A New Recurring Multistage Evolutionary Algorithm for Solving Problems Efficiently

Md. Monirul Islam[1,2], Mohammad Shafiul Alam[3], and Kazuyuki Murase[2]

[1] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh
[2] Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan
[3] Department of Computer Science and Engineering, Ahasanullah Univeristy of Science and Technology, Bangladesh

Abstract. This paper introduces a new approach, called recurring multistage evolutionary algorithm (RMEA), to balance the explorative and exploitative features of the conventional evolutionary algorithm. Unlike most previous work, the basis of RMEA is repeated and alternated executions of two different stages i.e. exploration and exploitation during evolution. RMEA uses dissimilar information across the population and similar information within population neighbourhood in mutation operation for achieving global exploration and local exploitation, respectively. It is applied on two unimodal, two multimodal, one rotated multimodal and one composition functions. The experimental results indicated the effectiveness of using different object-oriented stages and their repeated alternation during evolution. The comparison of RMEA with other algorithms showed its superiority on complex problems.
Keywords: Evolutionary algorithm, exploration, exploitation and optimization problem.

1 Introduction

Evolutionary algorithms, such as evolution strategies (ES) [1], evolutionary programming (EP) [3, 4], and genetic algorithms (GAs) [6], have been widely used in global optimization problems that have a great importance in science, engineering and business fields. The basic difference between EP (or ES) and GAs is the evolutionary operator used in producing offspring. Although EP was first introduced as an approach to artificial intelligence, it was extended later and applied successfully to many practical and continuous parameter optimization problems.

The mutation is the main operator in EP [3, 4]. Thus a number of innovative mutation operators e.g. Cauchy mutation [14], a combination of Cauchy and Gaussian mutation [2] and Lévy mutation [8] have been proposed to improve the performance of EP. The aim of such mutations is to introduce large variations in evolving individuals so that they can explore a wider region of the search space globally. This means that the improvement has been sought by increasing the

exploration capability of EP, which is very much important for problems with many local optima. However, both global exploration and local exploitation are necessary during evolution depending on whether an evolutionary process gets stuck into local optima or finds some promising regions in the search space. It is, therefore, necessary for an algorithm to maintain a proper balance between exploration and exploitation in finding a good near-optima for complex problems.

This paper introduces a new recurring multistage evolutionary algorithm (RMEA) based on mutation. RMEA attempts to maintain a proper balance between exploration and exploitation by its two recurring stages. It uses object-oriented mutation operators and selection strategies in achieving exploration-exploitation objectives of the two stages. Although there are many algorithms that use GAs [6] for exploration and local methods for exploitation (see review paper [7]), RMEA is the first algorithm, to our best knowledge, that uses only mutation. Its emphasis on using different recurring stages can increase the solution quality of an evolutionary algorithm.

RMEA differs from most EP [4] based algorithms (e.g. [2], [8] and [14]), and memetic algorithms [11] on two aspects. First, RMEA emphasizes on achieving global exploration and local exploitation by executing object-oriented operations repeatedly and alternatively. This approach is different from the one used in EP [4] and memetic algorithms [11]. EP does not execute exploration and exploitation operations separately rather it uses single stage execution module with self-adaptation rules. Memetic algorithms generally executes exploration and exploitation operations one by one or they use some heuristics to decide when and where exploration and exploitation operations are to be executed within the evolutionary process. The difficulty of this approach lies in avoiding deep local optima and realizing the potentials of very promising regions or finding good heuristics.

Second, RMEA uses only mutation operators for achieving the exploration and exploitation objectives of an evolutionary process. It is argued in this paper that mutation is a better candidate than crossover for achieving exploration and exploitation objectives. For example, the global exploration and local exploitation can easily be achieved by using mutations with large and small step sizes, respectively. Memetic algorithms generally use GAs [6] for exploration and local search methods or specially designed crossover for exploitation. However, it would be difficult to find a proper integration method in order to achieve synergistic effect of different methods or operators.

The rest of this paper is organized as follows. Section 2 describes RMEA in detail and gives motivations and ideas behind various design choices. Section 3 presents experimental results of RMEA and their comparison with other work. Finally, section 4 concludes with a summary of the paper and a few remarks.

