3	2	6	**1**
Wap07a	4	8	5	7	3	2	6	**1**
Wap08a	4	8	5	6	3	2	6	**1**
Average rank	5.08	5.72	4.32	5.6	4.72	3.76	4.32	**2.12**
Median rank	5	6	5	6	5	3	4	**1**

*Best results are highlighted in bold.

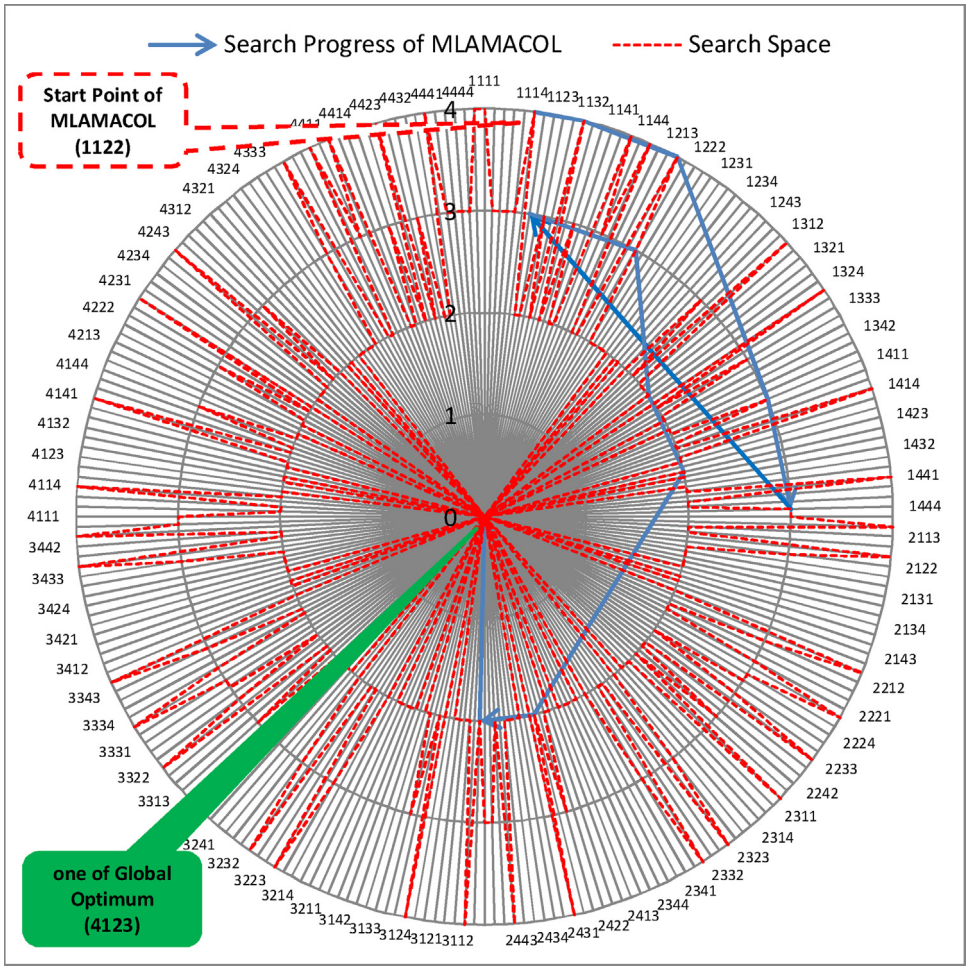

Fig. 5. Search space and search progress of MLAMACOL for sample graph.

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

MLAMACOL vs.	TPA	AMACOL	ILS	CHECKCOL	GLS	CLAVCA	CLAMACOL
p-values	3.88E-06	6.16E-06	4.61E-04	1.06E-06	1.04E-04	4.13E-04	2.65E-04

