LASPEA: Learning Automata-based Strength Pareto Evolutionary Algorithm for Multi-objective Optimization

Seyed Mahdi Jamei1, Mostafa Haghi Kashani2 and Ramin Karimi3

Abstract—Multi-objective optimization problems are currently gaining significant attentions from researchers because many real-world optimization problems consist of contradictory objectives. SPEA (Strength Pareto Evolutionary Algorithm) is one of the most successful multi-objective evolutionary algorithms for approximating the Pareto-optimal set for multi-objective optimization problems. In this paper, an improved version of SPEA-II, called LASPEA (Learning Automata-based Strength Pareto Evolutionary Algorithm) is proposed. The proposed algorithm incorporates problem-specific genetic operators and learning automata to improve the behavior of the optimization algorithm. Simulation results demonstrate the efficiency of the LASPEA in terms of convergence and diversity.

Index Terms—Multi-Objective Optimization, Evolutionary Algorithm, Learning Automata and Pareto-Front

I. INTRODUCTION

OVER the past decade, many multi-objective evolutionary algorithms (MOEAs) have been proposed [1]-[5]. The popularity of MOEAs is due to their ability to find multiple Pareto-optimal solutions in one single simulation run. Since evolutionary algorithms (EA) work with a population of solutions, a simple EA can be extended to maintain a diverse set of solutions. With an emphasis for moving toward the true Pareto-optimal region, an EA can be used to find multiple Pareto-optimal solutions in one single simulation run. Evolutionary techniques for multi-objective optimization are currently gaining significant attentions from researchers in various fields due to their effectiveness and robustness in searching for a set of global trade-off solutions [6]. This growing interest is reflected by the significantly increase number of different evolutionary-based approaches and variations of existing techniques published in technical literatures.

Multi-objective optimization is a highly demanding research topic because many real-world optimization problems consist of contradictory criteria or objectives. Considering these competing objectives concurrently, a multi-objective optimization problem (MOP) can be formulated as finding the best possible solutions that satisfy these objectives under different tradeoff situations. A family of solutions in the feasible solution space forms a Pareto-optimal front, which describes the tradeoff among several contradictory objectives of an MOP [7]. Generally, there are two goals in finding the Pareto-optimal front of a MOP: 1) to converge solutions as near as possible to the Pareto-optimal front; and 2) to distribute solutions as diverse as possible over the obtained non-dominated front. These two goals cause enormous search space in MOPs and let deterministic algorithms feel difficult to obtain the Pareto-optimal solutions. Therefore, satisfying these two goals simultaneously is a principal challenge for any algorithm to deal with MOPs [8]. In recent years, several evolutionary algorithms have been proposed to solve Multi-Objective Problems. For example, the strength Pareto evolutionary algorithm (SPEA) [5] and the non-dominated sorting genetic algorithm (NSGA-II) [9] are two most famous algorithms and have been reported to perform well. Several extensions of genetic algorithms (GAs) for dealing with MOPs are also proposed, such as the Niche Pareto Genetic Algorithm (NPGA) [3], the Chaos-Genetic Algorithm (CGA) [10], the Real Jumping Gene Genetic Algorithm (RJGGA) [11], PAES [12], Vector Evaluated Genetic Algorithm [13], and Multi-Objective Genetic Algorithm (MOGA) [14].

The main goals of each MOEA are convergence to the true Pareto optimal set, and maintenance of a uniform distribution of the Pareto front.

In this paper, an improved version of SPEA-II, called LASPEA (Learning Automata-based Strength Pareto Evolutionary Algorithm) is proposed. The proposed algorithm incorporates problem-specific genetic operators and learning automata to improve the behavior of the SPEA. Simulation results demonstrate the efficiency of the LASPEA in terms of convergence and diversity. The rest of the paper is organized as follows. In section II and section III, we have overviewed on multi-objective optimization and multi-objective evolutionary algorithm respectively. The proposed multi-objective evolutionary algorithm is presented in section IV. In section V, the experiments and results are provided. Finally, this paper concludes with a summary in section VI.

