

# Balancing exploration and exploitation in memetic algorithms: A learning automata approach

Mehdi Rezapoor Mirsaleh<sup>1</sup>  | Mohammad Reza Meybodi<sup>2</sup>

<sup>1</sup>Department of Computer Engineering and Information Technology, Payame Noor University, Tehran, Iran

<sup>2</sup>Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

## Correspondence

Mehdi Rezapoor Mirsaleh,  
Department of Computer Engineering and Information Technology, Payame Noor University (PNU), PO Box 19395-3697, Tehran, Iran.  
Email: mrezapoor@aut.ac.ir

## Abstract

One of the problems with traditional genetic algorithms (GAs) is premature convergence, which makes them incapable of finding good solutions to the problem. The memetic algorithm (MA) is an extension of the GA. It uses a local search method to either accelerate the discovery of good solutions, for which evolution alone would take too long to discover, or reach solutions that would otherwise be unreachable by evolution or a local search method alone. In this paper, we introduce a new algorithm based on learning automata (LAs) and an MA, and we refer to it as LA-MA. This algorithm is composed of 2 parts: a genetic section and a memetic section. Evolution is performed in the genetic section, and local search is performed in the memetic section. The basic idea of LA-MA is to use LAs during the process of searching for solutions in order to create a balance between exploration performed by evolution and exploitation performed by local search. For this purpose, we present a criterion for the estimation of success of the local search at each generation. This criterion is used to calculate the probability of applying the local search to each chromosome. We show that in practice, the proposed probabilistic measure can be estimated reliably. On the basis of the relationship between the genetic section and the memetic section, 3 versions of LA-MA are introduced. LLA-MA behaves according to the Lamarckian learning model, BLA-MA behaves according to the Baldwinian learning model, and HLA-MA behaves according to both



the Baldwinian and Lamarckian learning models. To evaluate the efficiency of these algorithms, they have been used to solve the graph isomorphism problem. The results of computer experimentations have shown that all the proposed algorithms outperform the existing algorithms in terms of quality of solution and rate of convergence.

#### KEYWORDS

exploitation, exploration, learning automata (LAs), local search, memetic algorithm (MA)

## 1 | INTRODUCTION

Genetic algorithms (GAs) belong to a famous class of evolutionary algorithms (EAs) that are inspired by natural evolution and used in a vast variety of applications in academia and industry. The GA creates a population of solutions (chromosomes) and applies genetic operators such as crossover and mutation to explore the search space. Premature convergence and not finding the global optimum within an acceptable time are 2 main problems in traditional GAs. The memetic algorithm (MA), which is an extension of the traditional GA, uses the combination of evolution for the exploration of the search space and local search methods for the exploitation of the search space. An MA can accelerate the discovery of the global optimum and avoid premature convergence by creating a balance between exploration and exploitation. There are 2 learning models for MAs in the literature: Lamarckian and Baldwinian. In the Lamarckian learning model, the local search method is used as a refinement of the genetic operator that modifies the genetic structure of a solution and places it back in the genetic population.<sup>1</sup> The Lamarckian learning model can increase the speed of the search process; however, it can damage schema processing by changing the genetic structure of solutions, which may lead to premature convergence.<sup>2,3</sup> The Baldwinian learning model improves the fitness of a solution by applying local search, whereas the genotype remains unchanged. This increases the chances of the solution to remain in the next generations. Similar to natural evolution, Baldwinian learning does not modify the genetic structure of a solution but increases its chances of survival. Unlike the Lamarckian learning model, the Baldwinian approach does not allow parents to transfer what they have learned to their children. In both approaches, the key point is providing a good balance between exploration and exploitation in the process of searching the search space.

In the first part of this paper, we present a new MA called LA-MA. LA-MA is a process involving the decision of selecting evolution or a local search method in locating the global optimum. The basic idea of LA-MA is to use learning automata (LAs) during the process of searching for solutions in order to create a balance between exploration and exploitation. For this purpose, we introduce a new criterion to calculate the probability of applying local search to each chromosome.

The proposed algorithm is composed of 2 parts: a genetic section and a memetic section. The histories of evolution are saved in the genetic section, and the histories of the local search are saved in the memetic section. The memetic section consists of a meme (local search). A meme



is composed of a set of LAs. The action probability vectors of LAs are updated according to a learning algorithm based on the reinforcement signal feedback from the genetic section after applying the local search method. Based on the relationship between the genetic section and the memetic section, 3 versions of the LA-MA, called LLA-MA, BLA-MA, and HLA-MA, are introduced. LLA-MA behaves according to the Lamarckian learning model. That is, the parent chromosome and the chromosomes of the next generation are selected based on the chromosome's fitness calculated in the genetic section. BLA-MA behaves according to the Baldwinian learning model. That is, the history of the local search extracted by LAs is the effective term for chromosome selection. HLA-MA behaves according to both the Baldwinian and Lamarckian learning models. In HLA-MA, the parent chromosome and the chromosomes of the next generation are selected based on the chromosome's fitness calculated in the genetic section as well as the history of the local search extracted by LAs in the memetic section. On the issue of selecting appropriate chromosomes that should undergo local search, fitness- and distribution-based strategies were studied for adopting the probability of applying local search on the population of chromosomes in search problems. All 3 versions of LA-MA were performed according to the fitness-based strategy to calculate the probability of applying local search on each chromosome.

The rest of this paper is organized as follows. In the next section, an overview of the MAs is presented. In Section 3, LAs are briefly described. LA-MA is introduced in Section 4. A theoretical analysis of the new algorithm is described in Section 5. The solutions to the OneMax and graph isomorphism problems, as 2 applications of LA-MA, are given in Section 6. This section includes implementation considerations, simulation results, and a comparison with other algorithms to highlight the contributions of the LA-MA. Section 7 is the conclusion.

## 2 | RELATED WORK

One of the issues in MA design is how often the local search should be applied, ie, local search frequency. Local search methods, depending on the complexity of the problem and its parameters, act very differently. Some of the local search methods perform well in some of the phases of the problem but are not suitable in the other phases.<sup>4</sup> In the work of Hart,<sup>5</sup> the effect of local search methods on MA performance has been considered, where various configurations of the local search frequency at different sections of the MA were investigated. Conversely, it was shown in the work of Goldberg et al<sup>6</sup> that a fixed local search frequency leads to solutions with acceptable targets if the computational complexity of the local search is relatively low. A quantum EA (QEA),<sup>7-9</sup> which is based on the concept of quantum computing such as a quantum bit, a quantum gate, and the superposition of states, tries to create a balance between exploration and exploitation. Like the EAs, the main characteristics of a QEA are the chromosome, the fitness function, and the population of chromosomes. However, instead of numeric, binary, or symbolic representation, a QEA uses a probabilistic representation named Q-bit. A Q-bit is the smallest unit of information. A Q-bit individual is defined by a string of Q-bits by which a linear superposition of solutions can be represented in the search space probabilistically. The variation operator of a QEA is named Q-gate. A Q-gate drives the individuals toward better solutions and eventually toward a global optimum. Initially, a Q-bit individual represents the linear superposition of all possible solutions with the same probability. Therefore, it can represent diverse individuals probabilistically. When the probability of each Q-bit converges to 1 or 0 by the Q-gate, the Q-bit individual converges to a global optimum.



Another MA is a probabilistic memetic framework called PrMF, which can be classified as either a Lamarckian or a Baldwinian learning model.<sup>10</sup> This algorithm represents a first attempt to balance exploration and exploitation according to a derived theoretical upper bound for local search intensity. An important aspect of the PrMF, which performs according to the distribution-based strategy, is the determination of certain probabilistic measures based on the neighborhood structure of the local search.

In the work of Feng et al.,<sup>11</sup> a modeling of adaptive search as a symbiosis of genetic and memetic mechanisms in evolutionary search, working in sync on solving problems, has been considered.

In the work of Le et al.,<sup>12</sup> a conceptual modeling of the MA with evolvable local search has been introduced. In particular, the linear program structure for local search and the associated local search self-assembling process in the lifetime learning process of the MA have been proposed.

In the work of Chen and Ong,<sup>13</sup> a conceptual modeling of meme complexes (memplex) in search as a set of coadapted memes has been introduced. In particular, credit assignment criteria for meme coadaptation and the role of emergent memplexes in the lifetime learning process of an MA have been described.

Combining reinforcement learning algorithms with EAs is another technique to obtain a balance between exploration and exploitation. For this purpose, LAs, which is based on the reinforcement learning algorithms, have been used in several EAs.<sup>14</sup> Learning automata 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. Learning automata change their action selection mechanism in favor of the most promising actions according to responses from the environment.<sup>15,16</sup>

A recently reported MA, ie, GALA, is obtained from combining a GA with LAs, where LAs play the role of local search.<sup>14</sup> GALA combines a GA, used for its global search function, with LAs, used for its local search function. Object migration automata (OMA) represent chromosomes in GALA. Each state in OMA has 2 attributes: the value of the gene and the degree of association with its value. Information on the history of the local search process shows the degree of association between genes and their values. GALA creates a balance between exploration and exploitation by using information on the history of the local search kept in the states of OMA. GALA performs according to the Lamarckian learning model because it modifies the genotype and only uses a chromosome's fitness to fitness function computation. Modified GALA (MGALA), which is an extension of GALA, performs according to the Baldwinian learning model.<sup>17</sup> Unlike GALA, which only uses the value of genes for fitness computation, MGALA uses all the information in the OMA representation of the chromosome (ie, the degree of association between genes and their alleles as well as the value of genes) to compute the fitness function.

Another reinforcement learning algorithm is cellular LAs (CLAs), which is used in several EAs to create a balance between exploration and exploitation.<sup>18,19</sup> CLAs are cellular automata (CAs) in which one or more LAs are assigned to each of its cells.<sup>20</sup> The LA residing in a particular cell determines its action on the basis of its action probability vector. Like CA, there is a rule that CLAs operate under it. The rule of CLAs and the actions selected by the neighboring LAs of any particular LA determine the reinforcement signal to the LAs residing in that cell.

Several optimization methods such as CLA-EC, which is obtained from the combination of CLAs and evolutionary computing (EC),<sup>18</sup> and CLA-DE, which is obtained from the combination of CLAs and a differential evolution (DE) algorithm,<sup>19</sup> are reported in the literature. Both methods, with the aid of CLAs, try to create a balance between exploration and exploitation. Both CLA-EC



and CLA-DE methods are based on Lamarckian learning methods because in both methods, the current solutions change in every iteration.

In the CLA-EC model, each chromosome of the population is assigned to 1 cell of the CLA, and each cell in the CLA is equipped with a set of LAs, each of which corresponds to a gene of the assigned chromosome. The operation of the CLA-EC can be described as follows: in the first step, all LAs in the CLA-EC choose their actions synchronously. The set of actions chosen by the LA 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 the CLA-DE model, each cell of the CLA is composed of 2 parts: 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 LAs in which each automaton is restricted to 1 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. Hence, they can impact the learning process of each other.