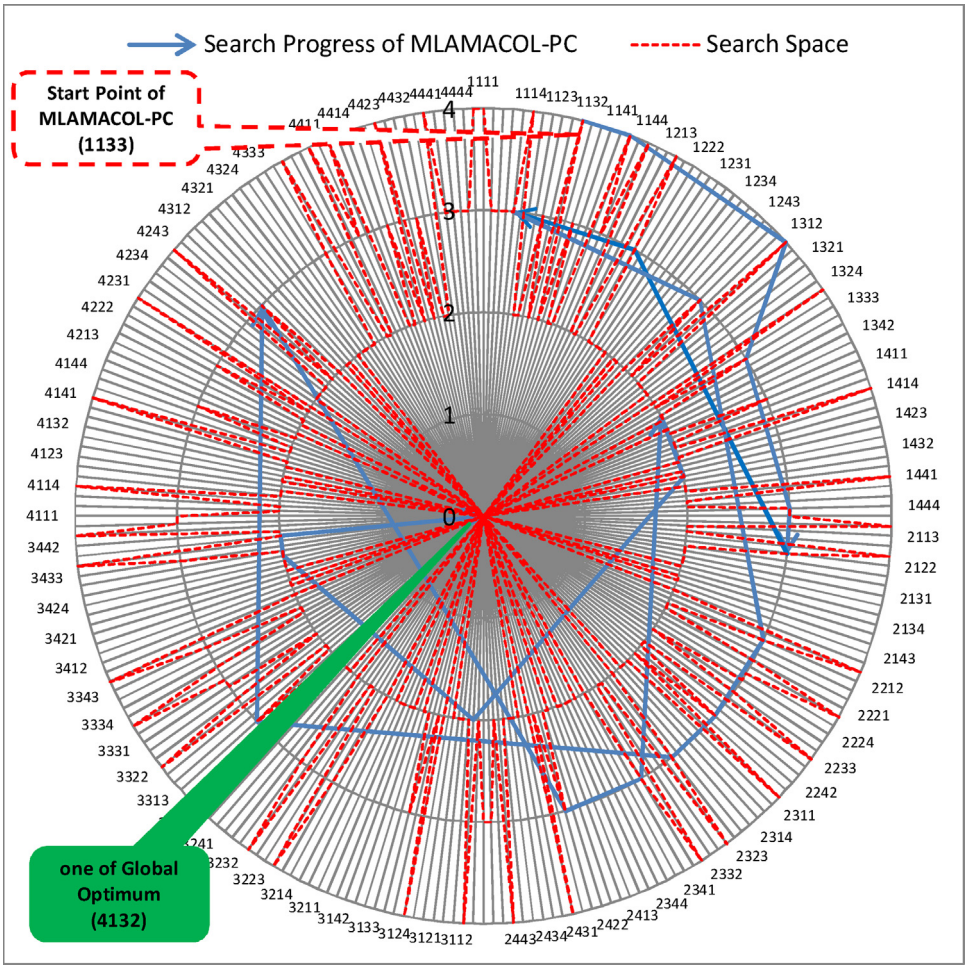

Fig. 6. Search space and search progress of MLAMACOL-PC for sample graph.

conflict ($|V^C|$)in each solution along a separate axis that starts in the center of the chart and ends on the outer ring. In Fig. 5, for the search progress of MLAMACOL algorithm, the initial population contains four chromosomes $P(0) = \{CR^1 = 1, CR^2 = 1, CR^3 = 2, CR^4 = 2\}$. The initial population $P(0)$ represents the solution 1122 for coloring the given graph G in which color 1 is selected for vertexes v_1 and v_2 and color 2 is selected for vertexes v_3 and v_4. As shown by thick arrows, solutions 1133, 1221, 1411, 1444, 1123, 1222, 1323, 1424, 2423, 3112 and 4123 have been accessed by MLAMACOL during a typical runs. In Fig. 6, the search progress of proposed algorithm with pure chance automaton has been shown. In this figure the initial population contains solution 1133 but an intricate sequence of solutions have been accessed to reach an optimal solution (4132) for coloring the given graph G during a typical runs.

7.5. Experiment 5

The goal of this experiment is to compare the entropy of action probability vector of memes associated to vertexes of graph G shown in Fig. 4 for MLAMACOL algorithm and MLAMACOL algorithm with pure chance automaton (MLAMACOL-PC). Entropy is a significant concept in the thermodynamics, representing the degree of disorder in a thermodynamic system [52]. Shannon has introduced this concept into the information theory, by the name of "information entropy". Entropy, in its basic, indicates a measure of uncertainty rather than a measure of information. More specifically, the information entropy is a case of the entropy of random variables defined as follows:

$$H(X) = -\sum_{X \in \chi} P(X) \log(P(X)) \tag{6}$$

where, X represents a random variable with set of values χ and probability mass function P(X).