1Seyed Mahdi Jamei is with the Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran, (Email: Jamei@Qodsiau.ac.ir)
2Mostafa Haghi Kashani is with the Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran, (Email: Kashani@Qodsiau.ac.ir)
3Ramin Karimi is with the Department of Computer Engineering, Malard Branch, Islamic Azad University, Tehran, Iran, (Email: rakarimi1@gmail.com)
II. MULTI-OBJECTIVE OPTIMIZATION

Objectives in multi-objective problems often conflict with each other, so improving one objective will affect other objectives. Thus, there is no best single optimal solution regarding all the objective functions. In MOPs, there is a set of Pareto-optimal solutions or Pareto-front. MOPs formulation is as follow [1]:

Minimize or maximize $F(x) = (F_1(x), ..., F_m(x))$  

where $x$ belongs to the decision space, $F$ consists of $m$ objective function. A MOP can have various some constraints as follow:

$$
\begin{align*}
&g_i(x) \leq 0, \quad i = 1, ..., p \\
&h_j(x) = 0, \quad j = p + 1, ..., q
\end{align*}
$$

In this case, the solutions of the feasible region in the search space satisfy all constraints. In a MOP with both minimization and maximization objective functions, it is better to convert all objectives to minimization or maximization forms. In a MOP with $m$ objectives for minimization, the definitions of Pareto-Dominance, Pareto-optimal Set and Pareto-Front are as follows [1]:

**Pareto-Dominance**: decision variables $x$ is said to dominate decision variables $y (x \preceq y)$, if and only if:

$$
\begin{align*}
&F_i(x) \leq F_i(y), \quad \text{for every } i \in \{1, 2, ..., m\} \\
&F_j(x) < F_j(y), \quad \text{for at least one index } j \in \{1, 2, ..., m\}
\end{align*}
$$

**Pareto-optimal Set**: a solution $k$ is said to be Pareto-optimal (or non-dominated) if there is no another solutions that dominate $k$. The set of all Pareto-optimal solutions in the decision space is called the Pareto-optimal Set (PS).

**Pareto-Front**: The image of the PS in the objective space is called the Pareto-Front (PF).

$$PF = \{f(x) \mid x \in PS\}$$  

III. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM (MOEA)

A typical evolutionary algorithm has the following steps:

1. Population initialization
2. Fitness evaluation
3. Crossover
4. Mutation
5. Selection

In order to perform selection step, objective function which sometimes called fitness function are required. Also, a multi-objective optimization deals with $m$ ($m \geq 2$) different objective function. Combining these different objective functions into a one objective function is an elementary task of each multi-objective evolutionary algorithm. This task is fulfilled by means of concepts such as Pareto-Dominance and Pareto-Front which are described in previous section [6].

IV. PROPOSED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

In this section, an improved version of SPEA-II [15], called LASPEA (Learning Automata-based Strength Pareto Evolutionary Algorithm) is proposed. LASPEA is based on SPEA-II and learning automata and incorporates problem-specific genetic operators and learning automata to improve the behavior of the SPEA-II. In this algorithm, despite the SPEA-II, the rates of crossover and mutation operators ($p_c$ and $p_m$) don't have predefined and fixed values and are adapted dynamically. These operators can affect the performance of the algorithm. The great value for $p_c$ helps the algorithm to detect the new solutions and improves the convergence. Small value for $p_c$ causes the search process gets stuck at local optima. If $p_m$ is too large, the algorithm behaves very randomly. So, the balance between $p_c$ and $p_m$ should be maintained. In the proposed algorithm, a learning automaton performs this task by adjusting the value of these operators. In addition, despite the SPEA-II, the proposed algorithm uses Dynamic Crowding Distance (DCD) method to maintain the diversity of the solutions. The concept of DCD was proposed at the first time in [16]. The steps of the proposed algorithm are as follow:

Step 1: At first, the value of $t$ is initialized by zero and then, initial population (Population$_0$) with size $N$ and external archive (Population$_F$) with size $\bar{N}$ are created.

Step 2: For each individual $i$ in Population$_0$ and Population$_F$, a strength value $S(i)$ is calculated. The value of $S(i)$ indicates the number of solutions which dominated by solution $i$ (the concept of dominance was presented in section 3) and calculated as bellow:

$$S(i) = \left| \{ j \mid j \in \text{Population}_0 \cup \text{Population}_F \land i < j \} \right|$$

In this equation, $\cup$ means the union of two sets and $i < j$ means the solutions I dominated the solution $j$.