A new variant of DE, called ADE-Grid, was proposed in the work of Kordestani et al,<sup>21</sup> which aims at controlling the diversity by adopting the mutation strategy, crossover rate, and scale factor during the run. In ADE-Grid, LAs 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 EA with LAs (OELA), which consists of a new fitness function based on the decomposition of the objective space and a quantization orthogonal crossover (QOX) with LAs, was proposed in the work of Dai et al.<sup>22</sup> 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 the aggregate function, was proposed to evenly decompose the objective space so that the diversity and the coverage of the obtained solutions to the true Pareto front can be well maintained. In addition, a QOX with LAs, which improves the search efficiency by using the past information, was also utilized in the algorithm, where the variables are purposely grouped by using the past information so as to quickly find optimal solutions.

In the work of Mahdavian et al,<sup>23</sup> 2 different classes of LA-based DE for the adaptive selection of crossover probability and mutation strategy in DE were proposed. In the first class, ie, GLADE, genomes of the population use the same mutation strategy and crossover probability. In the second class, ie, 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 our previous work,<sup>24</sup> a Michigan MA, called MLAMA-Net, was proposed for solving the community detection problem. The MLAMA-Net algorithm is an EA 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 that is isomorphic to the input network. Each node has a chromosome and an LA. The chromosome

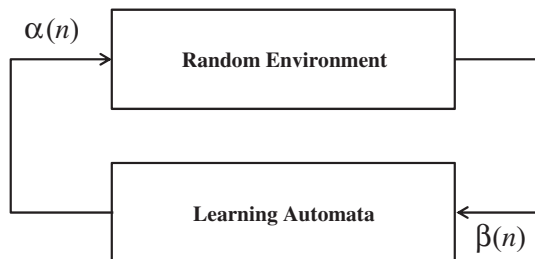


represents the community of corresponding nodes and saves the histories of exploration. The LA represents a meme and saves the histories of exploitation. The MLAMA-Net algorithm is a distributed algorithm in which each chromosome locally evolves by evolutionary operators and improves by local search. 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.

### 3 | LEARNING AUTOMATA

An LA<sup>15</sup> is an adaptive decision-making unit. It can be described as the 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 the automata after evaluating the input action. The LAs process the response of the environment and update its action probability vector. By repeating this process, the automaton learns to choose the optimal action such that the average penalty obtained from the environment is minimized. The environment is represented by a triple  $\langle \underline{\alpha}, \underline{\beta}, \underline{c} \rangle$ .  $\underline{\alpha} = \{\alpha_1, \dots, \alpha_r\}$  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  $\underline{c}$  corresponds to 1 input action  $\alpha_i$ . When the penalty probabilities are constant, the random environment is said to be a stationary random environment. It is called a nonstationary environment, if they vary with time. Depending on the nature of the reinforcement signal, there are 3 types of environments: P-model, Q-model, and S-model. The environments, in which the output can take only 1 of 2 values of 0 or 1, are referred to as P-model environments. The reinforcement signal in the 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 the S-model. The relationship between the LA and the random environment is shown in Figure 1.

There are 2 main families of LAs:<sup>16</sup> fixed-structure LAs and variable-structure LAs. Variable-structure LAs are represented by a triple  $\langle \underline{\beta}, \underline{\alpha}, T \rangle$ , where  $\underline{\beta}$  is the set of inputs,  $\underline{\alpha}$  is the set of output actions, and  $T$  is the learning algorithm that is used to modify the action probability vector. Learning algorithms are a critical factor affecting the performance of variable-structure LAs. Suppose the LA 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 Equation 1, if the selected action  $\alpha_i(k)$  is rewarded by the random environment, and it is updated as given in



**FIGURE 1** The relationship between the learning automaton and the random environment



Equation 2, if the action taken is penalized.  $a$  and  $b$  denote the reward and penalty parameters, and  $r$  is the number of actions that can be taken by an LA.

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

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

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

Learning automata have a vast variety of applications in combinatorial optimization problems,<sup>25-28</sup> computer networks,<sup>25,29-34</sup> queuing theory,<sup>35</sup> image processing,<sup>36</sup> information retrieval,<sup>37,38</sup> adaptive control,<sup>39</sup> neural network engineering,<sup>40,41</sup> cloud computing,<sup>42</sup> social networks,<sup>24,43-45</sup> and pattern recognition.<sup>46</sup>

## 4 | THE PROPOSED LA-MA

LA-MA is obtained from the combination of the MA and LAs. This algorithm is composed of 2 parts: genetic section and memetic section. Evolution is performed in the genetic section, and local search is performed in the memetic section. The memetic section is equipped with a set of LAs that are used to balance the process of exploration performed by evolution and the process of exploitation performed by local search. In what follows, the genetic and memetic sections are described.

### 4.1 | Genetic section

Similar to a canonical GA, the genetic section performs the global search and consists of a population of chromosomes with size  $N$ , a mutation operator, a crossover operator, and a fitness function. Each chromosome is composed of  $n$  genes, and each gene can take 1 of  $m$  possible values from set  $V = \{v_1, v_2, \dots, v_m\}$ . Chromosome  $i$  is denoted by  $CR^i = [CR_1^i, CR_2^i, \dots, CR_n^i]$ , where  $CR_k^i$  is the value of the  $k$ th gene,  $1 \leq i \leq N$ ,  $1 \leq k \leq n$ , and  $CR_k^i \in V$ . Initial population is created randomly. At each iteration, crossover operator is performed on selected parents with rate  $r_c$ . New chromosomes are selected based on a crowding mechanism,<sup>47</sup> and then, mutation operator is applied with rate  $r_m$ . The fitness of chromosome  $CR^i$  at generation  $t$ , which is referred to as genetic fitness, denoted by  $GF^i(t)$ , is a maximizing function.  $GF^i(t)$  is rescaled to a number in  $[0, 1]$ .

### 4.2 | Memetic section

The memetic section consists of a meme that corresponds to a local search method. The meme saves the effect (history) of its corresponding local search method. The meme is composed of  $n$  LAs, each of which corresponds to a gene of the chromosome. Specifically, the  $i$ th automaton corresponds to the  $i$ th gene of the chromosome. Each LA has  $m$  actions corresponding to  $m$  possible values that each gene can take from set  $V$ . The effect (history) of the local search method at



generation  $t$  is represented by the action probability vectors of LAs in the meme as given by the following equation:

$$M(t) = [M_1(t), M_2(t), \dots, M_n(t)], \quad (3)$$

where  $M_j(t) = [M_{j1}(t), M_{j2}(t), \dots, M_{jm}(t)]'$ ,  $1 \leq j \leq n$  and  $\forall j$ ,  $\sum_{k=1}^m M_{jk}(t) = 1$ .  $M_{jk}(t)$  denotes the probability that action  $k$  of the  $j$ th LA in the meme (corresponding to the value  $v_k$  of gene  $j$  of a chromosome) is selected in the exploitation process. In other words,  $M_{jk}(t)$  is the probability that value  $v_k$  is selected by the local search method for gene  $j$ .  $M_{jk}(0)$ , where  $1 \leq j \leq n$  and  $1 \leq k \leq m$ , is initially set to  $1/m$ . After the local search has been applied, the action probability vectors of the meme are updated according to a learning algorithm based on the reinforcement signal received from the genetic section.

$MF^\alpha(t) = \prod_{j=1}^n M_{jk}(t)$ , where  $k$  is the action of automaton  $j$  in the meme, which corresponds to the value of gene  $j$  in chromosome  $\alpha$  (that is,  $v_k = CR_j^\alpha$ ), is the probability of locating chromosome  $\alpha$  in the basin of attraction containing the best chromosome found so far by the local search method at generation  $t$ .  $MF^\alpha(t)$  is referred to as memetic fitness of chromosome  $\alpha$  at generation  $t$ . The memetic fitness of a chromosome changes when the action probability vectors of LAs in the meme are updated. Updating is performed on the basis of the result of applying the local search method on each chromosome. If the values of gene  $j$  of chromosome  $\alpha$  ( $CR_j^\alpha$ ) are the same before and after applying the local search method on chromosome  $\alpha$ , the action of the  $j$ th LA of the meme, which corresponds to the value  $CR_j^\alpha$  of chromosome  $\alpha$ , is rewarded and penalized, otherwise. It is worth noting that local search changes only the action probability vectors of the meme, not the values of the genes. That is, local search only changes the memetic fitness, not the genetic fitness. Unlike most previous works on the MA and LAs, which behave according to the Lamarckian learning model,<sup>14,18,19,21</sup> LA-MA behaves according to Lamarckian or Baldwinian learning models. On the basis of the way parent chromosomes are selected and whether or not we use LAs to generate new chromosomes to be added to the population, we introduce 3 learning models as follows.

- Lamarckian learning model (LLA-MA)
- Baldwinian learning model (BLA-MA)
- Hybrid (Lamarckian-Baldwinian) learning model (HLA-MA)

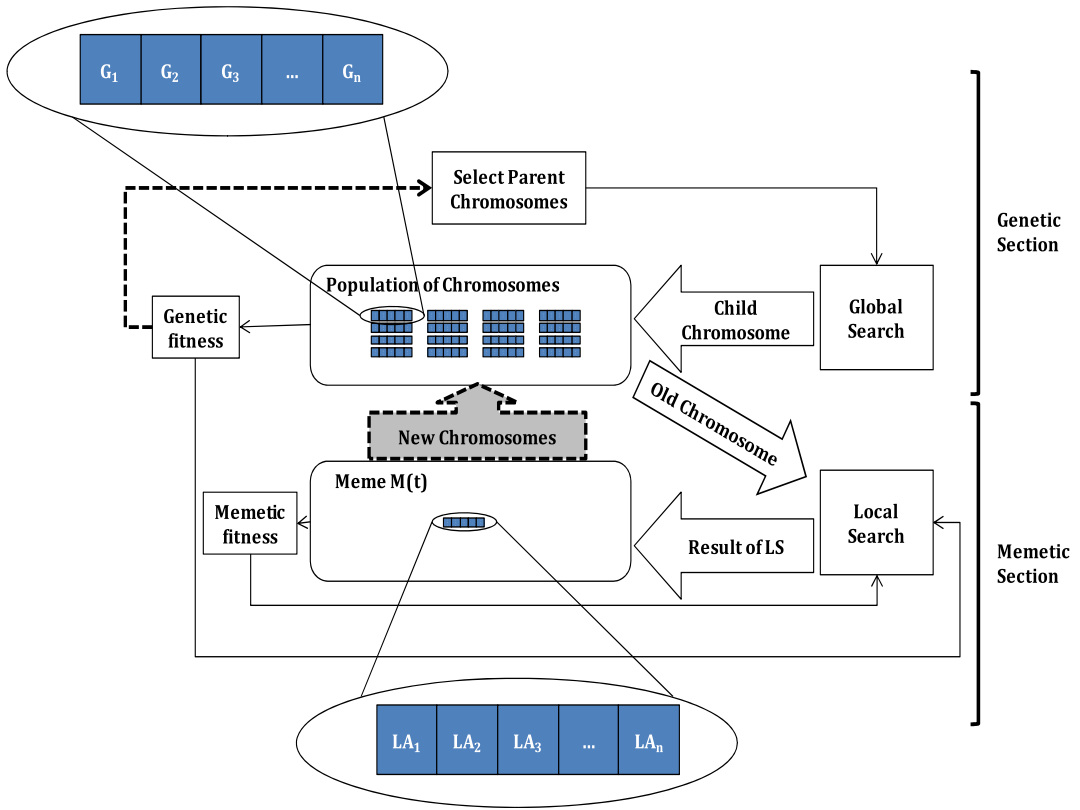
The relationship between the genetic section and the memetic section for LLA-MA, BLA-MA, and HLA-MA is shown in Figures 2, 3, and 4, respectively. To emphasize the differences between these learning models, we list the main characteristics of each of these models.