The entropy of action probability vector of meme i ($M^i(t)$) associated to vertex v_i is defined as follows:

$$H\left(M^i(t)\right) = -\sum_{j=1}^{d_i+1} M_j^i(t) \log\left(M_j^i(t)\right) \tag{7}$$

The entropy of $M^i(t)$ has its maximum value at generation 0; $max\left(H\left(M^i(t)\right)\right) = H\left(M^i(0)\right) = log(d_i + 1)$; when all actions have

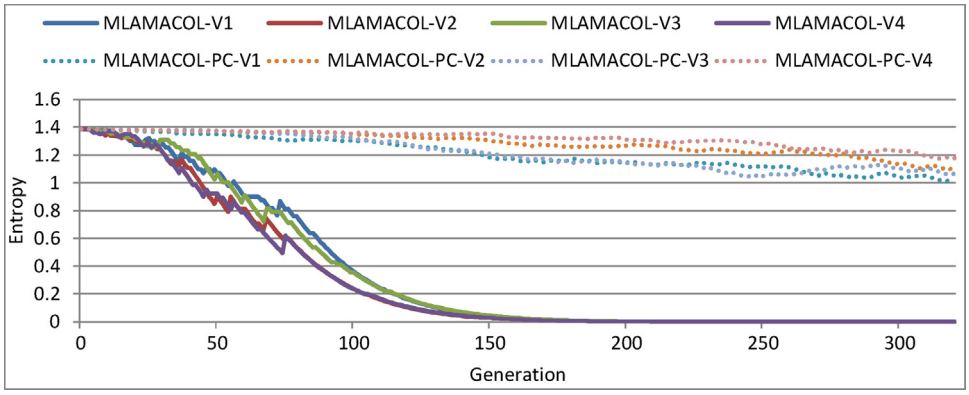

Fig. 7. Comparison of the entropy of the action probability vector of memes associated to vertexes of graph G at different generations. MLAMACOL-Vi is the entropy of the action probability vector of meme associated to vertex Vi in MLAMACOL algorithm and MLAMACOL-PC-Vi is the entropy of the action probability vector of meme associated to vertex Vi in MLAMACOL algorithm with pure chance automaton.

equal probabilities and has its minimum value; $min\left(H\left(M^i(t)\right)\right) = 0$; when the action probability vector is a unit vector. Fig. 7 shows the comparison of the entropy of the action probability vector of memes associated to vertexes of graph G shown in Fig. 4 for MLAMACOL algorithm and MLAMACOL-PC algorithm. The entropy of $M^i(t)$ represents the uncertainty of the decision by local search in vertex v_i. The larger entropy causes the more uncertain the decision maker. In reinforcement learning, the process of optimization increases the probability values of selecting optimal solution, i.e. the initial random solution will converge to a specific optimal solution. Therefore, the entropy can represent the degree of convergence in the learning process. Fig. 7 shows that in initial generations, entropy lines are not smooth because of no convergence in action probability vector of memes associated to vertexes of graph G. In last generations, the action probability vector converges to optimal action. This causes the probability of optimal action approaches to near 1 and the probability of other actions approaches to near 0. This means that, entropy lines converge to 0 in last generations. In other words, Fig. 7 shows the role of learning automata in convergence the finding optimal solution for coloring graph G shown in Fig. 4.

8. Conclusion

A new Michigan learning automata based algorithm called MLAMACOL is proposed for finding a near optimal solution for vertex coloring problem in this paper. The proposed algorithm is based on the LA-MA, a combination of learning automata and memetic algorithm, in which the learning automata has the role of local search and evolution is performed in memetic algorithm. Unlike the Pittsburg evolutionary algorithms in which each chromosome represents a full solution to the problem; the proposed algorithm represents the chromosomes based on the Michigan approach. In the proposed algorithm, each chromosome which is associated to a vertex of graph represents part of the solution. That is, each chromosome locally represents a color for its corresponding vertex and the whole of population is used to coloring of graph. The proposed algorithm is a distributed algorithm in which each chromosome locally evolves by evolutionary operators and improves by a local search. To show the efficiency of proposed algorithm, several computer simulations were conducted on hard-to-color benchmark graphs. The obtained results showed that the proposed algorithm outperforms the well-known algorithms both in terms of the required number of colors and the running time of algorithm.