Then, according to the value of $S(i)$, the value of $P(i)$ is calculated as equation (6):

$$P(i) = \Sigma_{j \in \text{Population}_0 \cup \text{Population}_F, j < i} S(j)$$

As can be seen in this equation, the value of $P$ for non-dominated solutions is equal to zero, because there is no other solution like $j$ which dominates them. For better understanding this concept, we consider a minimization problem with two objectives and show the P value for dominated and non-dominated solutions in Fig. 1.
Step 3: For each individual \(i\) in Population, the value of Dynamic Crowding Distance \(DCD(i)\) is calculated according to (7):

\[
DCD_i = \frac{d_i}{\text{Var}(1/\text{Var}_i)}
\]

where \(d_i\) is the crowding distance of individual \(i\) and is calculated based on Equation (8) and demonstrates the closeness of individual \(i\) to another individuals in the population. The great value for this parameter results in better distributions of solutions in the population.

\[
d_i = \frac{1}{N_{\text{obj}}} \sum_{z=1}^{N_{\text{obj}}} |F_{i+1}^z - F_{i-1}^z|
\]

where \(N_{\text{obj}}\) is the number of objectives, \(F_{i+1}^z\) is the \(z\)th objective of the \(i+1\)th individual and \(F_{i-1}^z\) is the \(z\)th objective of the \(i-1\)th individual.

\[
\text{Var}_i = \frac{1}{N_{\text{obj}}} \sum_{z=1}^{N_{\text{obj}}} (F_{i+1}^z - F_{i-1}^z - d_i)^2
\]

The crowding distance of \(i\)th individual in a minimization problem with two objectives is shown in Fig. 2.

Step 4: For each individual \(i\) in Population, and \(\text{Population}_t\), the value of fitness function is calculated according to (10):

\[
f(i) = \alpha \left( \frac{1}{p(i)+1} \right) + (1 - \alpha) DCD(i)
\]

where \(p(i)\) and \(DCD(i)\) are calculated based on (6) and (7) respectively and \(\alpha\) is coefficient between 0 and 1 and is adjustable based on application requirement.

For avoiding the equation (10) to be infinite, the "+1" is put in this equation.

Step 5: The non-dominated individuals in Population, and \(\text{Population}_t\) are sorted ascending and then, the first \(N\) member are copied to \(\text{Population}_{t+1}\) (\(N\) is the size of external archive \(\text{Population}_t\)).

Step 6: If the algorithm is repeated for specific times \(T\) (\(t \geq T\)), the non-dominated individuals in \(\text{Population}_{t+1}\) are considered as Pareto-solutions of multi-objective optimization problems (outputs of the algorithm) and stop. Otherwise, after incrementing the \(t\) \((t=t+1)\) a new population \(\text{Population}_t\) is created from \(\text{Population}_{t+1}\) using tournament selection and crossover and mutation operators and go to Step 2. The rates of crossover and mutation operator in this step are calculated adaptively and dynamically using the learning automata as bellow procedure:

A fixed structure learning automaton is used. This automaton has two actions as (11):

Action 1: \(p_c = p_c + \alpha_1\), \(p_m = p_m - \alpha_2\).

Action 2: \(p_c = p_c - \alpha_1\), \(p_m = p_m + \alpha_2\).

The probability of selecting each action and the values for \(p_c\) and \(p_m\) are initialized as 0.5, 0.5, and 0.01 respectively. For prevent the exceeding of \(p_c\) and \(p_m\) from lower bound and upper bound values, the algorithm controls that \(p_m\) and \(p_c\) belongs to \([p_{m\text{LB}}, p_{m\text{UB}}]\) and \([p_{c\text{LB}}, p_{c\text{UB}}]\) respectively. At the end of each generation, learning automaton selects an action and based on the selected action, updates the values of \(p_c\) and \(p_m\). The algorithm runs with the new values for \(p_c\) and \(p_m\) and generates reward or penalty feedback for automaton as bellow:

Assume that \(r_{\text{childs}}\) is sum of the ranks of the two generated individuals after crossover and \(r_{\text{parents}}\) is sum of the ranks of the parent individuals before crossover. The differentiated rank of crossover (\(\text{diff}_{\text{cross}}\)) is calculated as follow:

\[
\text{diff}_{\text{cross}} = r_{\text{childs}} - r_{\text{parents}}
\]

If we have \(n_{\text{cross}}\) crossover operations in a generation, the average differentiated ranks value of crossover (\(\bar{\text{diff}}_{\text{cross}}\)) is calculated as follow:

\[
\bar{\text{diff}}_{\text{cross}} = \frac{\sum \text{diff}_{\text{cross}}}{n_{\text{cross}}}
\]