2 Recurring Multistage Evolutionary Algorithm

A recurring two-stage evolutionary approach based on mutation is adopted in RMEA to ensure a proper balance between global exploration and local exploita-

tion during evolution. The exploration and exploitation stages in RMEA use the distance of dissimilar and similar individuals, respectively, as the standard deviation for mutation. In fact, the motivation behind such an idea is inspired by observing the following important facts.

- Exploration is a non-local operation so mutation involving the distance of dissimilar individuals, which is expected to be large, may guide an evolutionary process toward the unexplored regions of a search space.
- Exploitation is a local operation so mutation involving the distance of similar individuals, which is expected to be small, may guide the evolutionary process toward the local neighborhoods.

The major steps of RMEA can be described as follows.

Step 1) Generate an initial population consisting of μ individuals. Each individual $\vec{x_i}$, $i = 1, 2, \ldots, \mu$, is represented as a real valued vector. It is consisted of n independent components

$$\vec{x_i} = \{x_i(1), x_i(2), \ldots x_i(n)\} \tag{1}$$

Step 2) Randomly initialize two parameters K_1 and K_2 within a certain range. These user specified parameters control the behavior of RMEA by defining how many generations the exploration and exploitation operations to be executed repeatedly during evolution.

Step 3) Calculate the fitness value of each individual $\vec{x_i}$, $i = 1, 2, \ldots, \mu$, in the population based on the objective function. If the best fitness value is acceptable or the maximum number of generations has been elapsed, stop the evolutionary process. Otherwise, continue.

Step 4) Repeat the following steps 5-8 for K_1 generations, which constitutes a single pass of the exploration stage.

Step 5) For each individual $\vec{x_i}$, $i = 1, 2, \ldots, \mu$, select ϕ individuals from the population in such a way that the fitness value of them is very different comparing to that of $\vec{x_i}$. The ϕ individuals, therefore, can be considered as *strangers* i.e., distant individuals for $\vec{x_i}$. Here the parameter ϕ is specified by the user.

Step 6) Create μ offspring by applying mutation on each individual $\vec{x_i}$, $i = 1, 2, \ldots, \mu$, of the population. Each individual $\vec{x_i}$ creates a single offspring $\vec{x_i}'$ by: for $j = 1$ to n

$$k = 1 + r_j \ mod \ n \tag{2}$$

$$x_i'(k) = x_i(k) + \sigma_i(k) N_j(0, 1) \tag{3}$$

Here r_j is a random number generated anew for each value of j. Its value can be from zero to any positive number. $x_i(k)$ and $x_i'(k)$ are the k-th component of $\vec{x_i}$ and $\vec{x_i}'$, respectively. $\sigma_i(k)$ is the standard deviation for mutating the

k-th component of $\vec{x_i}$. It is set as the average genotype distance between the k-th component of $\vec{x_i}$ and its ϕ strangers. $N_j(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one for each value of j.

The essence of using modulo operation is that it may produce n or less than n distinct values for k depending on r_j. This arises two different scenarios for mutation. First, mutation changes all components of $\vec{x_i}$ one time when there are n distinct values for k. Second, when there are less than n distinct values, mutation does not change all components of $\vec{x_i}$ rather it changes some components one time and some components few times.

Step 7) Compute the fitness value of each offspring $\vec{x_i}'$, $i = 1, 2, \ldots, \mu$. Select μ individuals from parents and offspring for the next generation. If the fitness value of $\vec{x_i}'$ is at least equal to its parent $\vec{x_i}$, discard $\vec{x_i}$ and select $\vec{x_i}'$ for the next generation. Otherwise, discard $\vec{x_i}'$.

Step 8) If the best fitness value is acceptable or the maximum number of generations has been elapsed, stop the evolutionary process. Otherwise, continue.

Step 9) Repeat the following steps 10-13 for K_2 generations, which constitutes a single pass of the exploitation stage.

Step 10) For each individual $\vec{x_i}$, $i = 1, 2, \ldots, \mu$, select ϕ individuals from the population in such a way that the fitness value of them is very similar comparing to that of $\vec{x_i}$. The ϕ individuals, therefore, can be considered as *neighbors* i.e., nearest individuals for $\vec{x_i}$.