Main characteristics of LLA-MA:

1. Parent chromosomes from the current population are selected based on the genetic fitness using a tournament mechanism of size 2.
2. At each generation, a chromosome is generated by the set of LAs of the meme and added to the population.

Main characteristics of BLA-MA:

1. Parent chromosomes are selected from the current population on the basis of the memetic fitness using a tournament mechanism of size 2.
2. The set of LAs of the meme will be used only to determine the memetic fitness. Note that in BLA-MA, the set of LAs of a meme will not be used to generate new chromosomes to be added to the population.



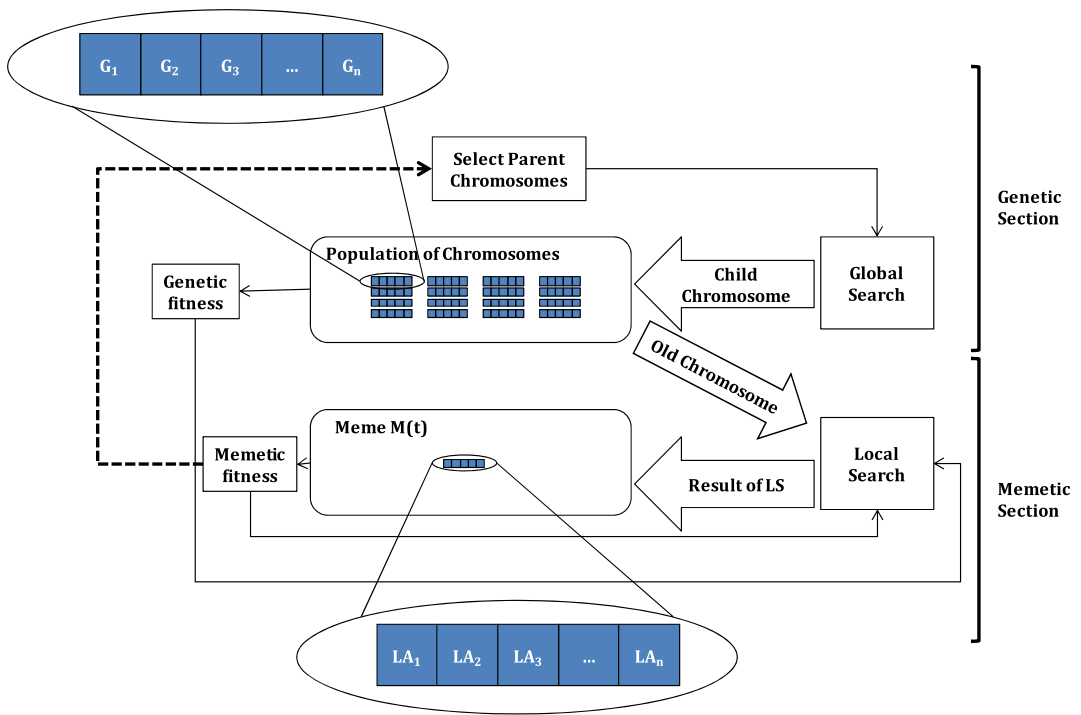
**FIGURE 2** The relationship between the genetic section and the memetic section in LLA-MA [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

#### Main characteristics of HLA-MA:

1. Parent chromosomes are selected using a tournament mechanism of size 2 based on hybrid fitness function  $HF^\alpha(t) = w_1 \cdot GF^\alpha(t) + w_2 \cdot MF^\alpha(t)$ , where  $w_1$  and  $w_2$  are 2 constants.
2. At each generation, a chromosome is generated by the set of LAs of the meme and added to the population.

The operation of the proposed algorithm can be described as follows.

Initial population is created randomly, and the probability of selecting an action for all LAs 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, the local search method is applied to all chromosomes based on a criterion described in the next section, and then, the action probability vectors of the meme (the history) are updated according to a learning algorithm based on the reinforcement signal received from the genetic section. In this step, chromosomes and their genetic fitness will not change. In the next step, parent chromosomes are selected based on the genetic fitness or memetic fitness (according to the relationship between the genetic and memetic sections in LLA-MA, BLA-MA, or HLA-MA, described in Figures 2, 3, and 4, respectively) for applying global search operators (crossover and mutation) at each generation. Then, in LLA-MA and HLA-MA, first, each LA in the meme chooses one of its actions (the value of the corresponding gene), and then, a new chromosome based on the set of actions chosen by the LA is generated and replaced



**FIGURE 3** The relationship between the genetic section and the memetic section in BLA-MA [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

with the worst chromosome of the population. Note that in BLA-MA, the set of LAs of a meme will not be used to generate new chromosomes.

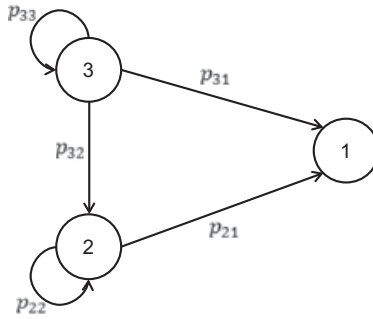
In what follows, we first give the pseudocode for LA-MA and is then demonstrated in more detail in Figure 5.

## 5 | THEORETICAL ANALYSIS OF LA-MA

The basic idea of LA-MA is to use LAs to create a balance between exploration and exploitation in the MA. To show this, the points of search space can be classified into 3 separate classes: Class 1, Class 2, and Class 3 ( $\text{Class 1} \cap \text{Class 2} = \Phi$ ,  $\text{Class 1} \cap \text{Class 3} = \Phi$ , and  $\text{Class 2} \cap \text{Class 3} = \Phi$ ). The first class (Class 1) includes all global optimum points. Class 2 includes points located in a basin of attraction containing Class 1 points.<sup>10</sup> These points can be converged to Class 1 by applying local search on them. In other words, the points in Class 2 have enough potential for converging to points of Class 1. Other points are in Class 3. These points, which are far from the global optimum, can be moved to Class 1 or Class 2 by applying evolutionary operators (global search) on them. Let  $p_1(t)$ ,  $p_2(t)$ , and  $p_3(t)$  be the probabilities of a chromosome being in Class 1, Class 2, and Class 3 at generation  $t$ , respectively. Clearly,  $p_1(t) + p_2(t) + p_3(t) = 1$ .

Let  $p_{ij}$  be the probability of moving a chromosome from Class  $i$  to Class  $j$  by applying the local search method or global search operators. If a chromosome is in Class 3, then it can be moved to Class 1 and Class 2, by applying global search operators, with probabilities  $p_{31}$  and  $p_{32}$ , respectively, and remains in Class 3 with probability  $p_{33}$ . If a chromosome is in Class 2, it can





**FIGURE 6** Transition of chromosomes among different classes

Here, we model exploration and exploitation in the MA as independent processes and analyze the probability of each process in locating the global optimum. Consider current generation  $t$ , the probability of finding at least 1 point of Class 1 as a result of global search on chromosomes of Class 3 can be derived according to the following equation:

$$p_{31} = p_1(t) + (1 - p_1(t)) \times p_1(t) + \dots + (1 - p_1(t))^{N-1} \times p_1(t) = 1 - (1 - p_1(t))^N, \quad (4)$$

where  $N$  is the population size. Furthermore, a chromosome in Class 3 can be moved to Class 2 or remains in Class 3 as a result of global search with probability  $p_{32} + p_{33} = 1 - p_{31} = (1 - p_1(t))^N$ .

The probability of finding at least 1 point of Class 1 as a result of the local search method on chromosomes of Class 2 can be derived according to the following equation:

$$p_{21} = p_2(t) + (1 - p_2(t)) \times p_2(t) + \dots + (1 - p_2(t))^{N-1} \times p_2(t) = 1 - (1 - p_2(t))^N. \quad (5)$$

Consequently,  $p_{22}$  can be derived as

$$p_{22} = 1 - p_{21} = (1 - p_2(t))^N. \quad (6)$$

Each chromosome can be converged to global optimum (Class 1) either by applying local search or applying global search. According to the discussion made in the previous paragraph,  $p^{gs} = p_3(t) \times p_{31}$  and  $p^{ls} = p_2(t) \times p_{21}$  are the probability of moving a chromosome to Class 1 using global search and the probability of moving a chromosome to Class 1 using local search, respectively. The design of an MA can be generalized as a decision of exploration against exploitation at each generation. Local search must be used when the probability that a chromosome reaches a point in Class 1 using the local search method ( $p^{ls}$ ) is higher than the probability that this chromosome reaches a point in Class 1 using global search ( $p^{gs}$ ), ie,

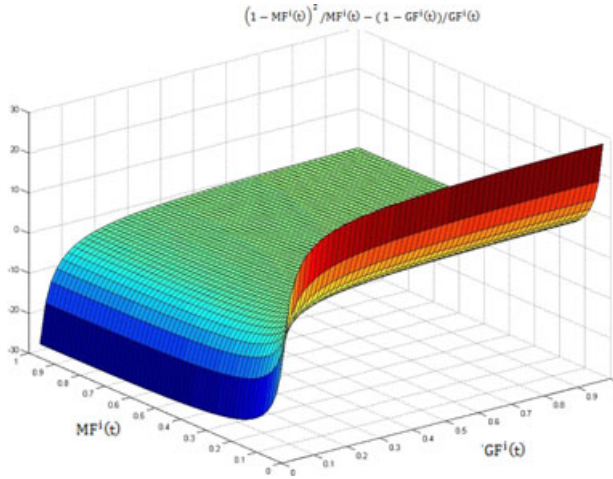
$$p^{ls} \geq p^{gs} \Leftrightarrow p_2(t) \times p_{21} \geq p_3(t) \times p_{31}.$$

Using Equations 4 and 5, we have

$$p_2(t) \times (1 - (1 - p_2(t))^N) \geq p_3(t) \times (1 - (1 - p_1(t))^N). \quad (7)$$

Therefore, if local search can be applied in each generation, then local search is more beneficial than global search if the above inequality holds. In the special case when  $N = 1$ , then inequality (7) can be written as

$$p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t) \Leftrightarrow p_2(t) \times p_2(t) \geq (1 - p_2(t) - p_3(t)) \times p_3(t). \quad (8)$$



**FIGURE 7** The plot of  $(1 - MF^i(t))^2 / MF^i(t) - (1 - GF^i(t)) / GF^i(t)$  versus  $GF^i(t)$  and  $MF^i(t)$  [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

Since the computation of  $p_1(t)$ ,  $p_2(t)$ , and  $p_3(t)$  is difficult for most problems, we may use the estimations of these parameters. For the proposed algorithm, we can use  $1 - GF^i(t)$  as an estimate for  $p_3(t)$ ,  $GF^i(t) \times MF^i(t)$  as an estimate for  $p_1(t)$ , and  $1 - p_1(t) - p_3(t) = GF^i(t) \times (1 - MF^i(t))$  as an estimate for  $p_2(t)$ . The estimation of  $p_{ij}$  that is the probability of moving a chromosome from Class  $i$  to Class  $j$  by global search or local search can be obtained using the estimated values of  $p_1(t)$ ,  $p_2(t)$ , and  $p_3(t)$  according to Equations 4 to 6. By substituting the estimated values of  $p_1(t)$ ,  $p_2(t)$ , and  $p_3(t)$  in inequality (8), we obtain

$$(1 - MF^i(t))^2 / MF^i(t) \geq (1 - GF^i(t)) / GF^i(t). \quad (9)$$

Figure 7 shows the plot of  $(1 - MF^i(t))^2 / MF^i(t) - (1 - GF^i(t)) / GF^i(t)$  versus  $GF^i(t)$  and  $MF^i(t)$ . From this Figure, we may say that inequality (9) will be satisfied (local search will be performed) as long as global search generates a chromosome with higher genetic fitness.

