한국지능시스템학회 논문지 2011, Vol. 21, No. 2, pp. 206-211
DOI : 10.5391/JKIIS.2011.21.2.206

연속탐색공간에 대한 진화적 해석

Evolutionary Analysis for Continuous Search Space

이준성* · 배병규**
Joon-Seong Lee* and Byeong-Gyu Bae**

*경기대학교 기계시스템공학과
**경기대학교 대학원 기계공학과

요 약

본 논문에서는 연속적인 파라미터 공간에 대한 최적화에 대해 진화적 알고리즘의 특징적인 형상화를 제시한다. 이 방법은유전알고리즘이 연속적인 탐색공간에서의 파라미터 식별에 대해 가장 강점을 지녔다는 점에 착안한 것이다. 유전알고리즘과 제안한 알고리즘과의 주요한 차이점은 개별적 또는 연속적인 묘사의 차이가 있다는 것이다. 잘 알려진 실험함수의 최적화문제를 도입하여 연속 탐색공간 문제에 대해 제안하는 알고리즘에 대해 계산시간 및 사용메모리 등의 성능이 우수하다는 효율성을 보였다.

Abstract

In this paper, the evolutionary algorithm was specifically formulated for optimization with continuous parameter space. The proposal was motivated by the fact that the genetic algorithms have been most intensively reported for parameter identification problems with continuous search space. The difference of primary characteristics between genetic algorithms and the proposed algorithm, discrete or continuous individual representation has made different areas to which the algorithms should be applied. Results obtained by optimization of some well-known test functions indicate that the proposed algorithm is superior to genetic algorithms in all the performance, computation time and memory usage for continuous search space problems.

Key Words : Evolutionary Algorithm, Continuous Search Space

1. 서 론

The advance of computer hardware has increased the popularity of an approach where all the parameters are identified simultaneously and, most commonly, optimization methods are used to find the parameter set by adjusting them until they provide the best agreement between the measured data and the computed model response. As a result, a number of calculus-based optimization techniques were proposed and incorporated to solve this optimization problem [1-3]. These techniques, however, can fail in the actual situation, for example, when the measured data are noisy and the model equations are inaccurate, since they can cause the objective function to be complex such as nonconvex and multimodal. These techniques are thus practically useful only if some regularization technique [4] is incorporated properly.

On the other hand, Evolutionary Algorithms(EAs), which have come to represent Genetic Algorithms (GAs) [5], Evolution Programming [6-9], Evolution Strategies [10] and their recombined algorithms [11], have appeared as robust optimization techniques in the last few decades. EAs are based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem. Each of these algorithms has clearly demonstrated its capability to yield good approximate solutions even in case of complicated multimodal, discontinuous, non-differentiable, and even noisy or moving response surface optimization problems. The popularity of the algorithms is primarily due to their probabilistic but efficient nature.

In this paper, the evolutionary algorithm was specifically formulated for optimization with continuous parameter space. The proposal was motivated by the fact that the genetic algorithms have been most intensively reported for parameter identification problems with continuous search space.

The basic ideas of the formulation of the proposed algorithm are that:

(a) Individuals are each represented by a continuous vector
(b) Reproductive processes are as similar as possible to genetic algorithms such that it can take ad-

접수일자 : 2011년 1월 19일
완료일자 : 2011년 4월 13일
본 연구는 2010학년도 경기대학교 학술연구비(일반연구과제)지원에 의하여 수행되었음.
This work was supported by Kyonggi University Research Grant in 2010. The authors thank Dr. T. Hurukawa for providing relevant advice.

vantage of probabilistic features in genetic algorithms.

