Index-based Genetic Algorithm for Continuous Optimization Problems

Ni Chen and Jun Zhang (Corresponding Author)
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Digital Life, Ministry of Education
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

ABSTRACT

Accelerating the convergence of Genetic Algorithms (GAs) is a significant and promising research direction of evolutionary computation. In this paper, a novel Index-based GA (termed IndexGA) is proposed for the acceleration of convergence by reducing the number of fitness evaluations (FEs) in the reproduction procedure, i.e. the process of crossover and mutation. The algorithm divides the solution space into multiple regions, each represented by a unique index. Individuals in the IndexGA are redefined as indexes instead of solutions. In the reproduction procedure, an evaluated region is never evaluated again, and the fitness is directly obtained from the memory. Moreover, to improve the fitness of the promising regions, the algorithm performs an orthogonal local search (OLS) operator on the best-so-far region in each generation. Numerical experiments have been conducted on 13 benchmark functions and an application problem of power electronic circuit (PEC) to investigate the performance of IndexGA. The results show that the index-based strategy and the OLS in IndexGA significantly enhance the performance of GAs in terms of both convergence rate and solution accuracy.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods.*

General Terms

Algorithms, Design, Experimentation

Keywords

Genetic algorithm, convergence acceleration, orthogonal local search

1. INTRODUCTION

Genetic algorithms (GAs) [1]-[3] are a class of evolutionary algorithms (EAs) for global optimization. A typical GA maintains a population of solutions in the search space. In each generation, the population undergoes the operations of selection, crossover, and mutation. Characterized by their robustness and the global convergence property, GAs have been applied in many fields like neural network training [4], image processing [5] and power electronics [6], etc.

However, as population-based optimization techniques, traditional GAs suffer from slow convergence. The algorithms consume large amounts of fitness evaluations (FEs) before converging to the global optimum.

To overcome the weakness of slow convergence, there are mainly two groups of GAs developed for enhancing the algorithms in terms of convergence rate. The first group of GAs is developed based on the fact that controlling the crossover probability *px* and mutation probability *pm* can benefit the search [7]. GAs in the first group control the two parameters adaptively by utilizing information from the fitness values [8], population distribution in search space [9], or a second-level GA [10]. However, these algorithms do not consider the improvement of solution refinement ability, and the acceleration of convergence is limited. The second group of GAs can be classified as the memetic algorithms (MAs) [11]-[13], which employ an additional local learning procedure [14]. The local learning procedure adopts three types of strategies, i.e. static [14], adaptive [15], and self-adaptive [16] strategies, according to the classification scheme in [11]. The static strategy uses the same local search operator during the whole search process, whereas the local search operators are altered with the feedback of algorithms in the adaptive and the self-adaptive strategies. Although the additional procedure succeeds in both improving the convergence rate and refining the solutions, the reproduction procedure of GAs, i.e. the crossover operator and the mutation operator, still consumes a large number of FEs.

In many real-world applications like power electronic circuit (PEC) optimization [6][9], the dimension of problem is low whereas the evaluation of fitness is expensive. These applications require the optimization algorithms to converge globally with fewest FEs. However, it is observed that the reproduction procedure of GAs usually reproduces similar individuals, especially on low-dimensional problems. Since the procedure focuses on global exploration instead of local exploitation, frequent evaluations of these similar individuals are a waste of the computational time.

The proposed IndexGA utilizes an index-based strategy to reduce the FEs in the reproduction procedure, especially on

GECCO'11, July 12–16, 2011, Dublin, Ireland.

low-dimensional problems with expensive evaluation functions. The algorithm divides the solution space into multiple regions, and assigns a unique index to each region. Individuals in the IndexGA are indexes instead of solutions, with each index representing a region in the solution space. The evaluation of an individual is defined to be the evaluation of a representative position in the corresponding region, and the algorithm memorizes the fitness of every evaluated region in a hash table. In the reproduction procedure, an evaluated region is never evaluated again, and the fitness is directly obtained from the memory. This way, the algorithm reduces the FEs in the reproduction process of GAs.

To improve the fitness of the promising regions, in each generation the algorithm performs an orthogonal local search (OLS) operator on the best-so-far region. Based on the orthogonal experimental design [19][20], the OLS views the process of local search as conducting experiments for a multifactor problem. According to an orthogonal array, the method first generates a set of solutions in the neighborhood for experiments. Then experiments are conducted by evaluating these solutions, and the experimental results are finally analyzed to estimate a promising position in the neighborhood. The estimated promising position is not necessarily included in the experiments. In addition, the proposed method adopts a strategy to adjust the step size of OLS adaptively.

In order to verify the effectiveness of the index-based strategy and the OLS in IndexGA, experiments are conducted on 13 benchmark functions and an application problem of power electronic circuit (PEC) [17]. The experimental results show that the proposed IndexGA is promising.

The rest of this paper is organized as follows. Section II describes the IndexGA in detail, followed by experiments on benchmark functions in Section III. In Section IV, the performance of IndexGA is further tested by application to a PEC design problem. Section V concludes the paper.

2. INDEX-BASED GA

2.1 Solution Space Division and Index Assignment

2.1.1 Solution Space Division

The IndexGA divides the solution space into multiple regions, and assigns each region an index. An index is defined as an integer that can be used as the unique representation of one region. In this paper, the strategies adopted for space division and index assignment are as follows.