References

[1] T.R. Jensen, B. Toft, Graph Coloring Problems, Wiley, 2011.
[2] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, 2017, pp. 85–103.
[3] J. Akbari Torkestani, M.R. Meybodi, A cellular learning automata-based algorithm for solving the vertex coloring problem, Expert Syst. Appl. 38 (2011) 9237–9247.
[4] M. Chiarandini, I. Dumitrescu, T. Stützle, Local Search for the Colouring Graph Problem. A Computational Study, 10, TU, Darmstadt, FB, Intellektik, FG Informatik Hochschulstr., Darmstadt, German, 2003, pp. 64289.
[5] M. Caramia, P. Dell'Olmo, Embedding a novel objective function in a two-phased local search for robust vertex coloring, Eur. J. Oper. Res. 189 (2008) 1358–1380.
[6] P. Galinier, A. Hertz, N. Zufferey, An adaptive memory algorithm for the k-coloring problem, Discrete Appl. Math. 156 (2008) 267–279.
[7] H.R. Lourenço, O. Martin, T. Stutzle, F. Glover, G. Kochenberger, Iterated local search, in: Handbook of Metaheuristics, 2002, pp. 321–353.
[8] M. Caramia, P. Dell'Olmo, G.F. Italiano, CHECKCOL: improved local search for graph coloring, J. Discrete Algoritms 4 (2006) 277–298.
[9] M. Chiarandini, I. Dumitrescu, T. Stützle, Stochastic local search algorithms for the graph colouring problem, in: Handbook of Approximation Algorithms and Metaheuristics, 2007, pp. 1–63.
[10] M. Rezapoor Mirsaleh, M.R. Meybodi, A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem, Memetic Comput. 8 (2016) 211–222.
[11] E.L. Lawler, A note on the complexity of the chromatic number problem, Inf. Process. Lett. 5 (1976) 66–67.
[12] B. Jesper Makholm, Exact algorithms for graph colouring and exact satisfiability, Oper. Res. Lett. 32 (2004) 547–556.
[13] R. Beigel, D. Eppstein, 3-coloring in time O (n1. 3289), J. Algorithms 54 (2005) 168–204.
[14] J.M. Byskov, Enumerating maximal independent sets with applications to graph colouring, Oper. Res. Lett. 32 (2004) 547–556.
[15] A. Hertz, D. de Werra, Using tabu search techniques for graph coloring, Computing 39 (1987) 345–351.
[16] C. Fleurent, J.A. Ferland, Genetic and hybrid algorithms for graph coloring, Ann. Oper. Res. 63 (1996) 437–461.
[17] R. el Dorne, J. Hao, A New Genetic Local Search Algorithm for Graph Coloring, 1998.
[18] P. Galinier, J.-K. Hao, Hybrid evolutionary algorithms for graph coloring, J. Comb. Optim. 3 (1999) 379–397.
[19] K.S. Narendra, M.A.L. Thathachar, Learning Automata: an Introduction, Prentice-Hall Inc., 1989.
[20] K.S. Narendra, M. Thathachar, Learning automata-a survey, IEEE Trans. Syst. Man Cybern. (1974) 323–334.
[21] M.A.L. Thathachar, K.M. Ramachandran, Asymptotic behaviour of a learning algorithm, Int. J. Control 39 (1984) 827–838 (1984/04/01).
[22] H. Beigy, M.R. Meybodi, A mathematical framework for cellular learning automata, Adv. Complex Syst. 7 (2004) 295–319.
[23] M. Thathachar, P.S. Sastry, Varieties of learning automata: an overview, Syst. Man Cybern. IEEE Trans. Part B: Cybern. 32 (2002) 711–722.
[24] G. Santharam, P. Sastry, M. Thathachar, Continuous action set learning automata for stochastic optimization, J. Franklin Inst. 331 (1994) 607–628.
[25] V. Phansalkar, M. Thathachar, Global convergence of feedforward networks of learning automata, in Neural Networks, International Joint Conference on IJCNN (1992) 875–880.
[26] M.A.L. Thathachar, P.S. Sastry, Varieties of learning automata: an overview, IEEE Trans. Syst. Man Cybern. Part B Cybern. 32 (2002) 711–722.