Step 11) Create μ offspring in the same way as described in step 6. However, the genotype distance of neighbors are used here instead of strangers used in step 6.

Step 12) Compute the fitness value of each offspring $\vec{x_i}'$, $i = 1, 2, \ldots, \mu$. Select μ individuals from parents and offspring for the next generation. If the fitness value of $\vec{x_i}'$ is better than its parent $\vec{x_i}$, discard $\vec{x_i}$ and select $\vec{x_i}'$ for the next generation. Otherwise, discard $\vec{x_i}'$.

Step 13) If the best fitness value is acceptable or the maximum number of generations has been elapsed, stop the evolutionary process. Otherwise, go to Step 4.

It is seen that RMEA uses the genotype distance of individuals as the standard deviation for mutation. The advantage of such an approach is that it makes the mutation operation self adaptive without using any adaptation scheme. The individuals in a population are spread over the entire search space at the beginning of an evolutionary process. As the evolutionary processes progresses, the population converges toward the optimal solution and the genotype distance between individuals reduces. This means mutation will explore a wider region of the search space at the beginning and a smaller region at the end of the evolutionary process. The necessary details of different stages and components of RMEA are given in the following subsections.

2.1 Exploration Stage

This stage facilitates to explore the wider region of a search space so that the chance of finding a good near-optimum solution by an evolutionary process is increased. RMEA uses the average genotype distance between an individual $\vec{x_i}$, that is going to be mutated, and the other ϕ individuals in the population as the standard deviation for mutation. The ϕ individuals are selected in such a way that they are very different with respect to $\vec{x_i}$ based on the fitness value. Since the fitness value of $\vec{x_i}$ and the ϕ individuals is very different, it is expected that their genotypes are likely to be very different. The population diversity tends to rise for using the average genotype distance of very different individuals and allowing offspring that have same fitness values as their parents for the next generation.

2.2 Exploitation Stage

It is quite natural to realize the potentials of already explored regions before further explorations. RMEA, therefore, executes several exploitation operations after exploration operations. The aim of exploitation stage is to reach the peaks of different explored regions so that the optimum solution, if exist in any peak, can easily be achieved. Like exploration stage, the same mutation is also used here. However, it differs from exploration stage in the way that the average genotype distance between an individual $\vec{x_i}$, that is going to be mutated, and its neighbors, is used as the standard deviation for mutation. It is expected that the the genotype of ϕ individuals and $\vec{x_i}$ is very similar resulting a small average distance i.e., standard deviation. This is beneficial for exploiting the local neighborhood because mutation produces offspring around parents.

2.3 Recurring Approach

It is known that executing exploration and exploitation operations separately, and combining them in one algorithm is beneficial for improving the performance of evolutionary algorithms. A number of approaches have been proposed in the literature that use GAs [6] for exploration and local search methods or specialized crossover operators for exploitation. According to [7], the following four issues must be addressed when exploration and exploitation operations are executed separately. First, when and where a local search method should be applied within the evolutionary cycle. Second, which individuals in the population should be improved by local search, and how they should be chosen. Third, how much computational effort should be allocated to each local search. Fourth, how genetic operators can be best integrated with local search in order to get a synergistic effect. It is here worth mentioning that the aforementioned four questions need also to be addressed even when the same method or operator is used for both exploration and exploitation.

To address these questions, a number of heuristics and of user specified parameters need to employ in any classical evolutionary algorithm. The employment of many user specified parameters and heuristics requires a user to know

rich prior knowledge, which often does not exist for complex real-world problems. Furthermore, it reduces the autonomy of an evolutionary process thereby may guide the process in the wrong direction resulting poor performance. A scheme that does not employ many heuristics and user specified parameters is clearly preferable. The repeated and alternated execution of exploration and exploitation operations on all individuals in a population could be the simple solutions for the first three questions, which is adopted in our proposed approach RMEA. The execution of exploration and exploitation operations on all the individuals is not problematic if the operations can be done adaptively by using the same evolutionary operator. Since RMEA uses only mutation for both exploration and exploitation operations, the fourth question is not arisen here.