For the division of D-dimensional continuous solution space with upper bounds $\boldsymbol{U}=[U^{(1)}, U^{(2)},\ldots, U^{(D)}]$ and lower bounds $\boldsymbol{L}=[L^{(1)}, L^{(1)},\ldots, L^{(D)}]$, the i-th dimension of the space is uniformly partitioned into R intervals $[L^{(i)}, L^{(i)}+(U^{(i)}-L^{(i)})/R]$,$[L^{(i)}+(U^{(i)}-L^{(i)})/R, L^{(i)}+2\times(U^{(i)}-L^{(i)})/R]$, …, $[L^{(i)}+(R-1)\times(U^{(i)}-L^{(i)})/R, U^{(i)}]$. Each region of the space is a hyper rectangle, which takes one of the R intervals on each dimension as its legal range for the corresponding variable. Thus, the divided solution space consists of R^D different regions, where R is the resolution of space partition.

According to the space division strategy, one region has the upper bound $L^{(i)}+(k^{(i)}+1)\times(U^{(i)}-L^{(i)})/R$ and the lower bound $L^{(i)}+k^{(i)}\times(U^{(i)}-L^{(i)})/R$ on the i-th dimension. Here we define the vector $[k^{(1)}, k^{(2)},\ldots, k^{(D)}]$ ($0\leqslant k^{(i)}<R$, i=1, 2, …, D) as the key vector of the region, which can be used to represent the region uniquely. An example of space division in 2-dimensional solution space and the key vectors for all the regions is illustrated in Fig. 1.

2.1.2 Index Assignment

In IndexGA, the index is utilized to identify a region. Since the fitness of each region is stored in a hash table, the index is also used as the key for hashing.

To assign every region an index, the key vector of the region is coded into an integer. For a key vector $K=[k^{(1)}, k^{(2)},\ldots, k^{(D)}]$, an integer n_K is calculated through the equation

$$n_K = \sum_{i=1}^{D} k_i \cdot R^{i-1} \tag{1}$$

where R is the resolution of space division. Take a key vector $Ke=[2, 1]$ of a region in Fig. 1 for example, when R=3, the corresponding index is $n_{Ke} = 2\times 3^0 + 1\times 3^1 = 5$. Note that two different regions in the solution space will never share the same index according to the above coding strategy.

2.2 Evolutionary Process of IndexGA

The traditional real-coded GAs maintain a population of N individuals, i.e. N positions in the D-dimensional solution space. In each generation, the operations of selection and reproduction, namely crossover and mutation, are performed on the population. The algorithm evaluates every individual reproduced in the crossover and mutation operators.

Differently, in IndexGA, the individuals in a population

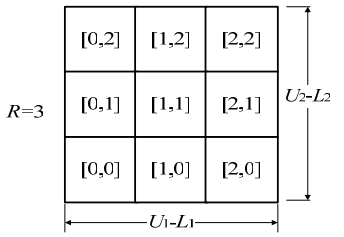

Figure 1. Regions in a 2-dimensional space when resolution R=3.

are redefined to be indexes instead of solutions. A population is comprised of N indexes, with each representing a region in the solution space. The evaluation of an individual is defined as the evaluation of a representative position in the corresponding region. The representative position together with the fitness of every evaluated region is memorized in a hash table. In the reproduction procedure,

a reproduced individual is evaluated when and only when the region has never been evaluated. For evaluated regions, the fitness values are directly obtained from the memory. To improve the fitness of the promising regions, in each generation the algorithm performs an OLS operator on the best-so-far region.

The flowchart of IndexGA is given in Fig. 2. According to the figure, the basic flow of IndexGA is the same as that

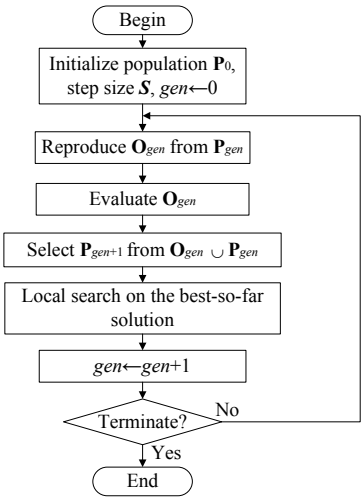

Figure 2. Basic flow of IndexGA.

of traditional GA, except that in IndexGA an additional procedure of local search is performed. The flow of IndexGA is described in detail as follows.

Step 1) Initialization: N different regions are randomly picked from the solution space to form the initial population $\mathbf{P}_0$, where N is the size of population. For each region, the algorithm calculates its index, randomly generates one position in the region as its representative position, and evaluates the region. The fitness and the representative position of the region are memorized in a hash table.

Moreover, the step size of local search is initialized to be

$$\boldsymbol{S}=(\boldsymbol{U}-\boldsymbol{L})/R, \tag{2}$$

where $\boldsymbol{U}$ and $\boldsymbol{L}$ are the upper and lower bounds of the solution space, and $\boldsymbol{R}$ is the resolution of space division. The counter of generation is set as gen=0.

Step 2) Reproduction: The algorithm reproduces N offspring through crossover and mutation to form $\mathbf{O}_{\text{gen}}$ in the following way. a) Two individuals are selected for reproduction from $\mathbf{P}_{gen}$ using a binary tournament strategy. Reselect if two identical indexes are selected. This way, crossover will never happen between identical regions. b) Two offspring are reproduced through crossover and mutation on the two selected regions. The crossover operator is performed with a probability of px, and the mutation operator is performed with a probability of pm. When a region is reproduced, the algorithm immediately calculates its index. c) Terminate if N offspring are generated, otherwise go back to a).