[27] M. Fahimi, A. Ghasemi, A distributed learning automata scheme for spectrum management in self-organized cognitive radio network, IEEE Trans. Mob. Comput. (2016).
[28] M.H. Mofrad, O. Jalilian, A. Rezvanian, M.R. Meybodi, Service level agreement based adaptive Grid superscheduling, Fut. Gener. Comput. Syst. 55 (2016) 62–73.
[29] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Adaptive Petri net based on irregular cellular learning automata with an application to vertex coloring problem, Appl. Intell. (2016) 1–13.
[30] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Finding the shortest path in stochastic graphs using learning automata and adaptive stochastic petri nets, J. Uncertainty Fuzziness Knowl.-Based Syst. 25 (2017) 427–455.
[31] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Cellular adaptive Petri net based on learning automata and its application to the vertex coloring problem, Discrete Event Dynam. Syst. (2017) 1–32.
[32] A. Rezvanian, M.R. Meybodi, Stochastic graph as a model for social networks, Comput. Human Behav. 64 (2016) 621–640.
[33] M. Rezapoor Mirsaleh, M.R. Meybodi, A learning automata-based memetic algorithm, Genet. Programm. Evolvable Mach. 16 (2015) 399–453.
[34] M.K. Sohrabi, R. Roshani, Frequent itemset mining using cellular learning automata, Comput. Human Behav. 68 (2017) 244–253.
[35] M. Ghavipour, M.R. Meybodi, Irregular cellular learning automata-based algorithm for sampling social networks, Eng. Appl. Artif. Intell. 59 (2017) 244–259.
[36] A.M. Saghiri, M.R. Meybodi, An approach for designing cognitive engines in cognitive peer-to-peer networks, J. Netw. Comput. Appl. 70 (2016) 17–40.
[37] A. Rezvanian, M.R. Meybodi, A new learning automata-based sampling algorithm for social networks, Int. J. Commun. Syst. 30 (5) (2015) 1–21.
[38] M. Ghavipour, M.R. Meybodi, An adaptive fuzzy recommender system based on learning automata, Electr. Commer. Res. Appl. 20 (2016) 105–115.
[39] M.M.D. Khomami, A. Rezvanian, M.R. Meybodi, Distributed learning automata-based algorithm for community detection in complex networks, Int. J. Mod. Phys. B 30 (2016) 1650042.
[40] M. Rezapoor Mirsaleh, M.R. Meybodi, A Michigan memetic algorithm for solving the community detection problem in complex network, Neurocomputing 214 (2016) 535–545.
[41] M.H. Mofrad, S. Sadeghi, A. Rezvanian, M.R. Meybodi, Cellular edge detection: combining cellular automata and cellular learning automata, AEU-Int. J. Electron. Commun. 69 (2015) 1282–1290.
[42] B. Damerchilu, M.S. Norouzzadeh, M.R. Meybodi, Motion estimation using learning automata, Mach. Vision. Appl. 27 (2016) 1047–1061.
[43] R. Rastegar, M.R. Meybodi, A new evolutionary computing model based on cellular learning automata, IEEE Conference on Cybernetics and Intelligent Systems 1 (2004) 433–438.
[44] R. Vafashoar, M.R. Meybodi, A.H. Momeni Azandaryani, CLA-DE: a hybrid model based on cellular learning automata for numerical optimization, Appl. Intellig 36 (2012) 735–748.
[45] J.K. Kordestani, A. Ahmadi, M.R. Meybodi, An improved Differential Evolution algorithm using learning automata and population topologies, Appl. Intellig. 41 (2014) 1150–1169.
[46] C. Dai, Y. Wang, M. Ye, X. Xue, H. Liu, An orthogonal evolutionary algorithm with learning automata for multiobjective optimization, IEEE Trans. Cybern. (2015) 1–14.
[47] M. Mahdaviani, J.K. Kordestani, A. Rezvanian, M.R. Meybodi, LADE: learning automata based differential evolution, Int. J. Artif. Intell. Tools 24 (2015) 1550023.
[48] M. Rezapoor Mirsaleh, M.R. Meybodi, A new criteria for creating balance between local and global search in memetic algorithms, Iran. J. Electr. Comput. Eng. (IJECE) 12 (2014) 31–37.
[49] P. Galinier, A. Hertz, A survey of local search methods for graph coloring, Comput. Oper. Res. 33 (2006) 2547–2562.
[50] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning, Oper. Res. 39 (1991) 378–406.
[51] F.T. Leighton, A graph coloring algorithm for large scheduling problems, J. Res. Nat. Bur. Stand. 84 (1979) 489–506.
[52] X. Zhuang, Z. Chen, Strategy entropy as a measure of strategy convergence in reinforcement learning, First International Conference on Intelligent Networks and Intelligent Systems (2008) 81–84.

Mehdi Rezapoor Mirsaleh received the B.Sc. in Computer Engineering from Kharazmi University, Tehran, Iran in 2000. He also, received the M.Sc. and Ph.D. degrees in Artificial Intelligence from Amirkabir University of Technology, Tehran, Iran in 2003 and 2016, respectively. His research interests include learning systems, machine learning and soft computing.

Mohammad Reza Meybodi received the B.Sc. and M.Sc. degrees in Economics from the Shahid Beheshti University, Tehran, Iran, in 1973 and 1977, respectively. He also received the M.Sc. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.