ISSN 2047-3338

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

Seyed Mahdi Jameii[1], Mostafa Haghi Kashani[2] and Ramin Karimi[3]

***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.

[1]Seyed Mahdi Jameii is with the Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran, (Email: Jamei@Qodsiau.ac.ir)

[2]Mostafa Haghi Kashani is with the Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran, (Email: Kashani@Qodsiau.ac.ir)

[3]Ramin Karimi is with the Department of Computer Engineering, Malard Branch, Islamic Azad University, Tehran, Iran, (Email: rakarimi1@gmail.com)

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.

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]:

$$\text{Minimize or maximize } F(x) = (F_1(x), \ldots, F_m(x)) \quad (1)$$

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

$$\begin{cases} g_i(x) \le 0, & i = 1, \ldots, p \\ h_j(x) = 0, & j = p+1, \ldots, q \end{cases} \quad (2)$$

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 follow [1]:

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

$$\begin{cases} F_i(x) \le F_i(y), & \text{for every } i \in \{1, 2, \ldots, m\} \\ F_j(x) < F_j(y), & \text{for at least one index } j \in \{1, 2, \ldots, m\} \end{cases} \quad (3)$$

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).

$$PF = \{f(x) \mid x \in PS\} \quad (4)$$

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 ($m \ge 2$) 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 for maintaining 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:

Step1: At first, the value of t is initialized by zero and then, initial population ($Population_t$) with size N and external archive ($\overline{Population_t}$) with size $\bar{N}$ are created.

Step 2: For each individual i in $Population_t$ and $\overline{Population_t}$, 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:

$$S(i) = \left|\{j | j \in Population_t \cup \overline{Population_t} \wedge i \prec j\}\right| \quad (5)$$

In this equation, $\cup$ means the union of two sets and $i \prec j$ 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):

$$P(i) = \sum_{j \in Population_t \cup \overline{Population_t}, j \prec i} S(j) \quad (6)$$

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.

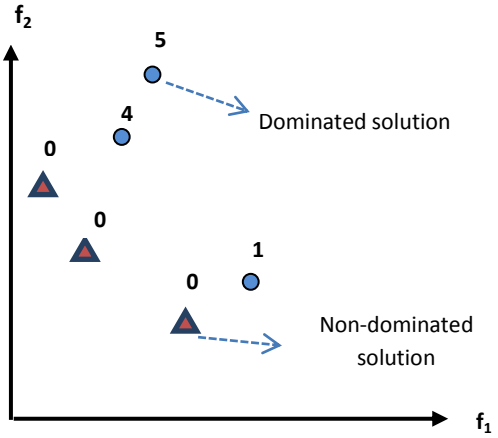

Fig. 1: The values of P for solutions in a minimization problem with two objectives

Step 3: For each individual i in Population$_t$ and $\overline{\text{Population}_t}$ the value of Dynamic Crowding Distance DCD(i) is calculated according to (7):

$$DCD_i = \frac{d_i}{\log(1/Var_i)} \quad (7)$$

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 other individuals in the population. The great value for this parameter results in better distributions of solutions in the population.

$$d_i = \frac{1}{N_{obj}} \sum_{z=1}^{N_{obj}} |F_{i+1}^z - F_{i-1}^z| \quad (8)$$

where N_{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.
Var_i is the variance of d_i of individuals which are neighbors of the i^{th} individual and is calculated based on Equation (9):

$$Var_i = \frac{1}{N_{obj}} \sum_{z=1}^{N_{obj}} (|F_{i+1}^z - F_{i-1}^z| - d_i)^2 \quad (9)$$

The crowding distance of i^{th} individual in a minimization problem with two objectives is shown in Fig. 2.

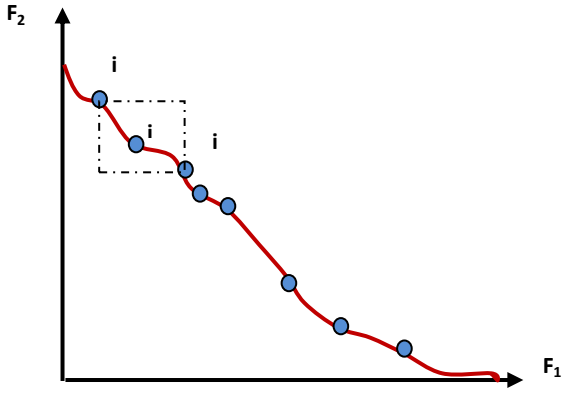

Fig. 2: Crowding distance of i^{th} individual in a minimization problem with two objectives [9]

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

$$f(i) = \alpha\left(\frac{1}{P(i)+1}\right) + (1-\alpha)DCD(i) \quad (10)$$

Where p(i) and DCD(i) are calculated based on (6) and (7) respectively and α 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$_t$ and $\overline{\text{Population}_t}$ are sorted ascending and then, the first $\bar{N}$ member are copied to $\overline{\text{Population}_{t+1}}$ ($\bar{N}$ is the size of external archive $\overline{\text{Population}_{t+1}}$).