Note that although the individuals in IndexGA are indexes instead of solutions, the reproduction operators of traditional real-coded GAs can still be utilized. The crossover and mutation operators are performed on the representative positions in the regions, thus traditional operators like blend crossover (BLX) [21] and uniform mutation for real-valued individuals can be directly applied.

Step 3) Evaluation: For each of the individuals in $\mathbf{O}_{gen}$, the region is evaluated if and only if it has never been evaluated. Since the fitness evaluated regions are memorized in a hash table, the fitness values of these regions in $\mathbf{O}_{gen}$ can be immediately obtained from the memory.

Step 4) Selection: According to the fitness values, N regions are selected from $\mathbf{P}_{gen} \cup \mathbf{O}_{gen}$ to comprise $\mathbf{P}_{gen+1}$.

Table 1. An Example of Multifactor Problem

	Factors		
	Temperature	Irrigation	Amount of fertilizer
Levels	20℃	10mm	100g/m^2
	25℃	20mm	150g/m^2
	30℃	30mm	200g/m^2

Table 2. An Multifactor Problem of Local Search

	Factors		
	1st dimension	2nd dimension	3rd dimension
Levels	+	+	+
	/	/	/
	−	−	−

Table 3. Orthogonal Experimental Design and the Experimental Results for the Problem in TABLE II

No. of Experiment	Factor			Fitness
	1st dimension	2nd dimension	3rd dimension	
1	−	−	−	10
2	−	/	−	15
3	−	+	+	30
4	/	−	−	15
5	/	/	+	30
6	/	+	−	25
7	+	−	+	30
8	+	/	−	25
9	+	+	/	35
F_+	90	90	90	
$F_/$	70	70	70	
F_-	55	55	55	
Estimated Best Combination	+	+	+	

The selection operator employed in IndexGA adopts elitism and never selects duplicates. Traditional selection operators like tournament selection can be modified in the following way to satisfy the above requirement. a) The best-so-far region is selected in the first place. b) After a region with index n is selected, all the regions in $\mathbf{P}_{gen} \cup \mathbf{O}_{gen}$ that share the same index n are removed from the previous population.

Step 5) Local search: Since a region will never be reevaluated in the reproduction procedure, the fitness of a

region never improves if local search is not performed. In IndexGA, the OLS is applied to the best-so-far region for fitness improvement. The method is based on orthogonal experimental design, and its detailed description will be presented in the following part.

Step 6) Termination Check: If the number of FEs exceeds the predefined maximum number, the algorithm terminates. Otherwise, increase *gen* by 1, go back to step 2) and start a new generation.

The orthogonal local search (OLS) is a local learning strategy for the fitness improvement of a region. Based on the orthogonal experimental design [19][20], the method considers the local search in a neighborhood to be a multifactor problem. The OLS first generates a set of solutions for experiments according to an orthogonal array. Then the procedure conducts experiments by evaluating these solutions, and finally analyzes the experimental results to estimate a promising position in the neighborhood.

2.3 OLS

2.3.1 Multifactor Problems

The orthogonal experimental design technique is a method for experiment planning on multifactor problems [19][20]. The multifactor problems involve multiple factors and multiple levels for each factor. To solve these problems, experiments are conducted to find the best combination of levels. An example of multifactor problem in agriculture is given in Table I, which shows three factors, i.e. temperature, irrigation and fertilizer, and three levels for each factor. To improve the yield of crops, experiments are conducted to find a best combination of levels for the factors. Since there are totally combinations, a maximum number of 3×3×3=27 experiments are required.

The local search in the neighborhood of a solution can be considered as a multifactor problem, an example of which is given in Table II. In Table II, each dimension of the 3-dimensional search space is a factor, the increased, unchanged and decreased values (denoted as '+', '/' and '−', respectively) of the variable on the corresponding dimension are considered as three levels. To find the combination of levels with best fitness, a maximum number of 27 experiments (i.e. 27 evaluations) are required.

2.3.2 Orthogonal Experimental Design

For a multifactor problem with N factors and Q levels, the total number of combinations Q^N grows exponentially with N. Thus, experimenting on all the combinations becomes infeasible when N is large.

The orthogonal experimental design technique is introduced to plan the experiments for multifactor problems. Based on the orthogonal array [19][20], the method designs M ($M<Q^N$) experiments to represent all the Q^N combinations. Moreover, it analyzes the experimental results to estimate the best combination. The estimated best combination is not necessarily included in the experiments. For more detailed information on orthogonal array and orthogonal experimental design, interested readers can refer to [19][20].

As an example, Table III presents the experiments designed for the problem in Table II according to an orthogonal array. The geometric interpretation of Table III is illustrated in Fig. 3. In Table III, 9 instead of all the 27 experiments are planned. For each experiment, the symbol '+', '−' and '/' denotes the three levels, i.e. the increased, decreased and unchanged values of the variable on the corresponding dimension, respectively. It can be observed from Table III and Fig. 3 that the designed experiments satisfy the following rules. a) For any two factors, each combination of levels appears the same number of times. b) For any factor, each level appears the same number of times. Therefore, experiments conducted according to Table III can well represent all the possible combinations of factors.

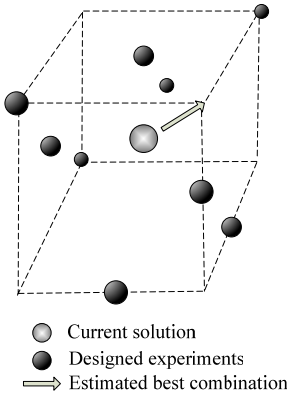

Figure 3. Geometric interpretation of TABLE III.