2. Formulation

The reproductive operations of the proposed algorithm is intended to be similar to those of genetic algorithms such that it can take the advantage of probabilistic features in genetic algorithms. The major difference of the proposed algorithm from GAs is that the representation of the individual is given by a search point itself: i.e., a real continuous vector. This formulation was made with an assumption that the direct use of the search point may search more efficiently than the representation decoded into a binary string as used in GAs. In this case, the population at generation k is given by

$$P^t = \{x_1^k, ..., u_\lambda^k\} \in X^\lambda. \tag{1}$$

This representation makes us grasp the concept of the individual in a different manner. While the binary string in GAs represent a DNA chromosome, a microscopic representation of human being, the continuous vector representation corresponds to a set of macroscopic information of human being.

The definition of the recombination and mutation becomes the probabilistic distribution of the macroscopic measures accordingly. If the two offspring individuals are formulated to be created from a pair of randomly-selected parental individuals as in GAs, r″: $I^2 \rightarrow$ I, the recombination operation may be defined as

$$\begin{cases} r''(\boldsymbol{x}_\alpha^k, \boldsymbol{x}_\beta^k) = (1-\mu_\alpha^k) \cdot \boldsymbol{x}_\alpha^k + \mu_\beta^k \cdot \boldsymbol{x}_\beta^k \\ r''(\boldsymbol{x}_\beta^k, \boldsymbol{x}_\alpha^k) = \mu_\alpha^k \cdot \boldsymbol{x}_\alpha^k + (1-\mu_\beta^k) \cdot \boldsymbol{x}_\beta^k \end{cases} \tag{2}$$

where x_α^k and x_β^k are parental individuals at generation k and the coefficient μ_i^k, $\forall i \in \{\alpha, \beta\}$, is defined by the normal distribution with mean 0 and standard deviation η_i^k:

$$\mu_i^k = N(0, \eta_i^{k^2}). \tag{3}$$

The standard deviation can be self-adaptive (variable with respect to t) or constant. The self-adaptive strategy has been reported to make the convergence rate required for each generation faster at the expense of the computation time and vice versa. The mutation is not incorporated in the algorithm since the recombination can allow individuals to make large alternations when the coefficient μ_i^k is large.

The evaluation of the fitness can be conducted with a linear scaling, which takes into account the worst individual of the population $P^{k-\omega}$ ω steps before:

$$\Phi(\boldsymbol{x}_i^k) = \max\{\psi(\boldsymbol{x}^k) | \boldsymbol{x}^k \in P^{k-\omega}\} - \psi(\boldsymbol{x}_i^k), \ \forall i \in \{1, ..., \lambda\} \tag{4}$$

as in GAs. Here, ω is the scaling window. Proportional selection, which is the most popular selection operation in GAs, can also be directly used in the proposed algorithms. In this selection, the reproduction probabilities of individuals ps : X →[0, 1] are given by their relative fitness,

$$p_s(\boldsymbol{x}_i^k) = \frac{\Phi(\boldsymbol{x}_i^k)}{\sum_{j=1}^{\lambda} \Phi(\boldsymbol{x}_j^k)}. \tag{5}$$

3. Software

The advantages of the proposed algorithm are its simple formulation as well as the compatibility with GENESIS [12], which is one of the most popular GA software. This software was modified about 300 lines out of 2,400 lines.

The software of the proposed algorithm will start by running the setup program, which creates input and template files. Typical examples of these files are Figs. 1 and 2. While input file contains input parameters and initial random number seeds, the template file describes the representation of parameters.

```
Experiments = 1
Total Trials = 1000
Population Size = 50
Structure Length = 30
Standard Deviation = 0.5
Generation Gap = 1.0
Scaling Window = 5
Report Interval = 200
Structures Saved = 200
Max Gens w/o Eval = 2
Dump Interval = 0
Dumps Saved = 0
Options = aceL
Random Seed = 123456789
Rank Min = 0.75
```

Fig. 1. Example of input file

```
individuals: 3

individual 0
min: -5.12
max 5.12
format::%7.2lf

individual 1
min: -5.12
max 5.12
format::%7.2lf

individual 2
min: -5.12
max 5.12
format::%7.2lf
```

Fig. 2. Example of template file

In the input file, the software allows a number of options which control the kinds of output produced, as well as certain strategies employed during the search. Each option is associated with a single character. The

major available options are listed in Table 1.