## 6 | EXPERIMENTAL RESULTS

### 6.1 | OneMax problem

The OneMax problem can be described as finding a string  $X = \{x_1, x_2, \dots, x_m\}$ , with  $x_i \in \{0, 1\}$ , that maximizes the following equation:

$$OneMax(X) = \sum_{i=1}^n x_i, \quad (10)$$

where  $x_i$  is the  $i$ th bit of string  $X$ , and  $n$  is the length of  $X$  and the global optimum value is  $n$  at  $X = 11 \dots 1$ . For solving OneMax by LA-MA, we define a chromosome to have  $n$  genes, and the value of genes is selected from  $\{0, 1\}$ .



### 6.1.1 | Genetic section for the OneMax problem

The genetic section in which the global search is performed consists of a population of chromosomes with size 1 ( $N=1$ ), mutation operator, and fitness function. If the mutation rate of each gene is  $r_m$ , then the probability of convergence of the only chromosome to global optimum (Class 1) by global search is calculated as  $p'_{31} = (1 - r_m)^{n-k} \times (r_m)^k$ , where  $k$  is the number of zero bits in the chromosome.

### 6.1.2 | Memetic section for the OneMax problem

In the OneMax problem, the memetic section consists of a meme that is composed of  $n$  LAs corresponding to  $n$  genes in the chromosome. Each LA has 2 actions, each of which corresponds to one of the values that a gene can take. The probability of choosing an action by an LA is initialized to 0.5.

For clarification purposes, let us consider one simple local search corresponding to the meme, where it first selects  $k$  bits (the number of zero bits in only a chromosome) randomly and then changes all zero bits of the selected  $k$  bits to 1. The average number of zero bits, which change to 1, can be computed as  $e = \sum_{i=0}^k i \times \binom{n-k}{k-i} \times \binom{k}{i} / \binom{n}{k}$ . Therefore, the probability of convergence of only the chromosome to global optimum (Class 1) by applying the local search method is  $p'_{21} = \binom{n-k}{k-e_r} \times \binom{k}{e_r} / \binom{n}{k}$ , where  $e_r = \text{round}(e)$ .

To study the efficiency of LA-MA in solving the OneMax problem, we have conducted 5 experiments (Experiments 1 to 5). For all experiments, an initial population with size 1 is created randomly, the mutation rate is set to 0.05, and the selection mechanism is (1 + 1). The criterion  $p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t)$  described in the previous section is used in applying the local search method in each generation. The algorithm terminates when global optimum is reached. The action probability vectors of the meme are updated according to the  $L_R - P$  learning algorithm with  $a = b = 0.5$ .

## 6.2 | Graph isomorphism problem

A graph is described by  $G = (E, V)$ , where  $V$  is the set of vertices and  $E \subset V \times V$  is the set of edges. Two graphs  $G = (E_1, V_1)$  and  $H = (E_2, V_2)$  are isomorphic if and only if their adjacency matrices  $M(G)$  and  $M(H)$  differ only by permutations of rows and columns, ie,  $M(G)$  and  $M(H)$  are related with a permutation  $\sigma$ , according to the following equation:

$$M(H) = P \times M(G) \times P^T \rightarrow [P \times M(G) \times P^T]_{i,j} = [M(H)]_{\sigma(i), \sigma(j)}, \quad (11)$$

where  $P$  is the permutation matrix of  $\sigma$ . If we define the difference between the 2 graphs as

$$J(\sigma) = ||M(H) - P \times M(G) \times P^T||, \quad (12)$$

- |                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ol style="list-style-type: none"> <li>1. <b>Procedure</b> Localsearch( CR )</li> <li>2. <math>g_1 = \text{Worst Gene}(\text{CR});</math></li> <li>3. <math>g_2 = \text{Select a Random Gene From Same Subset}(g_1);</math></li> <li>4. <math>\text{Swap}(\text{CR}(g_1), \text{CR}(g_2));</math></li> <li>5. <b>End</b> Localsearch;</li> </ol> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**FIGURE 8** Pseudocode for local search in the graph isomorphism problem



where  $\|\cdot\|$  is the matrix norm, defined as  $\|M\| = \sum_i \sum_j |m_{ij}|$ , the graph isomorphism problem can then be formulated as the search optimization problem in searching a permutation  $\sigma$  that minimizes  $J(\sigma)$ .

The mapping error of vertex  $k$  in graph  $G$  to vertex  $\sigma(k)$  in graph  $H$  is defined as

$$J_k(\sigma) = \sum_{m=1}^n |[M(H)]_{k,m} - [M(G)]_{\sigma(k),\sigma(m)}| + \sum_{m=1}^n |[M(H)]_{m,k} - [M(H)]_{\sigma(m),\sigma(k)}|, \quad (13)$$

where  $n$  is the number of vertices of  $G$  and  $H$ . For undirected graphs, the mapping error can be computed according to Equation 14 as follows:

$$J_k(\sigma) = 2 \times \sum_{m=1}^n |[M(H)]_{k,m} - [M(H)]_{\sigma(k),\sigma(m)}|. \quad (14)$$

**TABLE 1** The results of the statistical test for the HLA-MA algorithm with  $w_1 = 0.5$  and  $w_2 = 0.5$  vs those for the HLA-MA algorithm with other values of  $w_1$  and  $w_2$

$w_2$	$w_1$	Statistical Parameters	Length of Chromosome		
			16	32	64
0.5	0.5	Avg.	118	570	856
		Std.	1.54E+01	4.27E+01	3.52E+01
		p-value			
0	1	Avg.	124	589	992
		Std.	8.62E+00	4.17E+01	4.57E+01
		p-value	7.22E-02	9.17E-02	1.50E-13
1	0	Avg.	156	596	1105
		Std.	2.26E+01	5.93E+01	3.14E+01
		p-value	2.11E-08	6.11E-02	6.21E-23
0.9	0.1	Avg.	166	632	1123
		Std.	2.16E+01	1.83E+01	1.43E+01
		p-value	7.93E-11	4.71E-08	1.95E-26
0.8	0.2	Avg.	159	689	1423
		Std.	1.23E+01	1.13E+01	1.64E+01
		p-value	3.04E-12	5.09E-15	1.48E-35
0.7	0.3	Avg.	152	596	996
		Std.	9.50E+00	1.23E+01	1.49E+01
		p-value	3.28E-11	3.27E-03	1.55E-18
0.6	0.4	Avg.	143	635	968
		Std.	1.43E+01	3.53E+01	5.60E+01
		p-value	3.82E-07	4.97E-07	3.57E-10
0.4	0.6	Avg.	149	675	1103
		Std.	1.13E+01	3.68E+01	5.63E+01
		p-value	8.57E-10	4.17E-11	1.00E-18
0.3	0.7	Avg.	136	637	1136
		Std.	1.35E+01	3.86E+01	5.43E+01
		p-value	4.17E-05	5.70E-07	1.59E-20
0.2	0.8	Avg.	136	698	1130
		Std.	1.32E+01	3.75E+01	6.32E+01
		p-value	3.66E-05	4.60E-13	6.12E-19
0.1	0.9	Avg.	131	703	1098
		Std.	1.16E+01	3.86E+01	4.43E+01
		p-value	8.99E-04	2.45E-13	2.20E-20



Consequently, the mapping error of graphs  $G$  and  $H$  can be computed using the following equation:

$$J(\sigma) = \sum_{k=1}^n J_k(\sigma). \quad (15)$$

In this paper, we have used  $1/(1 + J(\sigma))$  as genetic fitness.

There are wide varieties of approaches that have been reported for solving the graph isomorphism problem in the literature. These approaches are classified as exact techniques that always find the optimal solution and approximation techniques that reduce the complexity at the cost of accepting suboptimal solutions. The first exact algorithm that significantly reduces the size of the search space is the backtracking algorithm proposed by Ullmann.<sup>48</sup> This algorithm has worst-case complexity of  $O(n!n^3)$ . Furthermore, the VF algorithm was proposed in the work of Cordella et al<sup>49</sup> with complexity order  $O(n!n)$ . An improved version of the VF algorithm, named VF2, was proposed in another work of Cordella et al,<sup>50</sup> in which the main improvement introduced is that the data structures employed during the exploration of the search space are organized in such a way that memory requirements are significantly reduced. Due to the exponential time requirement of exact algorithms, these techniques can only be used for small graphs, whereas there are very large graphs in a variety of applications. Hence, different approximation algorithms have been proposed for finding the near-optimal solution for the graph isomorphism problem in the literature. A GA was proposed in the work of Wang et al<sup>51</sup> for the graph isomorphism problem in which the characteristic of graphs, called status matching, has been adopted as a local search strategy. It is also illustrated that fitness functions using different matrix norms have great

**TABLE 2** Comparison of the number of fitness evaluations and results of the statistical test for different algorithms

Algorithm	Statistical Parameters	Length of Chromosome		
		16	32	64
HLA-MA	Avg.	118	570	856
	Std.	1.54E+01	4.27E+01	3.52E+01
	<i>p</i> -value			
LLA-MA	Avg.	124	589	992
	Std.	8.62E+00	4.17E+01	4.57E+01
	<i>p</i> -value	7.22E−02	9.17E−02	1.50E−13
BLA-MA	Avg.	156	596	1105
	Std.	2.26E+01	5.93E+01	3.14E+01
	<i>p</i> -value	2.11E−08	6.11E−02	6.21E−23
PrMF	Avg.	436	859	1569
	Std.	2.10E+02	1.86E+02	1.36E+02
	<i>p</i> -value	3.87E−09	3.72E−09	1.96E−22
CMA_0.5	Avg.	658	1254	2358
	Std.	9.83E+01	1.36E+02	1.90E+02
	<i>p</i> -value	2.80E−23	9.35E−22	1.02E−27
QEA	Avg.	768	2365	3125
	Std.	5.96E+01	1.25E+02	2.36E+02
	<i>p</i> -value	1.66E−31	1.23E−34	3.43E−30
GA	Avg.	1145	1896	4569
	Std.	6.85E+01	9.64E+01	1.56E+02
	<i>p</i> -value	1.34E−35	1.06E−33	2.30E−41