In Table III, the results for experiments, i.e. the fitness values, are given in the column 'Fitness'. The rows '$\boldsymbol{F}_+$', '$\boldsymbol{F}_-$', and '$\boldsymbol{F}_/$' calculate the total fitness when the corresponding factor takes the levels '+', '−' and '/', respectively. For example, $\boldsymbol{F}_+$ on the first dimension is the total fitness of the 7-th, 8-th, and 9-th experiments.

The estimated best combination of factors is presented in the row 'Estimated Best Combination', which is chosen based on the rows '$\boldsymbol{F}_+$', '$\boldsymbol{F}_-$', and '$\boldsymbol{F}_/$'. For each factor, the level with the best F-value is chosen. In this example, the best combination is the level '+' for all the factors, which is not included in the 9 experiments.

2.3.3 Procedure of OLS

The OLS can be considered as conducting experiments for a multifactor problem and analyzing the results for the best combination. The procedure is described in detail as follows.

a) The representative position of the best-so-far region is denoted as $\boldsymbol{X}=[x^{(1)}, x^{(2)}, \ldots, x^{(D)}]$. Generate a set of m positions $\boldsymbol{E}_1, \boldsymbol{E}_2, \ldots, \boldsymbol{E}_m$ in the solution space according to the orthogonal array $\mathbf{T}$. To generate the k-th solution $\boldsymbol{E}_k$, for each '+', '−' and '/' in (i=1, 2, …, D), set the value of $x^{(i)}$ to be $x^{(i)}+|N(0, s^{(i)})|$, $x^{(i)}-|N(0, s^{(i)})|$, and $x^{(i)}$, respectively. Here

$\boldsymbol{S}$=[$s^{(1)}$, $s^{(2)}$, …, $s^{(D)}$] is the step size for local search, and $N(0, s^{(i)})$ is a real number generated according to normal distribution with mean value of 0 and standard deviation of $s^{(i)}$.

b) To estimate a promising position in the neighborhood, evaluate $\boldsymbol{E}_1$, $\boldsymbol{E}_2$, … $\boldsymbol{E}_m$ to obtain the fitness values $f_1, f_2,\ldots, f_m$. Define three D-dimensional vectors '$\boldsymbol{F}_+$', '$\boldsymbol{F}_-$', and '$\boldsymbol{F}_/$' according to the equations.

$$F_{+}^{(j)} = \sum_{\mathrm{T}ij='+'}^{i} f_j \ (j = 1,2,...,D) \tag{3}$$

$$F_{-}^{(j)} = \sum_{\mathrm{T}ij='-'}^{i} f_j \ (j = 1,2,...,D) \tag{4}$$

$$F_{/}^{(j)} = \sum_{\mathrm{T}ij='/'}^{i} f_j \ (j = 1,2,...,D) \tag{5}$$

The vectors '$\boldsymbol{F}_+$', '$\boldsymbol{F}_-$', and '$\boldsymbol{F}_/$' reflect whether the increased, decreased, and unchanged values of the variables on different dimensions are promising, respectively. For example, a larger value of than and encourages an increase of $x^{(i)}$, whereas a larger value of than and encourages a decrease of $x^{(i)}$.

c) With the values of '$\boldsymbol{F}_+$', '$\boldsymbol{F}_-$', and '$\boldsymbol{F}_/$', generate a new solution ***EBest*** according to

$$Ebest^{(i)} = \begin{cases} x^{(i)} - |N(0,s^{(i)})| & F_{-}^{(i)} > F_{+}^{(i)} \ \textbf{\textit{and}} \ F_{-}^{(i)} > F_{/}^{(i)} \\ x^{(i)} + |N(0,s^{(i)})| & F_{+}^{(i)} > F_{-}^{(i)} \ \textbf{\textit{and}} \ F_{+}^{(i)} > F_{/}^{(i)} \\ x^{(i)} & \textbf{\textit{Otherwise}} \end{cases} \tag{6}$$

d) Update the fitness values and the representative position of the corresponding regions using $\boldsymbol{E}_1$, $\boldsymbol{E}_2$,…, $\boldsymbol{E}_m$ and ***EBest***.

e) Update the step size S. If any one of $\boldsymbol{E}_1$, $\boldsymbol{E}_2$,…, $\boldsymbol{E}_m$ and ***EBest*** improves the fitness of the best-so-far region, the orthogonal local search is considered to be a success, and the step size $\boldsymbol{S}$ is expended by multiplying a predefined real number *EXP* (*EXP*>1.0). Otherwise, the local search fails, and the step size S shrinks by multiplying a predefined real number *SHR* (*SHR*<1.0).

3. EXPERIMENTS ON BENCHMARKS

3.1 Benchmarks and Exerimental Configurations

Experiments are conducted on both unimodal and multimodal benchmark functions to study the performance of the proposed IndexGA. The 13 test functions, which are given in Table IV, are all minimum problems taken from the work of Yao *et al*. [22]. Among these functions, f_1-f_7 are unimodal functions, and f_8-f_{12} are multimodal functions with local optima. In Table IV, the column 'ε' presents the acceptable error, and only when the error of a solution reaches within $[0, \varepsilon]$ is the run considered successful. The minimum values of the functions are given in the column 'Opt', and the upper and lower bounds are in the column 'domain'. Since all the 13 functions are dimensionwise scalable, both 2-dimensional and 4-dimensional functions are tested in the following experiments.