To use the software, the user must write an evaluation procedure, which takes on structure as input and returns a double precision value. These procedure must be declared in the user's file starting with the sentences shown in Fig. 3. The display of GENESIS when optimizing an objective function with 30 variables, which can be optimally visible on screen, is shown in Fig. 4.

Table 1. Major available options

D	Display mode. Performance statistics are printed to the screen after each generation
e	use the "elitist" selection strategy.
i	read structures into the initial population.
l	log activity (starts and restarts) in the "log" file.
L	dump the last generation to the "ckpt" file. This allows the user to restart the experiment at a later time, using option "r".
M	maximize the evaluation function.
r	restart a previously interrupted execution.
R	Rank based selection.

```
double eval(vect, individuals)
double vect[ ]:
int individuals:
{
    register int i:
    double sum:

    sum = 0.0:
    for(i = 0: i < genes: i ++ )
        sum += vect[i] * vect[i]:

    return(sum):
}
```

Fig. 3. Example of evaluation function

As the software use most of the C files presented in GENESIS, it is possible to compare its performance with the GA directly, and the next section will discuss the comparison from several performance tests.

```
run until Trials = 10000          executing: measure

Gens Trials Lost Conv  Bias      Online      Offline       Best      Average
  87   8800    0    0 0.000    131.2571     92.2086     45.3817      70.5614

Current Best Structure:  90  Performance: 45.3817
    0.238    -2.354     0.725     0.988     2.073     2.003    -1.887     0.094
   -1.577     0.310     0.026     0.225     0.253    -0.415     0.657    -1.235
    0.090     0.721    -2.072     1.135     0.709    -0.350    -1.648    -1.123
    0.770    -1.804     1.901    -1.165    -0.712    -1.244
```

Fig. 4. Display of GENESIS

4. Comparison

4.1 Test functions

As it is impossible to predict the behavior of the algorithms by theoretical considerations, a set of test functions having continuous search space were prepared to demonstrate the capability of both the canonical GA, which is composed of the *one-point crossover* and proportional selection, and the proposed algorithm. The mathematical characteristics of the test functions are unimodal/multimodal, quadratic/non- quadratic, convex/non-convex and continuous/discontinuous. The test functions are as follows:

Func. I $$f_1(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{x_i^2}\ ,\ \boldsymbol{x} \in \boldsymbol{R^n}\ ; n = 30,$$
$$-5.12 \le x_i \le 5.12,\ x^* = [0, ..., 0]^T,$$
$$f1(x^*) = 0, \quad (6)$$

Func. II $$f_2(x) = 6n + \sum_{i=1}^{n} \lfloor \boldsymbol{x_i} \rfloor\ ,\ x \in \boldsymbol{R^n}\ ; n = 5,$$
$$-5.12 \le x_i \le 5.12,\ x^* \in [-5.12, ..., -5]^T,$$
$$f2(x^*) = 0, \quad (7)$$

Func. III $$f_3(\boldsymbol{x}) = nA + \sum_{i=1}^{n} \boldsymbol{x_i^2} - A\cos(\omega x_i)\ ,\ \boldsymbol{x} \in \boldsymbol{R^n}\ ;$$
$$n = 20, A = 10, \omega = 2\pi,$$
$$-5.12 \le x_i \le 5.12,\ x^* = [0, ..., 0]^T,$$
$$f3(x^*) = 0, \quad (8)$$

Func. IV Func. III except that $\omega = \pi/2$.

Func. I is the simplest quadratic function, which is also characterized by unimodality, continuity and convexity. It is often used as the first test case of nonlinear functions since many objective functions formulated in reality take this form. Func. II is a simple linear but discontinuous function, which comprises a number of plateaus. All gradient-based methods are not useful for this function due to its discontinuity. The discontinuity gives difficulty even to other optimization methods since there is no guidance towards the edge of each plateau. Funcs. III and IV are continuous, multimodal test funcitons, which are generated by modifying the value of parameter ω. The last term of the function yields a number of local optima, which renders optimization methods difficult to search, depending on the initial search point. Three- dimensional graphical representations of these functions are shown in Figs. 5 - 8.