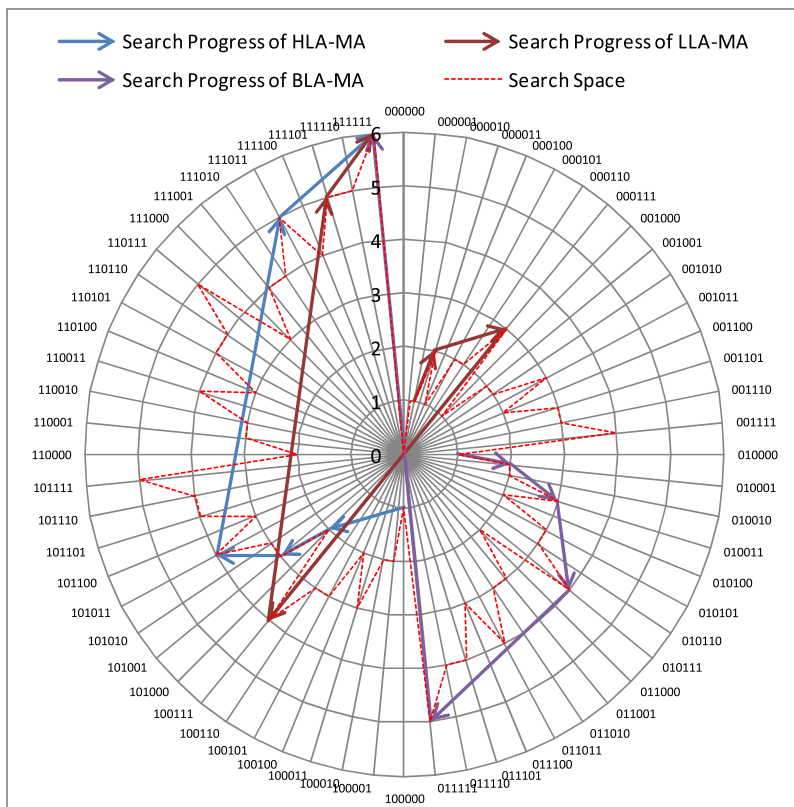


influence on the convergence of the algorithm. A multiobjective GA was proposed in the work of Choi et al.<sup>52</sup> for the graph isomorphism problem. The algorithm introduces an additional fitness function that reflects the potential of a solution. A hybrid GA with an incremental approach for the subgraph isomorphism problem was proposed in another work of Choi et al.<sup>53</sup> The incremental approach starts from finding a small-sized subgraph, and it gradually expands the size of the problem. An MA, which is based on permutation encoding, for the inexact graph isomorphism problem was proposed in the work of Bärecke and Detyniecki.<sup>54</sup> The algorithm shows that permutation-based encoding can even be applied to the graph isomorphism problem by differentiating between the phenotype and the genotype of the individuals.

### 6.2.1 | Genetic section for the graph isomorphism problem

The genetic section in the graph isomorphism problem is based on the traditional GA. The initial population is created randomly, and then, at the beginning of each iteration, crossover operator is performed on parent chromosomes with rate  $r_c$ , and mutation operator is applied with rate  $r_m$ .

**Crossover operator.** Crossover operator in the genetic section is performed as follows: two chromosomes  $CR^1$  and  $CR^2$  are selected as parent chromosomes. Moreover, 2 genes  $r_1$  and  $r_2$  are randomly selected from  $CR^1$  and  $CR^2$ . Then, for each gene in the range of  $[r_1, r_2]$  of  $CR^1$ , the value of the gene is exchanged with the value of the same gene in chromosome  $CR^2$ .



**FIGURE 9** Search space and search progress of the different versions of LA-MA for the OneMax problem with 6 bits [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]



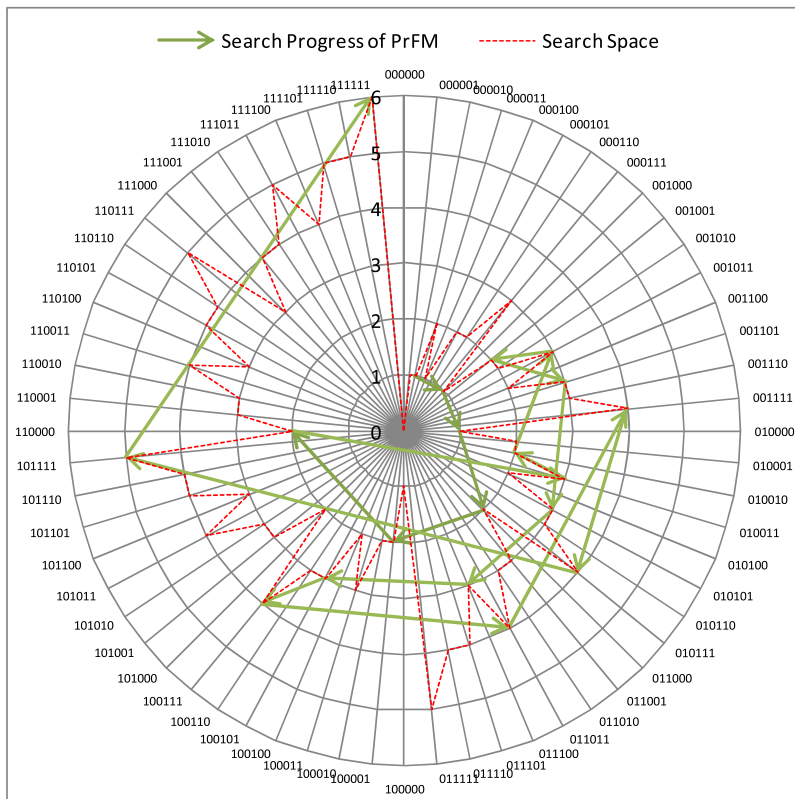
**Mutation operator.** Mutation operator is the same as the traditional GA. Two genes are selected randomly in parent chromosome, and then, their values are exchanged.

### 6.2.2 | Memetic section for the graph isomorphism problem

The memetic section in the proposed algorithm for the graph isomorphism problem consists of a meme that is composed of  $n$  LAs corresponding to  $n$  genes in the chromosome. Furthermore, for each LA, there are  $n$  actions corresponding to values of each gene in the chromosome. The action probability vectors are initialized with  $1/n$ . In designing the local search for the meme, we take into consideration the fact that if 2 graphs are isomorphic to each other, then the weight and the number of the input and output edges of isomorphic vertices must be equal.<sup>51</sup> The pseudocode for the local search method is given in Figure 8. This local search method consists of the following steps:

- The vertices are partitioned into a number of subsets with equal weights.
- The worst gene of the current chromosome is selected (line 2 in Figure 8).
- The value of the selected gene is swapped with the value of a random gene selected from the same subset (lines 3 and 4 in Figure 8).

To study the efficiency of LA-MA in solving the graph isomorphism problem, we have conducted 2 experiments (Experiments 6 and 7). For both experiments, an initial population is created



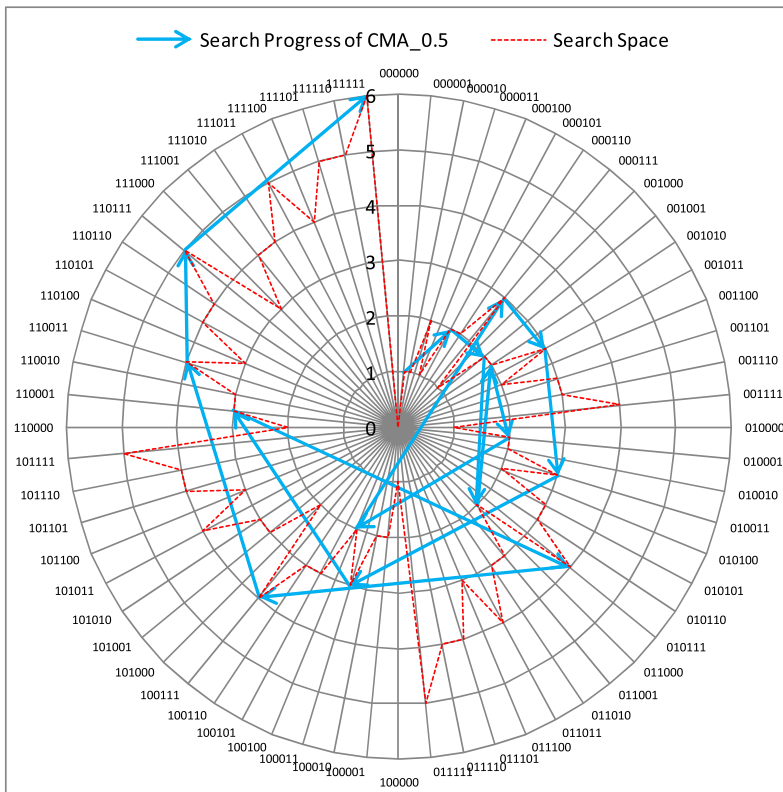
**FIGURE 10** Search space and search progress of the PrFM algorithm for the OneMax problem with 6 bits [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]



randomly, the mutation rate and the crossover rate are both set to 0.05, and the selection mechanism is  $(\mu + \lambda)$  based on the genetic fitness, memetic fitness, or hybrid fitness using a tournament mechanism of size 2 in LLA-MA, BLA-MA, and HLA-MA, respectively. The algorithm terminates when global optimum is reached. The action probability vectors of the meme are updated according to the  $L_{R-P}$  learning algorithm with  $a = b = 0.5$ . For this purpose, we have used a database with 10 000 couple of isomorphic graphs with different sizes.<sup>55</sup> This database is available at <http://amalfi.dis.unina.it/DBGGRAPH>.