Step 6: If the algorithm is repeated for specific times T (t≥T), the non-dominated individuals in $\overline{\text{Population}_{t+1}}$ 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 Population$_t$ is created from $\overline{\text{Population}_{t+1}}$ 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: $p_c = p_c + \alpha_1$, $p_m = p_m - \alpha_2$. (11)
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, o.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 $[p_{mLB}, p_{mUB}]$ and $[p_{cLB}, p_{cUB}]$ 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 $rank_{childs}$ is sum of the ranks of the two generated individuals after crossover and $rank_{parents}$ is sum of the ranks of the parent individuals before crossover. The differentiated rank of crossover ($diff_{cross}$) is calculated as follow:

$$diff_{cross} = rank_{childs} - rank_{parents} \quad (12)$$

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

$$\overline{diff}_{cross} = \frac{\sum diff_{cross}}{n_{cross}} \quad (13)$$

Thus, $\overline{diff}_{cross}$ shows the overall impacts of crossover within current generation.

Similarly, the differentiated rank of mutation ($diff_{mut}$) is calculated as follow:

$$diff_{mut} = rm_{child} - rm_{parent} \quad (14)$$

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

$$\overline{diff}_{mut} = \frac{\sum Gr_{mut}}{n_{mut}} \quad (15)$$

If (action 1 is selected by automaton and $\overline{diff}_{cross} > \overline{diff}_{mut}$) or (action 2 is selected by automaton and $\overline{diff}_{cross} < \overline{diff}_{mut}$), 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 ZDT_1 ، ZDT_2 و ZDT_3 which are minimization problems [18]. These test functions and their specifications are introduced in Table 1.

Table 1: Multi-objective Test Problems [18]

Problems	N	Bounds	Objective Functions
ZDT_1	30	[0,1]	$f_1=x_1$; $f_2=g(x)[1-\sqrt{x_1/g(x)}]$; $g(x)=1+9(\sum_{i=2}^{n} x_i)/(n-1)$
ZDT_2	30	[0,1]	$f_1=x_1$; $f_2=g(x)[1-(x_1/g(x))^2]$; $g(x)=1+9(\sum_{i=2}^{n} x_i)/(n-1)$
ZDT_3	30	[0,1]	$f_1=x_1$; $f_2=g(x)[1-\sqrt{x_1/g(x)}-(x_1/g(x))\sin(10\pi x_1)]$; $g(x)=1+9(\sum_{i=2}^{n} x_i)/(n-1)$

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

Table 2: The Simulation Parameters and Values

Parameters	Values
T (Maximum number of generations)	300
Population size	100
Archive size	30
Initial p_m value	0.01
p_{mLB} (lower bound of mutation rate)	0.001
p_{mUB} (upper bound of mutation rate)	0.5
Initial p_c value	0.5
p_{cLB} (lower bound of crossover rate)	0.1
p_{cUB} (upper bound for crossover rate)	0.9
α_1 (changing step of crossover rate)	0.05
α_2 (changing step of mutation rate)	0.001
a, b (reward and penalty parameter)	0.1
α (Coefficient in fitness function)	0.5

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.

$$\text{Minimize } F = w_1f_1 + w_2f_2 + w_3f_3 \quad (16)$$

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:

$$Y = (\sum_{i=1}^{N} d_i)/N \quad (17)$$

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:

$$d_i \min_{k=1\ to\ 50} \sqrt{\sum_{m=1}^{M} \left(f_m^i - f_m^{*(k)}\right)^2} \quad (18)$$

In the above equation, M is the number of objectives, f_m^i 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:

$$\Delta = \frac{\sqrt{\sum_{i=1}^{N}(\bar{d}-d_i)^2}}{N} \qquad (19)$$

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 solution i from other solutions in the set of obtained non-dominated solutions and is calculated as follow:

$$d_i = \min_{j=1,2,\ldots,N} \sum_{m=1}^{M} (f_m^i - f_m^j) \qquad (20)$$

where f_m^i and f_m^j are the values of objective m for the i^{th} and j^{th} solutions. $\bar{d}$ 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 *Y* for the NSGA-II, NSGA-II a, SPEA-II, and LASPEA after 10 independent experiments on ZDT_1, ZDT_2, and ZDT_3 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 problem-specific knowledge and also tunes the rate of crossover operator adaptively using learning automata which improves the convergence metric of the Pareto-solutions.