Thus, \(\bar{\text{diff}}_{\text{cross}}\) shows the overall impacts of crossover within current generation.
Similarly, the differentiated rank of mutation (\( \text{diff}_{\text{mut}} \)) is calculated as follow:

\[
\text{diff}_{\text{mut}} = \frac{r_{\text{mut}}^\text{child} - r_{\text{mut}}^\text{parent}}{n_{\text{mut}}}
\]  

(14)

where \( r_{\text{mut}}^\text{child} \) is the rank of new individual generated after mutation and \( r_{\text{mut}}^\text{parent} \) is the rank of original individual. If we have \( n_{\text{mut}} \) mutation operations in a generation, the average differentiated ranks of mutation (\( \bar{\text{diff}}_{\text{mut}} \)) is calculated as follow:

\[
\bar{\text{diff}}_{\text{mut}} = \frac{\sum \text{diff}_{\text{mut}}}{n_{\text{mut}}}
\]  

(15)

If (action 1 is selected by automaton and \( \bar{\text{diff}}_{\text{cross}} > \bar{\text{diff}}_{\text{mut}} \)) or (action 2 is selected by automaton and \( \bar{\text{diff}}_{\text{cross}} < \bar{\text{diff}}_{\text{mut}} \)), the selected action of automaton will be rewarded (positive feedback), otherwise, it will be penalized (negative feedback).

V. SIMULATION RESULTS

In order to evaluate the performance of the proposed algorithm, we consider convergence to the Pareto-optimal front and maintaining diversity among the non-dominated solutions as metrics, because these metrics are the main goals of any optimization algorithm [1].

The proposed algorithm is compared with some others evolutionary algorithms such as NSGA-II [9], NSGA-II a [17], SPEA-II [15] on different test functions. All of these algorithms are implemented using Matlab software and the same population size and initial values are considered for all of them. The used test functions are ZDT1, ZDT2, ZDT3 which are minimization problems [18]. These test functions and their specifications are introduced in Table 1.

Table 1: Multi-objective Test Problems [18]

<table>
<thead>
<tr>
<th>Problems</th>
<th>N</th>
<th>Bounds</th>
<th>Objective Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZDT1</td>
<td>30</td>
<td>[0,1]</td>
<td>( f_1=x_1; f_2=g(x)\left[1-\sqrt{x_1/g(x)}\right] ); ( g(x)=1+9\left(\sum_{i=2}^{n}x_i\right)/(n-1) )</td>
</tr>
<tr>
<td>ZDT2</td>
<td>30</td>
<td>[0,1]</td>
<td>( f_1=x_1; f_2=g(x)\left[1-x_1/g(x)\right]^2 ); ( g(x)=1+9\left(\sum_{i=2}^{n}x_i\right)/(n-1) )</td>
</tr>
<tr>
<td>ZDT3</td>
<td>30</td>
<td>[0,1]</td>
<td>( f_1=x_1; f_2=g(x)\left[1-\sqrt{x_1/g(x)}\right]^2 ); ( g(x)=1+9\left(\sum_{i=2}^{n}x_i\right)/(n-1) )</td>
</tr>
</tbody>
</table>

The parameters of the proposed algorithm and their initial values are presented in Table 2:

Table 2: The Simulation Parameters and Values

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T ) (Maximum number of generations)</td>
<td>300</td>
</tr>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>Archive size</td>
<td>30</td>
</tr>
<tr>
<td>Initial ( p_a ) value</td>
<td>0.01</td>
</tr>
<tr>
<td>( p_{al,b} ) (lower bound of mutation rate)</td>
<td>0.001</td>
</tr>
<tr>
<td>( p_{ub,b} ) (upper bound of mutation rate)</td>
<td>0.5</td>
</tr>
<tr>
<td>Initial ( p_c ) value</td>
<td>0.5</td>
</tr>
<tr>
<td>( p_{l,b} ) (lower bound of crossover rate)</td>
<td>0.1</td>
</tr>
<tr>
<td>( p_{u,b} ) (upper bound for crossover rate)</td>
<td>0.9</td>
</tr>
<tr>
<td>( a ) (changing step of crossover rate)</td>
<td>0.05</td>
</tr>
<tr>
<td>( \alpha ) (changing step of mutation rate)</td>
<td>0.001</td>
</tr>
<tr>
<td>( a, b ) (reward and penalty parameter)</td>
<td>0.1</td>
</tr>
<tr>
<td>( \alpha ) (Coefficient in fitness function)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

At first, we combine different objective functions into one object using weighted sum approach as follow and apply the classic genetic algorithm and obtain 50 non-dominated solutions as the reference Pareto-front.