### Experiment 1

This experiment is aimed at finding the optimal values of parameters  $w_1$  and  $w_2$  in the hybrid fitness function ( $HF^\alpha(t) = w_1 \cdot GF^\alpha(t) + w_2 \cdot MF^\alpha(t)$ ) for 3 different versions of the OneMax problem, ie, 16 bits, 32 bits, and 64 bits, for the HLA-MA algorithm. For this purpose, we studied the effect of parameters  $w_1$  and  $w_2$  on the number of fitness evaluations. Note that HLA-MA is equivalent to LLA-MA when  $w_1 = 1$  and  $w_2 = 0$  and HLA-MA is equivalent to BLA-MA when  $w_1 = 0$  and  $w_2 = 1$ . In order to provide confidence from the statistical point of view, we have performed several statistical tests ( $t$ -test) at the 95% significance level and 49 degrees of freedom. The null hypothesis is that there is no difference between the average number of fitness evaluations required by the HLA-MA algorithm with  $w_1 = 0.5$  and  $w_2 = 0.5$  and the average number of fitness evaluations required by the HLA-MA algorithm with other values of  $w_1$  and  $w_2$ . These  $t$ -tests have been performed after ensuring that data follow a normal distribution (by using



**FIGURE 11** Search space and search progress of the CMA\_0.5 algorithm for the OneMax problem with 6 bits [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

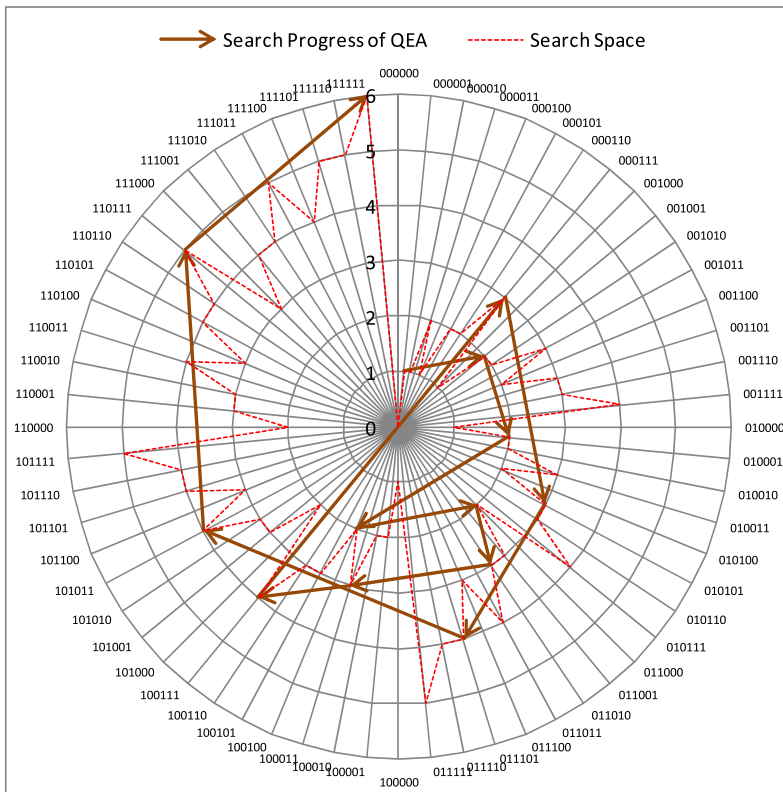


the Kolmogorov-Smirnov test). Table 1 shows the results for different versions of the OneMax problem with respect to the average number of fitness evaluations (average of 50 runs), standard deviation, and  $p$ -values of the 2-tailed  $t$ -test. From Table 1, we may conclude the following:

- For all 3 versions of the OneMax problem, the minimum value of the number of fitness evaluations is obtained when  $w_1 = 0.5$  and  $w_2 = 0.5$ .
- The difference between the performance of HLA-MA with  $w_1 = 0.5$  and  $w_2 = 0.5$  and the performance of HLA-MA with other values of  $w_1$  and  $w_2$  is statistically significant ( $p$ -value  $< 0.05$ ) for all 3 versions of the OneMax problem.

### Experiment 2

This experiment's goal was to evaluate the efficiency of the solution produced by LA-MA. In this experiment, all 3 versions of LA-MA (LLA-MA, BL-MA, and HLA-MA) are tested, and the results are compared with the result obtained for 4 other EAs: PrMF,<sup>10</sup> QEA,<sup>9</sup> canonical MA (CMA\_0.5), and GA for the OneMax problem in terms of the number of fitness evaluations performed by each algorithm. The PrMF is an MA in which the local search method is performed according to a defined neighborhood structure for local search.<sup>10</sup> The QEA is an EA that is based on the concept of quantum computing.<sup>7,9</sup> The CMA\_0.5 algorithm is an MA with local search similar to the local search used in our algorithm but performed with a probability of 0.5. Three versions of OneMax are considered: 16 bits, 32 bits, and 64 bits. In order to provide confidence



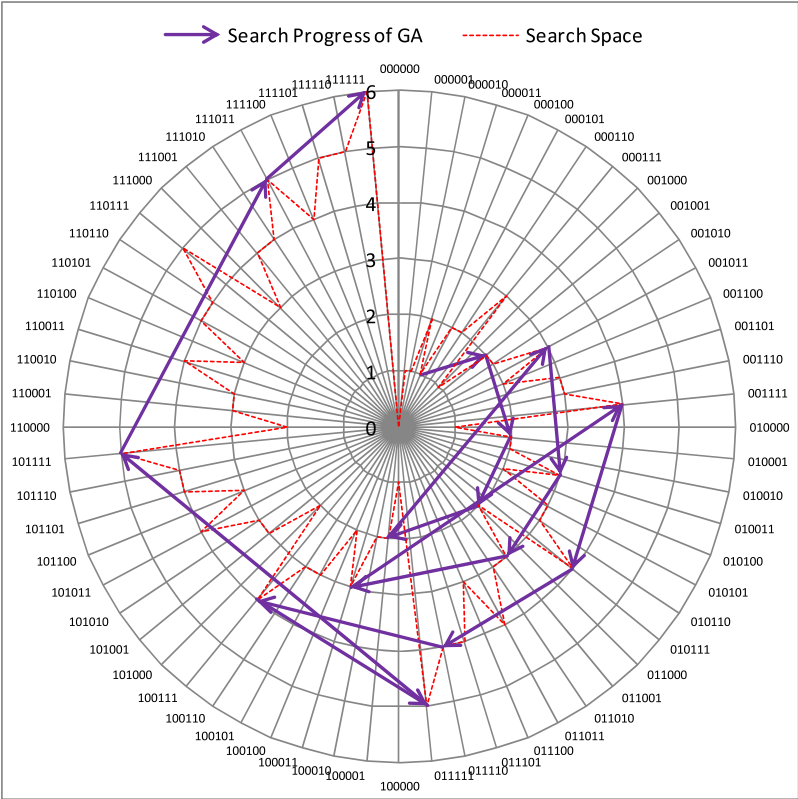
**FIGURE 12** Search space and search progress of the QEA algorithm for the OneMax problem with 6 bits [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

from the statistical point of view, we have performed several statistical tests ( $t$ -test) at the 95% significance level and 49 degrees of freedom. The null hypothesis is that there is no difference between the average number of fitness evaluations required by the HLA-MA algorithm and the average number of fitness evaluations required by other algorithms. These  $t$ -tests have been performed after ensuring that data follow a normal distribution (by using the Kolmogorov-Smirnov test). Table 2 presents the results for different versions of the OneMax problem with respect to the average number of fitness evaluations (average of 50 runs), standard deviation, and  $p$ -values of the 2-tailed  $t$ -test. From Table 2, we may conclude the following:

- All 3 versions of LA-MA outperform the other algorithms in terms of the number of fitness evaluations.
- HLA-MA performs better than BLA-MA and LLA-MA in terms of the number of fitness evaluations.
- The difference between the performance of HLA-MA and the performance of other algorithms is statistically significant ( $p$ -value < 0.05) for all 3 versions of the OneMax problem.

**Experiment 3**

The goal of this experiment was to study the search progress of the different algorithms for the OneMax problem. The radar charts in Figures 9 to 13 show the search space and search progress of the different versions of LA-MA (LLA-MA, BL-MA, and HLA-MA) and other algorithms for a



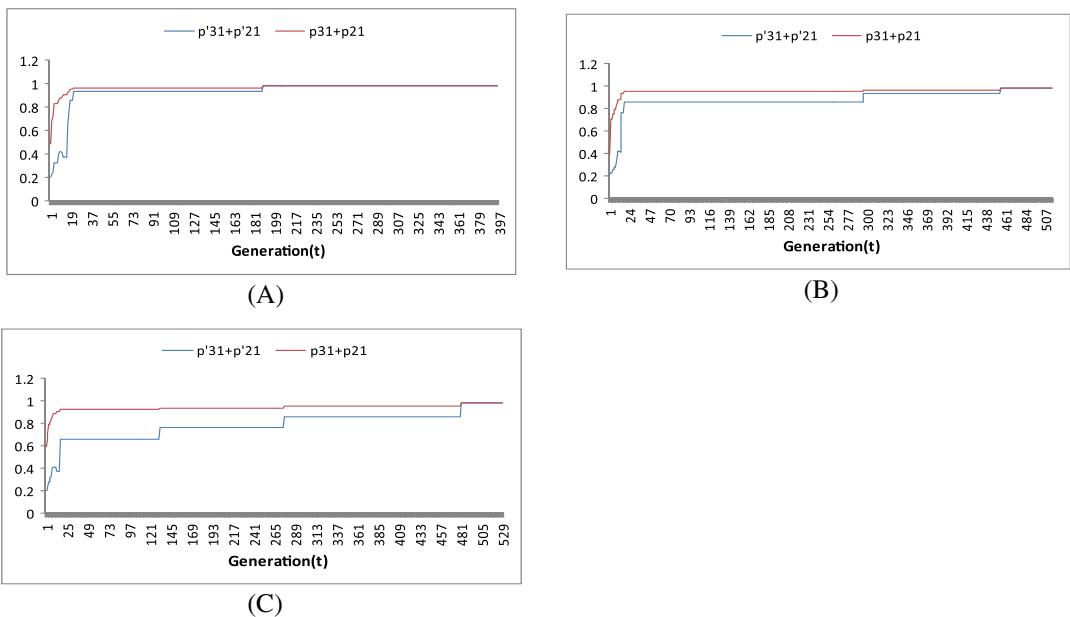
**FIGURE 13** Search space and search progress of the GA algorithm for the OneMax problem with 6 bits [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]



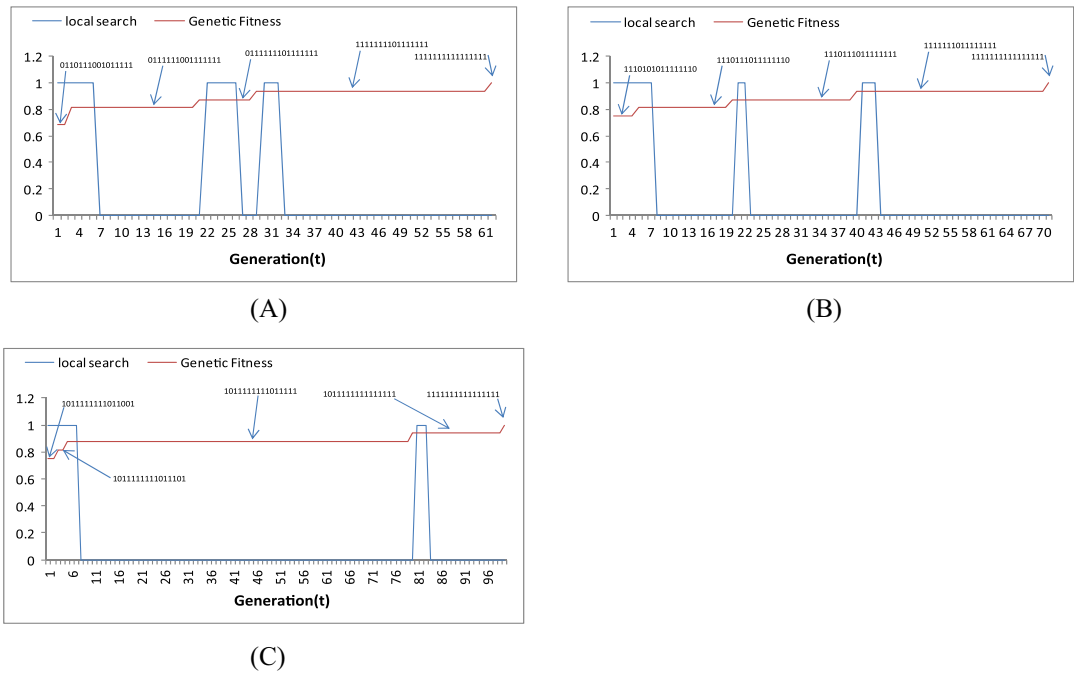
typical run for a 6-bit version of the OneMax problem. These Figures show the representation of each chromosome and its genetic fitness along a separate axis that starts at the center of the chart and ends on the outer ring. In Figure 9, for the search progress of HLA-MA, the initial population contains chromosome 000000 whose genetic fitness is zero. As shown by thick arrows, chromosomes 100000, 101000, 101001, 101011, 111011, and 111111 have been accessed by the algorithm during a typical run. For the search progress of LLA-MA, the initial population contains chromosome 000000, and chromosomes 000010, 000011, 000111, 100111, 111101, and 111111 have been accessed by the algorithm during a typical run. Furthermore, in BLA-MA, the initial population contains chromosome 000000, and chromosomes 010000, 010001, 010011, 010111, 011111, and 111111 have been accessed by the algorithm during a typical run. In Figures 10 to 13, the search progress of algorithms PrFM, CMA\_0.5, QEA, and GA is shown. In these Figures, the initial population contains chromosome 000000 whose genetic fitness is zero, but an intricate sequence of chromosomes has been accessed to reach the global optimum (111111) during a typical run.