3 Experimental Studies

The aim our experimental studies is to evaluate the performance of RMEA and to observe the effect of separating the exploration and exploitation operations in an evolutionary approach based on mutation. Both RMEA and CEP [3] are applied on six benchmark test functions. These are unimodal (f_1 and f_2), unrotated multimodal (f_3 and f_4), rotated multimodal (f_5) and composition (f_6) functions. The method described in [12] is used here to create the rotated function. It left multiples the variable x in the original function by the orthogonal matrix $\mathbf{M}$ to get a corresponding new variable y of the rotated function. The composition function CF3 proposed in [9] is used here. It is constructed by combining ten f_3s. The following is the the analytical forms of the six functions.

1) Sphere function: $f_1(x) = \sum_{i=1}^{D} x_i^2$
2) Schwefel's function: $f_2(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$
3) Griewanks's function: $f_3(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
4) Rastergin's function: $f_6(x) = \sum_{i=1}^{D} \left[x_i^2 - 10cos(2\pi x_i) + 10\right]$
5) Rotated Griewanks's function:
$f_5(x) = \frac{1}{4000} \sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D} \cos\left(\frac{y_i}{\sqrt{i}}\right) + 1 \qquad y = \mathbf{M} * x$
6) Composition function (CF3[]):
$f_6(x) = \sum_{i=1}^{10} \{w_i * [f_3((x - o_{inew} + o_{iold})/\lambda_i * M_i) + bias_i\}] + f_{bias}$

A. Experimental Setup

It is seen from section 2 that three user specified parameters K_1, K_2 and ϕ are used in RMEA. Among them, K_1 and K_2 are most important in the sense that they controls the behavior of RMEA. Three different sets of value are used to investigate the effect of K_1 and K_2. They are 1 and 1, 2 and 4, and 4 and 8. The value of ϕ is set 3 for all functions. The value of different parameters in function f_6 was set same as used in [9]. CEP [3] is implemented in this work according to [5].

The population size μ was set 50 for both RMEA and CEP. The tournament size and the initial standard deviation used by CEP were set 5 and 3, respectively.

Table 1. Performance of RMEA with different values for K_1, K_2 and CEP [4] on six different functions. The number of function evaluation and the dimension of functions were set 150,000 and 30, respectively.

	Mean best result for function					
	f_1	f_2	f_3	f_4	f_5	f_6
RMEA(1,1)	5.50e-017	1.44e-014	6.89e-014	3.59e-005	5.49e-014	1.36e-008
RMEA(2,4)	1.05e-017	2.21e-015	**6.41e-020**	1.47e-007	**7.99e-017**	**1.39e-010**
RMEA(4,8)	**9.44e-018**	**1.96e-015**	8.90e-020	**1.21e-007**	9.10e-017	3.25e-010
CEP	9.14e-004	2.16e+002	8.73e-002	4.37e+001	8.45e+001	7.66e-004

The number of function evaluations (FEs) was set 150,000 for functions $f_1 - f_5$ and 50,000 for function f_6. The dimension of functions $f_1 - f_5$ was set 30, while it was set 10 for f_6. These values were chosen to make fair comparison with other work.

B. Results and Comparison

Table 1 shows the mean best result and standard deviation of RMEA and CEP on six functions over 50 independent runs. The numbers inside the parenthesis along RMEA indicate the values of K_1 and K_2 used in experiments. Fig. 1 shows the convergence characteristics of each function in terms of the mean best fitness.

It is clear that RMEA with different values for K_1 and K_2 performs much better than CEP [5]. The mean best fitness of RMEA with any value for K_1 and K_2 is better than CEP by several order magnitude. The *t*-test shows that the worst RMEA is significantly better than CEP for all six problems. The convergence characteristics of RMEA with different values for K_1,K_2 and CEP is similar at the very beginning of an evolutionary process (Fig. 1). As the evolutionary process progresses, the difference is very much clear. CEP appears to be trapped at the poor local optima or progresses very slowly. RMEA, on the other hand, successfully gets rid of local minima and progresses very aggressively toward a good near-optima. It is also clear from Table 1 that either RMEA(2,4) or RMEA(4,8) is better than RMEA(1,1). This indicates the necessity of executing exploration and exploitation operations repeatedly. The *t*-test shows that either RMEA(2,4) or RMEA(4,8) is significantly better than RMEA(1,1) for complex functions $f_3 - f_6$.