In the experiments, three algorithms are tested, including the traditional real-coded GA (rGA), real-coded GA with OLS (rGA+OLS), and the proposed IndexGA. The rGA+OLS is a traditional GA that uses the same OLS as in IndexGA, and is tested to study the effect of the region-based strategy of IndexGA. Each test is run 30 independent trials and a maximum of 8×10^4 FEs is limited.

Table 4. Test Functions

	ε	Functions	Opt	Domain
Unimodal	10^{-12}	$f_1 = \sum_{i=1}^{D} x_i^2$	0	$[-100,100]^D$
	10^{-8}	$f_2 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	0	$[-10,10]^D$
	10^{-8}	$f_3 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	0	$[-100,100]^D$
	10^{-8}	$f_4 = \max_i(\lvert x_i \rvert, \ 1 \le i \le D)$	0	$[-100,100]^D$
	10^{-8}	$f_5 = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	0	$[-30,30]^D$
	10^{-2}	$f_6 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	0	$[-100,100]^D$
	10^{-2}	$f_7 = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	0	$[-1.28,1.28]^D$
Multimodal	10^{-2}	$f_8 = -\sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$-418.983D$	$[-500,500]^D$
	10^{-3}	$f_9 = \sum_{i=1}^{D} [(x_i)^2 - 10\cos(2\pi x_i) + 10]$	0	$[-5.12,5.12]^D$
	10^{-2}	$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i)^2}\right) - \left(\exp\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	0	$[-32,32]^D$
	10^{-2}	$f_{11} = 1 + \sum_{i=1}^{D} x_i^2 / 4000 - \prod_{i=1}^{D} \cos(x_i / \sqrt{i})$	0	$[-600,600]^D$
	10^{-3}	$f_{12} = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{j=1}^{D-1}(y_j - 1)^2[1 + 10\sin^2(\pi y_{j+1}) + (y_D - 1)^2]\right\} * + \sum_{i=1}^{D} u(x_j, 10, 100, 4)$	0	$[-50,50]^D$
	10^{-3}	$f_{13} = \frac{1}{10}\left\{10\sin^2(3\pi x_1) + \sum_{j=1}^{D-1}(x_j - 1)^2[1 + \sin^2(3\pi x_{j+1})] + (x_D - 1)^2\left(1 + \sin^2(2\pi x_D)\right)\right\} + \sum_{i=1}^{D} u(x_j, 5, 100, 4)$ *	0	$[-50,50]^D$

Table 5. Results Comparisons of the Three Test Algorithms on 2-dimensional Functions

	rGA			rGA+OLS			IndexGA		
	Avg.	Std dev.	Eval	Avg.	Std dev.	Eval	Avg.	Std dev.	Eval
f_1	0.000220641	0.000744573	--	8.9815×10^{-14}	4.83653×10^{-13}	1351	**0**	**0**	**1203**
f_2	0.000595222	0.000541483	--	1.89834×10^{-13}	1.01882×10^{-12}	1513	**0**	**0**	**1347**
f_3	0.00262603	0.00255895	--	4.20726×10^{-31}	1.07264×10^{-30}	1196	**0**	**0**	**995**
f_4	0.0106169	0.00699289	--	4.3327×10^{-17}	2.33323×10^{-16}	1961	**0**	**0**	**1496**
f_5	0.757738	1.03022	--	7.65226×10^{-6}	2.97917×10^{-5}	36669	$\mathbf{6.82975\times10^{-27}}$	$\mathbf{1.7529\times10^{-27}}$	**16748**
f_6	**0**	**0**	1215	**0**	**0**	**179**	**0**	**0**	267
f_7	0.000523491	0.00050037	1487	0.000634287	0.000545264	3158	**0.0012897**	**0.00149373**	**432**
f_8	-837.965	0.00111909	1264	-837.966	1.227×10^{-13}	1080	**-837.966**	$\mathbf{8.03887\times10^{-14}}$	**940**
f_9	2.7482×10^{-5}	5.5309×10^{-5}	13952	5.921×10^{-17}	3.188×10^{-16}	3223	**0**	**0**	**919**
f_{10}	0.00608043	0.00725876	44783	1.522×10^{-7}	7.504×10^{-7}	**729**	$\mathbf{8.25569\times10^{-16}}$	$\mathbf{8.86202\times10^{-16}}$	775
f_{11}	0.00439234	0.00368851	31021	**0.00049307**	**0.0018449**	3427	0.00641089	0.00369072	**1355**
f_{12}	4.55362e-005	0.000119837	17219	9.62248×10^{-20}	3.04873×10^{-19}	1579	$\mathbf{2.35566\times10^{-31}}$	$\mathbf{8.75812\times10^{-47}}$	**628**
f_{13}	9.65714e-005	0.000297685	16742	5.57078×10^{-13}	2.99996×10^{-12}	665	$\mathbf{1.34969\times10^{-32}}$	$\mathbf{5.47382\times10^{-48}}$	**650**