#### Experiment 4

This experiment's goal was to evaluate the accuracy of the solution produced by LA-MA. In this experiment, we show that in all 3 versions of LA-MA, the value of  $p_{31} + p_{21}$  (the probability of convergence of the proposed algorithm to global optimum) estimated in the algorithm confirms with the calculated value  $p'_{31} + p'_{21}$  as described before. Note that  $p_1(t) = GF^i(t) \times MF^i(t)$ ,  $p_2(t) = GF^i(t) \times (1 - MF^i(t))$ ,  $p_3(t) = 1 - GF^i(t)$ , and according to Equations 4 and 5, we have  $p_{31} = p_1(t)$  and  $p_{21} = p_2(t)$ . Figure 14 shows the plot of the calculated probability ( $p'_{31} + p'_{21}$ ) and estimated probability ( $p_{31} + p_{21}$ ) for LLA-MA (A), BLA-MA (B), and HLA-MA (C). For this experiment, we assume that the length of the chromosome is 64. As the plots show, the simulation results confirm the theoretical analysis.



**FIGURE 14** Comparison of the calculated probability ( $p'_{31} + p'_{21}$ ) and estimated probability ( $p_{31} + p_{21}$ ) of convergence to global optimum for (A) HLA-MA, (B) LLA-MA, and (C) BLA-MA [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]



**FIGURE 15** Comparison of the efficiency of the defined criterion ( $p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t)$ ) in (A) HLA-MA, (B) LLA-MA, and (C) BLA-MA [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

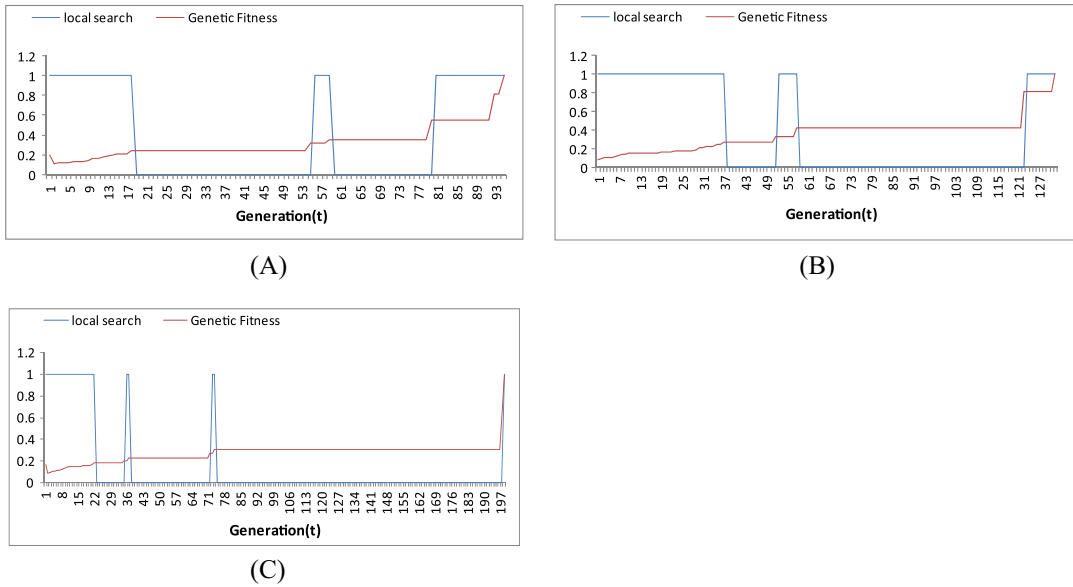
### Experiment 5

The goal of conducting this experiment was to study the applicability of the criterion  $p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t)$  described in Section 5 for applying local search in different versions of LA-MA. Figure 15 shows the plots of the genetic fitness versus generation number and the application of local search operation during the execution of the algorithm. In the second plot, a value of 1 at a given generation means that at this generation, local search has been applied. This experiment has been performed for HLA-MA (Figure 15A), LLA-MA (Figure 15B), and BLA-MA (Figure 15C). From these Figures, we may say that local search will be performed as long as global search generates a chromosome with higher fitness.

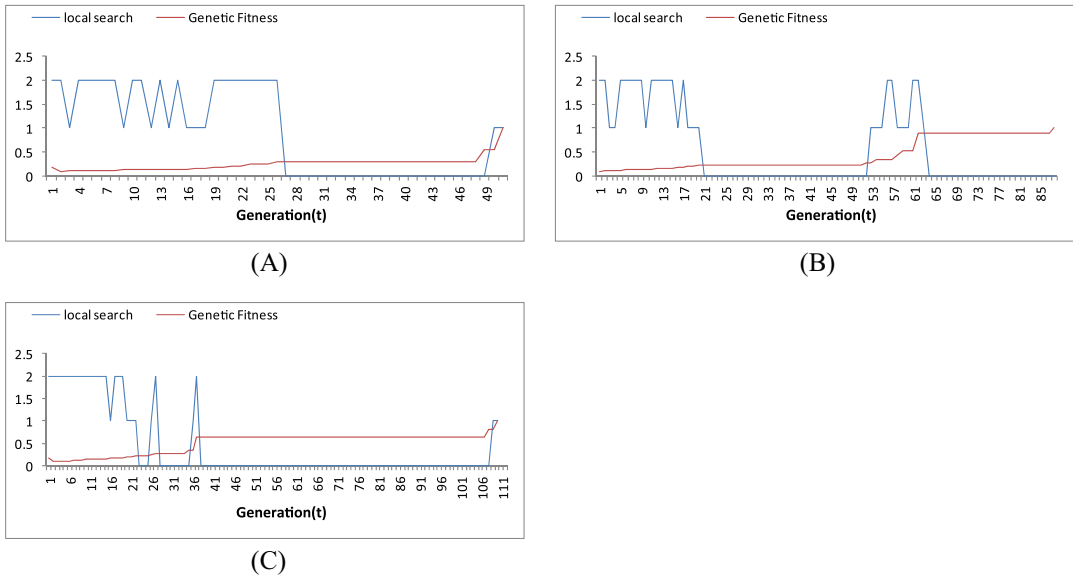
### Experiment 6

The goal of conducting this experiment was to study the applicability of the criterion for applying local search ( $p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t)$ ) in different versions of LA-MA. Figure 16 shows the plots of the genetic fitness versus generation number and the application of local search operation during the execution of the algorithm. In the second plot, a value of 1 at a given generation means that at this generation, local search has been applied. This experiment has been performed for HLA-MA (Figure 16A), LLA-MA (Figure 16B), and BLA-MA (Figure 16C). From these Figures, we may say that local search will be performed as long as global search generates a chromosome with higher fitness.

We repeat the previous experiment for a population with size 2 ( $N = 2$ ). An initial population is created randomly for a graph with size 20. Figure 17 shows the plots of the genetic fitness versus generation number and the application of local search operation during the execution of the algorithm. In the second plot, a value of 2 at a given generation means that at this generation, local search has been applied on both chromosomes of the population, and a value of

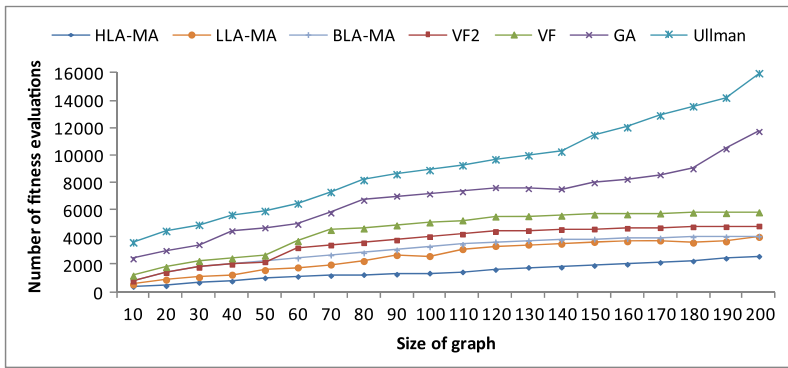


**FIGURE 16** Comparison of the efficiency of the defined criterion ( $p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t)$ ) in (A) HLA-MA, (B) LLA-MA, and (C) BLA-MA [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]



**FIGURE 17** Comparison of the efficiency of the defined criterion ( $p_2(t) \times p_2(t) \geq p_1(t) \times p_3(t)$ ) in (A) HLA-MA, (B) LLA-MA, and (C) BLA-MA [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

1 means that the local search has been applied on one of the chromosomes of the population. This experiment has been performed for HLA-MA (Figure 17A), LLA-MA (Figure 17B), and BLA-MA (Figure 17C). From these Figures, we may say that local search will be performed as long as global search generates a chromosome with higher fitness.



**FIGURE 18** Number of fitness evaluations vs graph size for different algorithms [Color figure can be viewed at [wileyonlinelibrary.com](http://wileyonlinelibrary.com)]

### Experiment 7

This experiment's goal was to evaluate the efficiency of the *solution produced by different* versions of LA-MA. In this experiment, all 3 versions of LA-MA are compared with 4 other algorithms: Ullmann,<sup>48</sup> VF, VF2,<sup>50</sup> and GA<sup>51</sup> for the graph isomorphism problem in terms of the number of fitness evaluations. The source codes for these algorithms are available at <http://amalfi.dis.unina.it/graph>. The initial population with size 100 is created randomly, the chromosome size is equal to the size of the graph, the mutation rate and the crossover rate are both set to 0.05, and the selection mechanism is  $(\mu + \lambda)$  based on the genetic fitness, memetic fitness, or hybrid fitness using a tournament mechanism of size 2 in LLA-MA, BLA-MA, and HLA-MA, respectively. Each algorithm terminates when either solution has been found or a number of fitness evaluations exceed 10 000. The result of this experiment is shown in Figure 18. Each result is the average of 50 runs. Graph size is varied from 10 to 200 by an increment of 10. The results show the superiority of all 3 versions of the proposed algorithm.