Table 3: The mean values of the convergence metric Y

	ZDT_1	ZDT_2	ZDT_3
LASPEA	0.0011	0.0009	0.0038
SPEA-II	0.0015	0.0113	0.0284
NSGA-II	0.0048	0.0317	0.0079
NSGA-IIa	0.0040	0.0103	0.0051

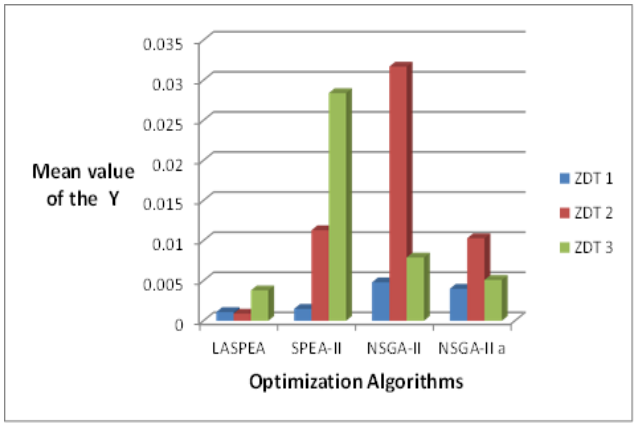

Fig. 3: The mean values of the convergence metric Y

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 ZDT 1, ZDT2, and ZDT 3 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

	ZDT_1	ZDT_2	ZDT_3
LASPEA	0.1122	0.1219	0.4820
SPEA-II	0.7816	0.7154	0.6218
NSGA-II	0.3903	0.4308	0.7385
NSGA-IIa	0.2814	0.4001	0.6217

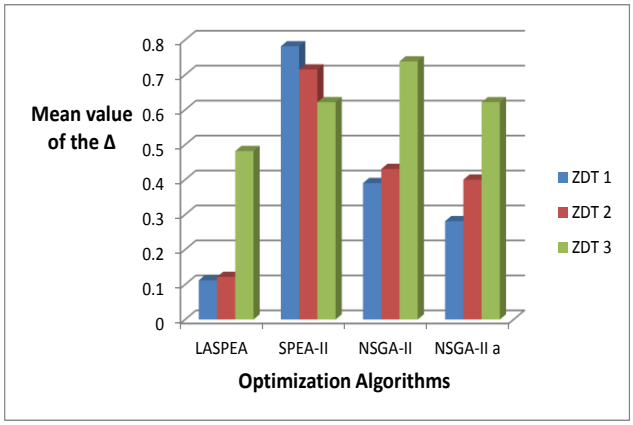

Fig. 4: The mean values of the diversity metric Δ

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 ZDT_1, ZDT_2, and ZDT_3. 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

[1]. K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley and Sons, New York, ISBN 047187339, 2002.

[2]. C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization," in Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416–423, 1993.

[3]. J. Horn, N. Nafploitis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in Proceedings of the First IEEE Conference on Evolutionary Computation, Z. Michalewicz,Ed. Piscataway, NJ: IEEE Press, pp. 82–87, 1994.

[4]. N. Srinivas and K. Deb, "Multiobjective function optimization using non-dominated sorting genetic algorithms," Evol. Comput., vol. 2, no.3, pp. 221–248, Fall 1995.

[5]. Zitzler, E. & Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, pp. 257-271, 1999.

[6]. Zhou, B. Qu, H. Li, S. Zhao, P. Nagaratnam, Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art", Swarm and Evolutionary Computation, Vol. 1, No. 1, pp. 32-49, 2011.

[7]. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, Second Edition, May 2002.

[8]. A.H.F. Dias, and J. A. Vasconcelos, Multiobjective genetic algorithms applied to solve optimization problems. IEEE Transactions on Magnetics, Vol. 38, No. 2, pp. 1133-1136, 2002.

[9]. Deb, K.; Pratap, A.; Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, 2002.

[10]. R. Qi, F. Qian, S. Li, and Z. Wang, Chaos-genetic algorithm for multiobjective optimization, Proceedings of the Sixth World Congress on Intelligent Control and Automation, pp. 1563-1566, 2006.

[11]. K. S. N. Ripon, S. Kwong, and K. F. Man, A real-coding jumping gene genetic algorithm (RJGGA) for multiobjective optimization. Information Sciences, Vol. 177, No. 2, pp. 632-654, 2007.

[12]. J. Knowles, and D. Corne, "M-PAES: A Memetic Algorithm for Multiobjective Optimization", Congress on Evolutionary Computation, pp. 325-332, Piscataway, New Jersey, July 2000.

[13]. Z. Wenqiang, S. Fujimura, Improved Vector Evaluated Genetic Algorithm with Archive for Solving Multiobjective PPS Problem, International Conference on E-Product E-Service and E-Entertainment (ICEEE 2010), China, Nov. 2010.

[14]. Q. Long, C. Wu, T. Huang, X. Wang, "A genetic algorithm for unconstrained multi-objective optimization", Swarm and Evolutionary Computation, Vol. 22, pp. 1-14, 2015.

[15]. E. Zitzler and M. Laumanns and L. Thiele, "SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm", Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich 2001.

[16]. S. Jeyadevi, S. Baskar, C. K. Babulal, M. W. Iruthayarajan, Solving multiobjective optimal reactive power dispatch using modified NSGA-II, Electrical Power and Energy Systems, Vol. 33, No. 2, pp. 219–228, 2011.

[17]. G. L. D Souza, K. Chandra Sekaran, and A. Kandasamy, Improved NSGA-II Based on a Novel Ranking Scheme, Journal of Computing, Vol. 2, No. 2, 2010.

[18]. K. k E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," Evolutionary Computation., Vol. 8, No. 2, pp. 173–195, 2000.