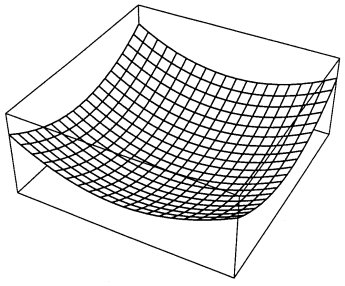

Fig. 5. 3D representation of Func. I

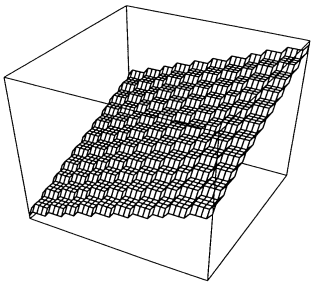

Fig. 6. 3D representation of Func. II

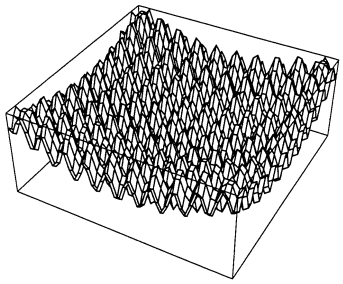

Fig. 7. 3D representation of Func. III

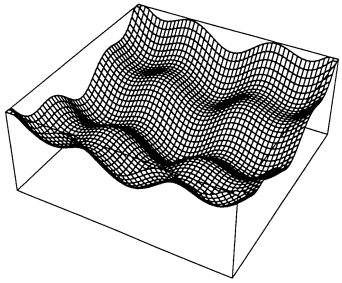

Fig. 8. 3D representation of Func. IV

Table 2 Internal parameters for both the algorithms

	Canonical GA	Proposed algorithm
Population size	50	50
Bit length per variable	30	
Crossover rate	0.95	
Mutation rate	0.001	
Standard deviation		0.5(constant)
Generation gap	5	5
Scaling window	1.0	1.0

4.2 Performances

Internal parameters selected for both the algorithms are listed in Table 2. The crossover and mutation rates in the canonical genetic algorithm, 0.6 and 0.0001, are typically used in many publications. The standard deviation of the proposed algorithm was set to be constant for simplicity. All the other internal parameters were set identical for both the algorithms. Note here that the elitist strategy was incorporated in the selection process of both algorithms. For generality, ten runs were performed for each test, and the average performance of each algorithm was taken.

The results of the search performance of both the algorithms until 2,300 generations are shown in Figs. 9-12. In the figures, real lines indicate the outcome from the proposed method whereas the results by the canonical GA are indicated by the broken lines. The result of the first test clearly reflects the superiority of the proposed algorithm on the unimodal function optimization. The second result, then, indicates that the proposed algorithm still out-performs the canonical GA even when the function is partially discontinuous. The canonical GA, in fact, could not find the global optimum $x^*=0$ in all the ten runs within the prescribed number of generations whereas the proposed algorithm resulted in the global optimum in all the cases. The proposed algorithm also have a better performance on Func. III and IV than genetic algorithm. The genetic algorithm was nearly in the state of premature convergence, having a slow convergence rate before reaching the global optimum, for both the tests.