## 7 | CONCLUSION

In this paper, we have proposed a new MA called LA-MA. LA-MA is obtained from the combination of the MA and LAs. These LAs are responsible for balancing exploration and exploitation in the proposed MA. For this purpose, a criterion was introduced for the estimation of success of local search at each generation. According to the relationship between the genetic section and the memetic section, 3 versions of LA-MA, called LLA-MA, BLA-MA, and HLA-MA, were also introduced. The OneMax and graph isomorphism problems were considered as case studies in order to investigate the performance of these MAs. The results of experimentations showed the superiority of the proposed algorithms with respect to the quality of the solution and the rate of convergence. This research can be pursued in several directions such as by applying the proposed algorithms to solve optimization problems when the environment is dynamic, such as the dynamic shortest path problem or the dynamic traveling salesman problem, and improving the proposed algorithms by designing new criteria for balancing exploration and exploitation.

### ORCID

Mehdi Rezaipoor Mirsaleh  <http://orcid.org/0000-0002-1459-5792>



## REFERENCES

1. Chen X, Ong YS, Lim MH, Tan KC. A multi-facet survey on memetic computation. *IEEE Trans Evol Comput.* 2011;15:591-607.
2. Krasnogor N, Smith J. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IJCSI Int J Comput Sci Issues.* 2005;9:474-488.
3. Bandurski K, Kwedlo W. A Lamarckian hybrid of differential evolution and conjugate gradients for neural network training. *Neural Process Lett.* 2010;32:31-44.
4. Ong YS, Keane AJ. Meta-Lamarckian learning in memetic algorithms. *IEEE Trans Evol Comput.* 2004;8:99-110.
5. Hart WE. Adaptive Global Optimization With Local Search [PhD thesis]. San Diego: Computer Science and Engineering Department, University of California; 1994.
6. Goldberg DE, Voessner S. Optimizing global-local search hybrids. *Urbana.* 1999;51:220-228.
7. Han KH, Kim JH. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans Evol Comput.* 2002;6:580-593.
8. Zhang G. Quantum-inspired evolutionary algorithms: a survey and empirical study. *J Heuristics.* 2011;17:303-351.
9. Han K-H, Kim J-H. Quantum-inspired evolutionary algorithms with a new termination criterion, He gate, and two-phase scheme. *IEEE Trans Evol Comput.* 2004;8:156-169.
10. Nguten QH, Ong YS, Lim MH. A probabilistic memetic framework. *IEEE Trans Evol Comput.* 2009;13:604-623.
11. Feng L, Ong Y-S, Chen C, Chen X. Conceptual modeling of evolvable local searches in memetic algorithms using linear genetic programming: a case study on capacitated vehicle routing problem. *Soft Comput.* 2016;20:3745-3769.
12. Le MN, Ong YS, Jin Y, Sendhoff B. A unified framework for symbiosis of evolutionary mechanisms with application to water clusters potential model design. *IEEE Comput Intell Mag.* 2012;7:20-35.
13. Chen X, Ong YS. A conceptual modeling of meme complexes in stochastic search. *IEEE Trans Syst Man Cybern Part C Appl Rev.* 2012;42:612-625.
14. Rezapoor Mirsaleh M, Meybodi MR. A hybrid algorithm for solving graph isomorphism problem. Paper presented at: Proceedings of the Second International Conference on Information and Knowledge Technology (IKT2005); 2005; Iran, Tehran.
15. Narendra KS, Thathachar MAL. *Learning Automata: An Introduction.* Englewood Cliffs, NJ: Prentice-Hall; 1989.
16. Thathachar MAL, Sastry PS. Varieties of learning automata: an overview. *IEEE Trans Syst Man Cybern Part B: Cybernetics.* 2002;32:711-722.
17. Rezapoor Mirsaleh M, Meybodi MR. A learning automata-based memetic algorithm. *Genet Program Evolvable Mach.* 2015;16:399-453.
18. Rastegar R, Meybodi MR. A new evolutionary computing model based on cellular learning automata. Paper presented at: IEEE Conference on Cybernetics and Intelligent Systems, vol. 1; 2004; Singapore.
19. Vafashoar R, Meybodi MR, Momeni Azandaryani AH. CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. *Appl Intell.* 2012;36:735-748.
20. Beigy H, Meybodi MR. A mathematical framework for cellular learning automata. *Adv Compl Syst.* 2004;7:295-319.
21. Kordestani JK, Ahmadi A, Meybodi MR. An improved Differential Evolution algorithm using learning automata and population topologies. *Appl Intell.* 2014;41:1150-1169.
22. Dai C, Wang Y, Ye M, Xue X, Liu H. An orthogonal evolutionary algorithm with learning automata for multiobjective optimization. *IEEE Trans Cybern.* 2015;46:1-14.
23. Mahdavian M, Kordestani JK, Rezvanian A, Meybodi MR. LADE: Learning automata based differential evolution. *Int J Artif Intell T* 2015;24:1-13.
24. Rezapoor Mirsaleh M, Meybodi MR. A Michigan memetic algorithm for solving the community detection problem in complex network. *Neurocomputing.* 2016;214:535-545.
25. Akbari Torkestani J, Meybodi MR. Learning automata-based algorithms for finding minimum weakly connected dominating set in stochastic graphs. *Int Journal Uncertain Fuzz Knowl-Based Syst.* 2010;18:721-758.
26. Akbari Torkestani J. An adaptive focused Web crawling algorithm based on learning automata. *Appl Intell.* 2012;37:586-601.

27. Moradabadi B, Beigy H. A new real-coded Bayesian optimization algorithm based on a team of learning automata for continuous optimization. *Genet Program Evolvable Mach.* 2014;15:169-193.
28. Rezapoor Mirsaleh M, Meybodi MR. A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem. *Memet Comput.* 2016;8:211-222.
29. Akbari Torkestani J, Meybodi MR. An efficient cluster-based CDMA/TDMA scheme for wireless mobile ad-hoc networks: a learning automata approach. *J Netw Comput Appl.* 2010;33:477-490.
30. Akbari Torkestani J, Meybodi MR. Mobility-based multicast routing algorithm for wireless mobile ad-hoc networks: a learning automata approach. *Comput Commun.* 2010;33:721-735.
31. Akbari Torkestani J, Meybodi MR. An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata. *Comput Netw.* 2010;54:826-843.
32. Jahanshahi M, Dehghan M, Meybodi M. LAMR: learning automata based multicast routing protocol for multi-channel multi-radio wireless mesh networks. *Appl Intell.* 2013;38:58-77.
33. Kumar N, Lee J-H, Rodrigues JJ. Intelligent mobile video surveillance system as a Bayesian coalition game in vehicular sensor networks: learning automata approach. *IEEE Trans Intell Transp Syst.* 2015;16:1148-1161.
34. Saghiri AM, Meybodi MR. A self-adaptive algorithm for topology matching in unstructured peer-to-peer networks. *J Netw Syst Manag.* 2016;24:393-426.
35. Vahidipour SM, Meybodi MR, Esnaashari M. Learning automata-based adaptive Petri net and its application to priority assignment in queuing systems with unknown parameters. *IEEE Trans Syst Man Cybern: Syst.* 2015;45:1373-1384.
36. Mofrad MH, Sadeghi S, Rezvanian A, Meybodi MR. Cellular edge detection: combining cellular automata and cellular learning automata. *AEU-Int J Electron Commun.* 2015;69:1282-1290.
37. Hasanzadeh M, Meybodi MR. Distributed optimization grid resource discovery. *J Supercomput.* 2015;71:87-120.
38. Rezvanian A, Meybodi MR. Finding minimum vertex covering in stochastic graphs: a learning automata approach. *Cybern Syst.* 2015;46:698-727.
39. Unsal C, Kachroo P, Bay JS. Multiple stochastic learning automata for vehicle path control in an automated highway system. *IEEE Trans Syst Man Cybern Part A: Syst Hum.* 1999;29:120-128.
40. Beigy H, Meybodi MR. A learning automata-based algorithm for determination of the number of hidden units for three-layer neural networks. *Int J Syst Sci.* 2009;40:101-118.
41. Meybodi MR, Beigy H. Neural network engineering using learning automata: determining of desired size of three layer feed forward neural networks. *J Fac Eng.* 2001;34:1-26.
42. Morshedlou H, Meybodi MR. Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments. *IEEE Trans Cloud Comput.* 2014;2:156-167.
43. Rezvanian A, Meybodi MR. A new learning automata-based sampling algorithm for social networks. *Int J Commun Syst.* 2015;30:1-21.
44. Rezvanian A, Meybodi MR. Finding maximum clique in stochastic graphs using distributed learning automata. *Int J Uncertain Fuzz Knowl-Based Syst.* 2015;23:1-31.
45. Khomami MMD, Rezvanian A, Meybodi MR. Distributed learning automata-based algorithm for community detection in complex networks. *Int J Mod Phys B.* 2016;30:1-20.
46. Thathachar M, Phansalkar VV. Learning the global maximum with parameterized learning automata. *IEEE Trans Neural Netw.* 1995;6:398-406.
47. Deb K. *Multi-objective Optimization Using Evolutionary Algorithms*. Vol. 16. New York: Wiley; 2001.
48. Ullmann JR. An algorithm for subgraph isomorphism. *J ACM.* 1976;23:31-42.
49. Cordella LP, Foggia P, Sansone C, Vento M. Performance evaluation of the VF graph matching algorithm. In: *Proceedings of the International Conference on Image Analysis and Processing*; 1999; Venice, Italy.
50. Cordella LP, Foggia P, Sansone C, Vento M. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Trans Pattern Anal Mach Intell.* 2004;26:1367-1372.
51. Wang YK, Fan KC, Horng JT. Genetic-based search for error-correcting graph isomorphism. *IEEE Trans Syst Man Cybern Part B: Cybern.* 1997;27:588-597.
52. Choi J, Yoon Y, Moon BR. An efficient genetic algorithm for subgraph isomorphism. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*; 2012; Philadelphia, PA.
53. Choi H, Kim J, Moon BR. A hybrid incremental genetic algorithm for subgraph isomorphism problem. In: *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*; 2014; Vancouver, BC.



54. Bärecke T, Detyniecki M. Memetic algorithms for inexact graph matching. Paper presented at: IEEE Congress on Evolutionary Computation (CEC); 2007; Singapore.
55. Foggia P, Sansone C, Vento M. A database of graphs for isomorphism and sub-graph isomorphism benchmarking. In: Proceedings of the 3rd IAPR TC-15 International Workshop on Graph-based Representations; 2001; Venice, Italy.

**How to cite this article:** Rezapoor Mirsaleh M, Meybodi MR. Balancing exploration and exploitation in memetic algorithms: A learning automata approach. *Computational Intelligence*. 2017;1–28. <https://doi.org/10.1111/coin.12148>