Tables 2 and 3 compare the performance of RMEA(2,4) with that of improved fast EP (IFEP) [14], adaptive EP with Lévy mutation (ALEP) [8] and real coded mematic algorithm (RCMA) with crossover hill climbing (XHC) [10]. To make the fair comparison, RMEA is reimplemented with the same number of function evaluation and problem dimension as used in RCMA with XHC. Like RMEA, both IFEP and ALEP use only mutation in producing offspring. IFEP mixes Cauchy and Gaussian mutations in one algorithm, while ALEP mixes Lévy mutations with four different distributions. RCMA with XHC executes exploration and exploitation operations separately. It uses PBX crossover [10] and

BGA mutation [13] for exploration and a specialized crossover operator XHC [10] for exploitation.

It is clear from Table 2 that the performance of RMEA is better than IFEP and ALEP on all four functions we compared here. RCMA with XHC outperforms RMEA on one unimodal function (Table 3). However RMEA outperforms RCMA with XHC with one unimodal and two multimodal functions (Table 3). Although we could not perform t-test, the better performance of RCMA with XHC and RMEA seems to be significant.

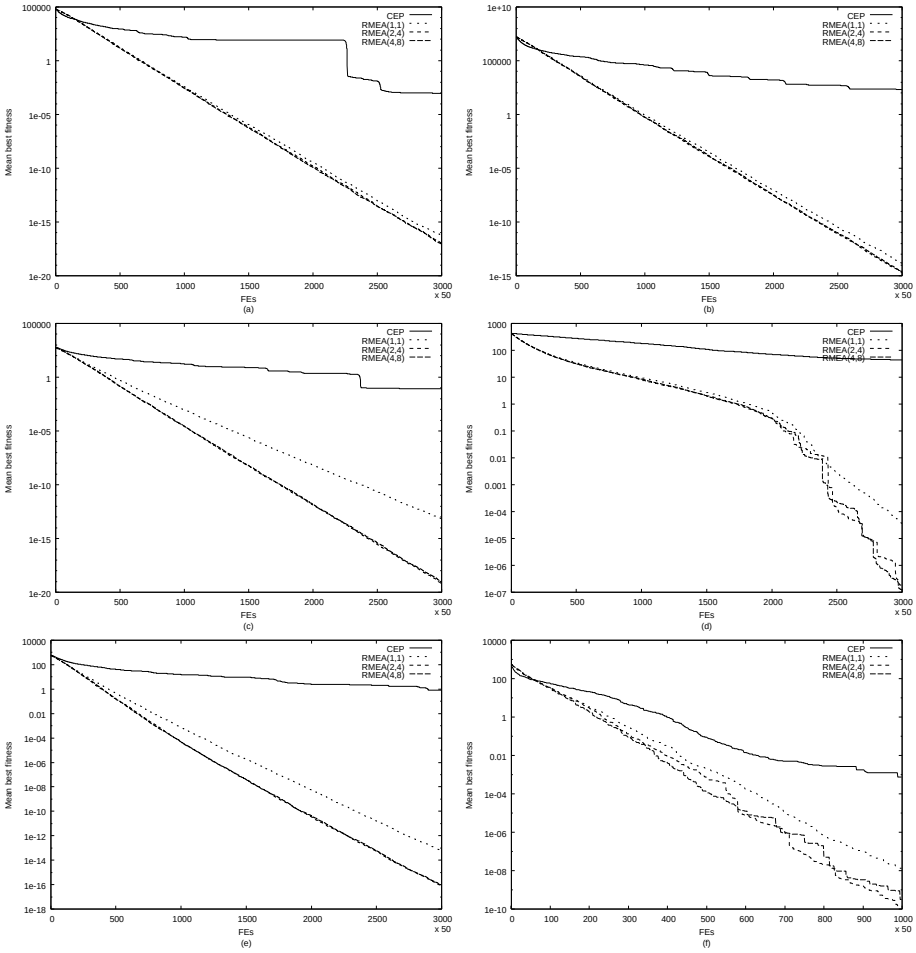

Fig. 1. Convergence characteristics of RMEA with different values for K_1, K_2 and CEP [4] on six functions: (a) f_1, (b)f_2, (c) f_3, (d) f_4, (e) f_5 and (f) f_6.