Table 6. Results Comparisons of the Three Test Algorithms on 4-dimensional Functions

	rGA			rGA+OLS			IndexGA		
	Avg.	Std dev.	Eval	Avg.	Std dev.	Eval	Avg.	Std dev.	Eval
f_1	9.21302×10^{-5}	0.000137975	--	**0**	**0**	**2287**	**0**	**0**	2404
f_2	0.00117603	0.000818223	--	1.52009×10^{-266}	**0**	2584	**0**	**0**	**2487**
f_3	0.232851	0.166609	--	2.52435×10^{-30}	2.86897×10^{-30}	2329	**0**	**0**	**2292**
f_4	0.0349781	0.0204387	--	1.58595×10^{-225}	**0**	3246	**0**	**0**	**3068**
f_5	1.47119	1.4443	--	0.357517	0.68861	--	**0.00104904**	**0.00564849**	**72790**
f_6	**0**	**0**	2603	**0**	**0**	**564**	**0**	**0**	796
f_7	0.00128997	0.00119848	7189	0.00130867	0.00116896	10355	**0.000285295**	**0.00021327**	**1184**
f_8	-1675.93	0.000429359	25510	**-1675.93**	2.87607×10^{-13}	**2088**	**-1675.93**	$\mathbf{1.76123\times10^{-13}}$	2556
f_9	0.000109476	0.000164333	28519	8.87442×10^{-11}	4.77902×10^{-10}	5175	**0**	**0**	**3073**
f_{10}	0.00844547	0.00607087	28290	$\mathbf{3.43089\times10^{-15}}$	$\mathbf{1.92416\times10^{-15}}$	**1473**	$\mathbf{3.43089\times10^{-15}}$	2.13163×10^{-15}	1602
f_{11}	0.0166971	0.00829507	40590	**0.0121566**	0.00778985	33203	0.0133832	**0.00726012**	**18525**
f_{12}	6.9769×10^{-5}	0.000143477	13338	$\mathbf{1.17783\times10^{-31}}$	$\mathbf{4.37906\times10^{-47}}$	1981	1.21655×10^{-31}	1.16169×10^{-32}	**1438**
f_{13}	5.46616×10^{-5}	9.88089×10^{-5}	15399	$\mathbf{1.34969\times10^{-32}}$	$\mathbf{5.47382\times10^{-48}}$	**1389**	1.43187×10^{-32}	2.23535×10^{-33}	1561

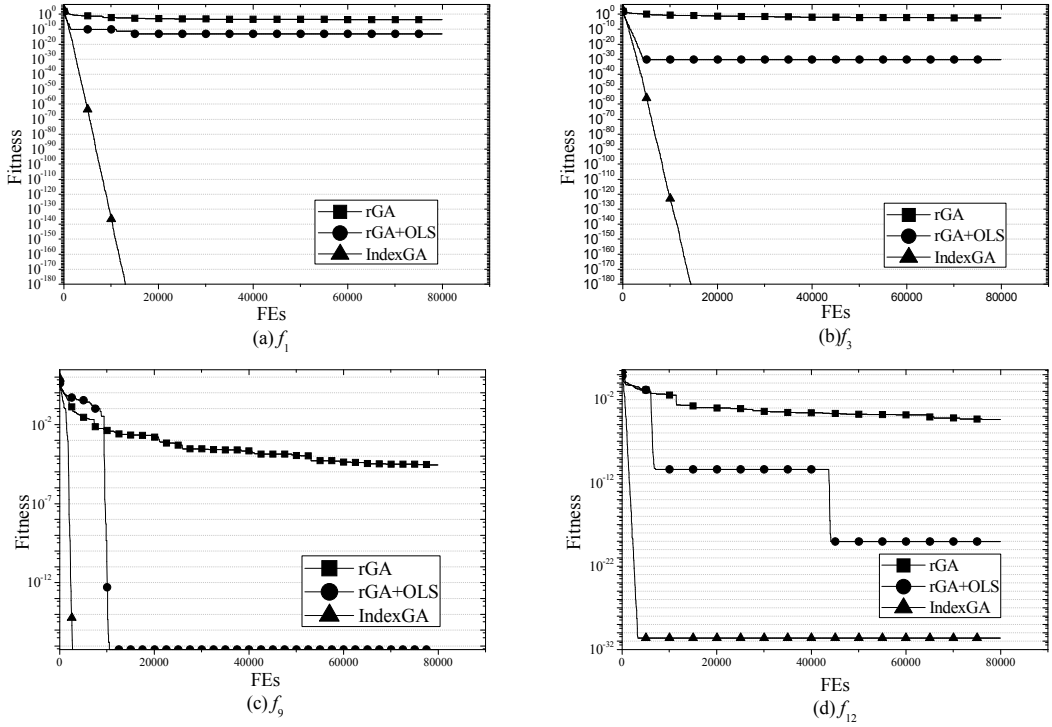

Figure 4. Evolutionary curves of the three tested algorithms on 2-dimensional functions.

rGA+OLS, and the IndexGA: the size of population N, the crossover probability px, and the mutation probability pm. For these parameters, the settings are as follows: N=50, px=0.7, pm=0.01. The OLS in rGA+OLS and IndexGA involves three parameters: the initial step size $\boldsymbol{S}_0$, the expending rate EXP and the shrinking rate SHR for step size adjustment. These parameters are set as $\boldsymbol{S}_0 = (\boldsymbol{U}-\boldsymbol{L})/R$, $EXP=(0.85)^{-1}$ and SHR=0.7, where $\boldsymbol{U}$ and $\boldsymbol{L}$ are the upper and lower bounds of solution space, and R is the resolution of space division.Specially for IndexGA, one more parameter, i.e. the resolution R for space division, is introduced. In the experiments, the resolution R is set as R=80 for unimodal functions f_1-f_7, and R=1600 for multimodal functions f_8-f_{13}.