Minimize \( F= w_1 f_1 + w_2 f_2 + w_3 f_3 \)  

(16)

In the above equation, \( w_1, w_2 \) and \( w_3 \) are weighting factor and their values vary between 0 and 1 in such a way that sum of them must be 1. Also, the values objective functions are normalized between 0 and 1.

As said before, we consider convergence and diversity as metric for evaluating the proposed algorithm. These metrics are defined as follow [1]:

Convergence means the average distance between the obtained non-dominated solutions and the reference Pareto-front and is calculated as bellow:

\[
Y = \frac{\left(\sum_{i=1}^{N} d_i\right)}{N}
\]  

(17)

In this equation, \( N \) is the number of non-dominated solutions obtained from the proposed algorithm and \( d_i \) is the minimum Euclidean distance of obtained non-dominated solutions \( i \) from the reference Pareto-front and is calculated as follow:

\[
d_i = \min_{k=1 \text{ to } 50} \sqrt{\sum_{m=1}^{M} (f_i^m - f_{m}^{(k)})^2}
\]  

(18)

In the above equation, \( M \) is the number of objectives, \( f_i^m \) is the value of objective \( m \) for the \( i^{th} \) member in the set of obtained non-dominated solutions and \( f_{m}^{(k)} \) is the value of objective \( m \) for the \( k^{th} \) solution in the reference Pareto-front.

Diversity is useful for evaluating diversity among the obtained non-dominated solutions, if an algorithm finds a
smaller Δ value, it is able to find a better diverse set of non-dominated solutions. This metric is defined as follow:

\[ \Delta = \sum_{i=1}^{N} (\hat{d} - d_i)^2 \tag{19} \]

In this equation, N is the number of non-dominated solutions obtained from the proposed algorithm and \( \hat{d} \) is the minimum Euclidean distance of solution i from other solutions in the set of obtained non-dominated solutions and is calculated as follow:

\[ d_i = \min_{j=1,2,...,N} \sum_{m=1}^{M} (f^i_m - f^j_m) \tag{20} \]

where \( f^i_m \) and \( f^j_m \) are the values of objective m for the \( i^{th} \) and \( j^{th} \) solutions. \( \hat{d} \) is the average of \( d_i \) and M is the number of objective.

Table 3 and Fig. 3 depict the mean and variance of the convergence metric \( \Upsilon \) for the NSGA-II, NSGA-II a, SPEA-II, and LASPEA after 10 independent experiments on ZDT1, ZDT2, and ZDT3 test problems. As can be seen in this table, the proposed algorithm has lower values for convergence metric which indicate better performance compared to the other algorithms. This is because the proposed algorithm incorporates DCD method and also tunes the rate of mutation operator adaptively using learning automata which improves the diversity metrics of the Pareto-solutions.

Table 4 and Fig. 4 depict the mean and variance of the diversity metric Δ for the NSGA-II, NSGA-II a, SPEA-II, and LASPEA after 10 independent experiments on ZDT1, ZDT2, and ZDT3 test problems. As can be seen in this table, the proposed algorithm has lower values for diversity metric which indicate better performance compared to the other algorithms. This is because the proposed algorithm incorporates DCD method and also tunes the rate of mutation operator adaptively using learning automata which improves the diversity metrics of the Pareto-solutions.

Table 4: The mean values of the diversity metric Δ

<table>
<thead>
<tr>
<th></th>
<th>ZDT1</th>
<th>ZDT2</th>
<th>ZDT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASPEA</td>
<td>0.1122</td>
<td>0.1219</td>
<td>0.4820</td>
</tr>
<tr>
<td>SPEA-II</td>
<td>0.7816</td>
<td>0.7154</td>
<td>0.6218</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>0.3903</td>
<td>0.4308</td>
<td>0.7385</td>
</tr>
<tr>
<td>NSGA-IIa</td>
<td>0.2814</td>
<td>0.4001</td>
<td>0.6217</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, by incorporating the learning automata and problem-specific genetic operators, an improved version of SPEA-II, called LASPEA, is proposed. The performance of the proposed algorithm is experimented on some benchmark multi-objective problems called ZDT1, ZDT2, and ZDT3. The experiment results show that the LASPEA not only can converge the non-dominated solutions to the Pareto-optimal front but also can enhance the solution diversity to spread the achieved extent for all multi-objective test problems.
REFERENCES