Table 2. Comparison among RMEA, IFEP [14] and ALEP [8] on four functions. All results have been averaged over 50 independent runs. The number of function evaluation and the dimension of functions were set 150,000 and 30, respectively.

	Mean best result for function			
	f_1	f_2	f_3	f_4
RMEA(2,4)	**1.05e-017**	**2.21e-015**	**6.41e-020**	**1.47e-007**
IFEP	4.16e-005	-	4.53e-002	-
ALEP	6.32e-004	4.18e-002	2.4e-002	5.85e+000

Table 3. Comparison between RMEA and RCMA with XHC [10] on four functions. All results have been averaged over 50 independent runs. The number of function evaluation and the dimension of functions were set 100,000 and 25, respectively.

Algorithm	Mean best result for function			
	f_1	f_2	f_3	f_4
RMEA(2,4)	1.25e-013	**1.63e-011**	**1.08e-015**	**9.86e-004**
RCMA with XHC	**6.50e-101**	3.80e-007	1.3e-002	1.4e+000

4 Conclusions

RMEA introduces a recurring multi-stage framework for evolutionary algorithms in order to unravel the conflicting goals of exploitation and exploration during evolution. It has demonstrated very promising results, outshining some other algorithms on complex problems. Such an inspiring performance by RMEA is quite reasonable, because RMEA employs quite a different mechanism than the others. While most algorithms seem to stagnate during evolution especially at the late generation, RMEA still continues optimization process at a graceful rate. In fact, RMEA achieves log-linear convergence rate for all the six tested functions. This is because the alternating and repeating stages in RMEA ensure better immunity from stagnation. The principle characteristics of RMEA are controlled by three user-specified parameters K_1, K_2 and ϕ. Future work on RMEA includes making these parameters self-adaptive taking into account their effects on both fitness and diversity.

Acknowledgement. MMI is currently a Visiting Associate Professor at University of Fukui supported by the Fellowship from Japanese Society for Promotion of Science (JSPS). This work was in part supported by grants to KM from JSPS, Yazaki Memorial Foundation for Science and Technology, and University of Fukui.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution Strategies: A Comprehensive Introduction. Natural Computing 1, 3-52 (2002)

2. Chellapilla, K.: Combining Mutation Operators in Evolutionary Programming. IEEE Transactions on Evolutionary Computation 2, 91-96 (1998)
3. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence Through Simulated Evolution. New York: Wiley (1966)
4. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. New York: IEEE Press (1995)
5. Gehlhaar, D. K., Fogel, D.B.: Tuning Evolutionary Programming for Conformationally Flexible Molecular Docking. In: Proc. of the Fifth Annual Conference on Evolutionary Programming, pp. 419429. Cambridge, MA: MIT Press (1996)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic Publishers, Boston (1989)
7. Krasnogor, N., Smith, J.E.: A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues. IEEE Transactions on Evolutionary Computation 9, 474- 488 (2005)
8. Lee, C., Yao, X.: Evolutionary Programming Using Mutations Based on the Lévy Probability Distribution. IEEE Transactions on Evolutionary Computation 8, 1-13 (2004)
9. Liang, J.J., Suganthan, P.N., Deb, K.: Novel Composition Test Functions for Numerical Global Optimization. In: Proc. of IEEE Swarm Intelligence Symposium, pp. 68-75 (2005)
10. Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded Memetic Algorithms with Crossover Hill-climbing. Evolutionary Computation 12, 273-302 (2004)
11. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program, C3P Report 826 (1989)
12. Salomon, R.: Reevaluating Genetic Algorithm Performance Under Coordinate Rotation of Benchmark Functions. BioSystems 39, 263-278 (1996)
13. Schlierkamp-Voosen, D., M$\ddot{u}$hlenbein, H.: Strategy Adaptation by Competing Subpopulations. In: Parallel Problem Solving from Nature 3, pp.199-208. Springer-Verlag, Berlin, Germany (1994)
14. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming Made Faster. IEEE Transactions on Evolutionary Computation 3, 82-102 (1999)