3.2 Results and Discussions

The experimental results for the rGA, rGA+OLS, and the proposed IndexGA on 13 test functions are reported in Table V-VI, with Table V containing results on 2-dimensional functions and Table VI presenting results on 4-dimensional functions. In Table V-VI, the results are presented in three columns for each algorithm. The column 'Avg.' contains the mean value of solutions averaged over 30 independent runs, and the column 'Std dev.' reports the standard deviation of the solutions obtained. The column 'Eval' presents the average FEs needed to reach within the errors defined in Table IV. In each of the three columns, the best results among the three algorithms are marked in **boldface**.

3.2.1 Comparison on Solution Accuracy

According to Table V-VI, the traditional GA suffers from difficulty in solution refinement. In contrast, both rGA+OLS and IndexGA improve the rGA significantly in terms of mean results and standard deviation. Since rGA+OLS and IndexGA both employ the OLS strategy for local search, the results validate the effectiveness of OLS.

For the comparison between rGA+OLS and IndexGA, in general, the IndexGA uses less FEs to reach the errors than rGAs+OLS on most of the 2-dimensional functions (f_1-f_5, f_7-f_{10}, f_{12}, f_{13}) and the 4-dimensional functions (f_1-f_5, f_7-f_9). The results confirm that the index-based strategy in IndexGA is effective in improving the solution accuracy. This is possibly because the index-based strategy reduced the FEs in the global exploration, and thus provides more chances for on local learning in the promising regions.

3.2.2 Comparison on Search Speed

According to the results presented in the column 'Eval' in Table II-III and the evolutionary curves illustrated in Fig. 4, the rGA+OLS and IndexGA consume much less FEs to reach the errors than rGA does. Especially on f_1-f_5, both rGA+OLS and IndexGA can achieve solutions within the errors on 2-dimensional functions, whereas the rGA can not reach the errors. These results show the effectiveness of OLS in accelerating convergence.

For the comparison between rGA+OLS and IndexGA, the IndexGA uses less FEs to reach the predefined errors on most of the 2-dimensional functions (f_1-f_5, f_7-f_{10}, f_{12}, f_{13}) and the 4-dimensional functions (f_2-f_5, f_7, f_9, f_{11}, f_{12}). These results indicate faster convergence of IndexGA. The accelerated converging speed is possibly because the index-based strategy reduces the FEs in the global exploration. However, it should also be noted in Table II-III that the effect of IndexGA on reducing FEs weakens as the dimension of problem increases and the number of regions increases.

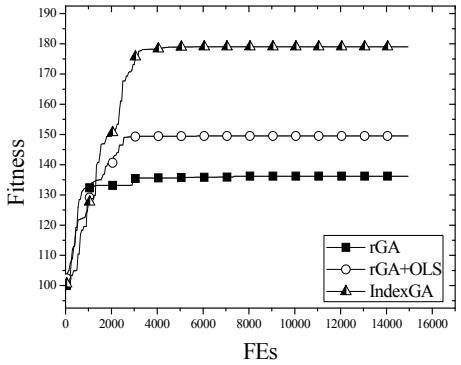

Figure 5. Evolutionary curves of the three tested algorithms on a PEC design problem.

4. INDEXGA FOR PEC APPLICATION

4.1 Design Example

In this section, the indexGA is applied to the PEC design and optimization problem. The design example is a buck regulator with overcurrent protection [17][18]. A power conversion stage (PCS) and a feedback network (FN) comprise the circuit. The components R_1, R_2, R_{C3}, R_4, C_2, C_3 and C_4 in the FN are to be optimized by the traditional real-coded GA and the proposed IndexGA.

According to [17], the optimization problem can be formulated as a maximization problem defined as

$$\Phi_F(FN) = \sum_{R_L=R_{L\_\min},\delta R_L}^{R_{L\_\max}} \sum_{v_{in}=v_{in\_\min},\delta v_{in}}^{V_{in\_\max}} [OF_5(R_L, v_{in}, FN) + OF_6(R_L, v_{in}, FN) + OF_7(RL, v_{in}, FN)] + OF_8(FN) \quad . (7)$$

Here FN=[R_1, R_2, R_{C3}, R_4, C_2, C_3, C_4] is the solution vector, the upper and lower bounds of all the components are presented in Table IV. $R_{L\_\min}$ and $R_{L\_\max}$, $V_{in\_\min}$ and $V_{\text{in\_max}}$ are the upper and lower bounds of R_L and v_{in}, respectively. The functions OF_5, OF_6, OF_7 and OF_8 are objective functions defined in [17].

Three algorithms are tested on the problem, including the rGA, rGA+OLS, and the proposed index GA. Parameter configurations for all the tested algorithms are the same as that used in Section III, and all of the algorithms are executed 10 independent trials. For each trial, the maximum number of FEs is limited to 15000.

4.2 Results

The experimental results of the rGA, rGA+OLS, and the proposed index GA are reported in Table V. In Table V, the column 'Avg.' presents the averaged results of 10 independent runs, the column 'Best' presents the best results obtained in the 10 trials, and the column 'Std dev.' reports

the standard deviation. Fig. 5 illustrates the evolutionary curves of the tested algorithms on functions f_1, f_3, f_9, f_{12}.

According to Table V and Fig. 4, the two algorithms with OLS, i.e. rGA+OLS and IndexGA can obtain better average results than the traditional rGA. Moreover, the IndexGA outperforms rGA+OLS in terms of mean results, best result, and the standard deviation. These results further confirm the effectiveness of IndexGA when applied to the PEC problem.

5. CONCLUSIONS

A GA with index-based strategy to reduce the FEs in the reproduction procedure is proposed. The IndexGA divides the solution space into multiple regions, with each region identified by a unique index. Individuals in the IndexGA are redefined to be indexes instead of solutions. In the reproduction procedure, an evaluated region is never evaluated again. In addition, an OLS procedure is adopted to improve the fitness of promising regions. The IndexGA improves the performance of GAs in terms of both convergence rate and solution accuracy. The effectiveness of the index-based strategy and the OLS are verified by numerical experiments on 13 benchmark functions and an application problem of PEC. Moreover, the experimental results show that the IndexGA is especially promising on low-dimensional real-world application problems with expensive evaluation functions.

The performance of IndexGA on high-dimensional problems deserves further study. In addition, the modified version of IndexGA can be applied to discrete real-world application problems. They are our future work.

Table 7. Results Comparisons on the PEC Design Problem

	Avg.	Best	Std dev.
rGA	136. 159	167. 461	**20. 1519**
rGA+OLS	149. 453	190. 847	23. 919
IndexGA	**178. 987**	**192. 238**	22. 8683

6. ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) No. 61070004, and by NSFC Joint Fund with Guangdong under Key Project U0835002. The authors are with the Key Laboratory of Digital Life, Ministry of Education, China, and also with the Key Laboratory of Software technology, Education Department of Guangdong Province. The corresponding author is Jun Zhang, email: junzhang@ieee.org.

7. REFERENCES

[1] Holland, J. H. "Adaptation in natural and artificial systems". Ann Arbor: Univ. Michigan Press, 1975.

[2] Holland, J. H. Genetic algorithms. Scientific American, 1992, pp. 44-50.

[3] Cavicchio, D. J. Adaptive search using simulated evolution. Ph.D. Dissertation, University of Michigan, 1970.

[4] Leung, Frank H.F., Lam, H.K., Ling, S. H. and Tam, Peter K.S., "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," IEEE Trans. on Neural Networks, Vol. 14, No.1, pp. 79-86, 2003.

[5] Maulik, U. "Medical image segmentation using genetic algorithms," IEEE Trans. on Infomation technology in biomedicine, Vol. 13, No.2, pp.166-173, 2009.

[6] Zhang, J., Chung, Henry S. H., Lo, W. L. "Pseudocoevolutionary genetic algorithms for power electronic circuits optimization," IEEE Trans. on Syst, Man, and Cybrnetics-Part C: Applications and reviews, Vol. 36, No.4, pp.590-598, 2006.

[7] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," IEEE Trans. on Evol. Comput., vol. 3, no. 2, pp. 124–141, Jul. 1999.

[8] Srinivas, M. and Patnaik, L.M. "Adaptive probabilities of crossover and mutation in genetic algorithms," IEEE Trans. on Syst, Man, and Cybernetics, Vol. 24, pp. 656-667, 1994.

[9] Zhang, J., Chung, Henry S. H., Lo, W. L. "Clustering-based adaptive crossover and mutation probabilities for Genetic Algorithms," IEEE Trans. on Evol. Comput., Vol. 11, pp. 326-335, 2007.

[10] John J. Grefenstette, "Optimization of control parameters for genetic algorithms," IEEE Trans. on Syst, Man, and Cybernetics, Vol. SMC-16, pp.122-128, 1986.

[11] Krasnogor, Natalio and Smith, Jim, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," IEEE Trans. on Evol. Comput., Vol. 9, No. 5, 2005.

[12] Ong, Yew Soon and Keane, Andy J., "Meta-lamarckian learning in memetic algorithms," IEEE Trans. on Evol. Comput., Vol. 8, No. 2, 2004.

[13] Smith, J. E., "Coevolving memetic algorithms: a review and progress Report," Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 37, pp. 6-17, 2007.

[14] Holstein, D. and Moscato, P., "Memetic algorithms using guided local search: A case study," in New Ideas in Optimization, D. Corne, F. Glover, and M. Dorigo, Eds. New York: McGraw-Hill, pp. 235–244, 1999.

[15] Krasnogor, N., Blackburne, B., Burke, E., and Hirst, J., "Multimeme algorithms for protein structure prediction," in Lecture Notes in Computer Science, 2002, Proc. Parallel Problem Solving From Nature—VII, pp. 769–778.

[16] Krasnogor, N. and Gustafson, S., "A study on the use of "self-generation" in memetic algorithms," Natural Comput., vol. 3, no. 1, pp. 53–76, 2004.

[17] Zhang, J., Chung, S. H., Lo, W. L., Hui, S. Y. R., and Wu, A., "Implementation of a decoupled optimization technique for design of switching regulators using genetic algorithm," IEEE Trans. Power Electron., vol. 16, no. 5, pp. 752-763, Nov. 2001.

[18] Zhang, J., Shi, Y., and Zhan, Z. H., "Power electronic circuits design: A particle swarm optimization approach," The 7th International Conference on Simulated Evolution And Learning, X. Li et al. (Eds.): SEAL 2008, LNCS 5361, pp. 605–614, 2008.

[19] Montgomery, D. C., Design and Analysis of Experiments, 3rd ed. New York: Wiley (1991).

[20] Hicks, C. R., Fundamental Concepts in the Design of Experiments, 4th ed. TX: Saunders College Publishing (1993).

[21] Eshelman, L. J. and Schaffer, J. D., "Real-coded genetic algorithms and interval-schemata," in Foundations of Genetic Algorithms 2. San Mateo, CA: Morgan Kaufman, 1993, pp. 187–202.

[22] Yao, X., Liu, Y., and Lin, G. M., "Evolutionary programming made faster," IEEE Trans. on Evol. Comput., vol. 3, no. 2, pp. 82–102, Jul. 1999.