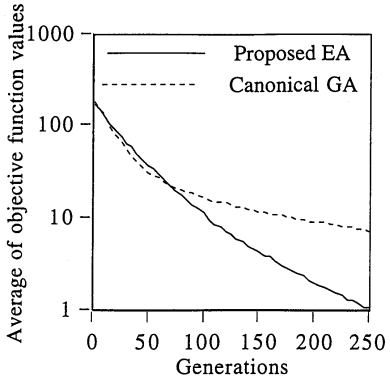

Fig. 9. Average of objective function values vs generations for Func. I

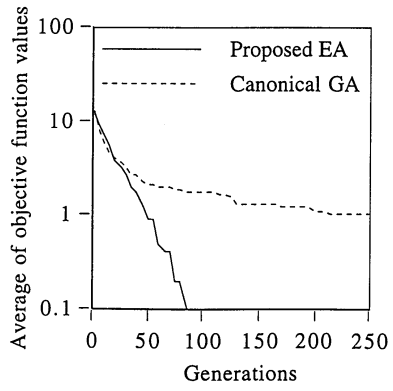

Fig. 10. Average of objective function values vs generations for Func. II

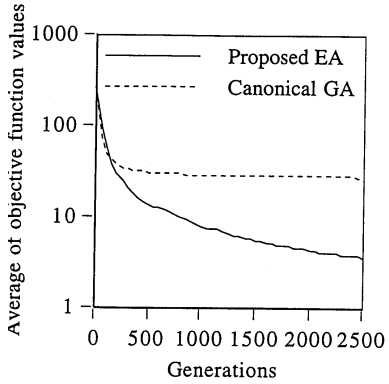

Fig. 11. Average of objective function values vs generations for Func. III

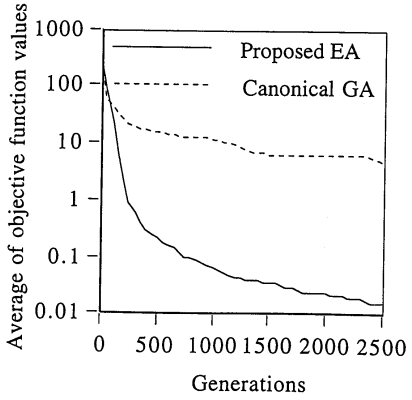

Fig. 12. Average of objective function values vs generations for Func. IV

4.3 Computation time and memory usage

The computation time and memory required for the optimization of Func. I up to 2,000 generations with SparkStation III were compared for both algorithms. Parameters listed in Table 2 were used, except the bit length per variable for genetic algorithm, which were varied every five bits within a range of 10 and 30.

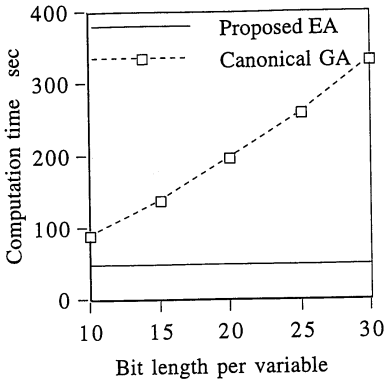

Fig. 13. Computation time vs bit length

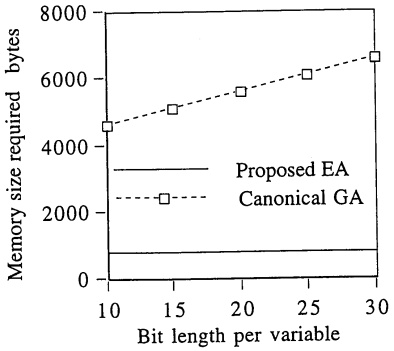

Fig. 14. Memory usage vs bit length

The results of the computation time and memory usage are shown in Figs. 13 and 14. These results indicate that, for the canonical genetic algorithm, the more computation time and memory usage the longer the bit length per variable. The fact that the computation time increase with the precision of each variable increased is obviously caused by the additional computational effort for decoding. The reason for the increase of the memory usage is that both the binary and decoded representations must be saved in memory. Although the 30-bit representation was far beyond the real representation in precision, the genetic algorithm required seven times of computation time and eight times of memory usage of the proposed algorithm.

Table 3. Comparison between genetic algorithms and proposed algorithm

	GAs	Proposed algorithm
Representation	Genotype	Phenotype
Resolution	Restricted	
Recombination	Main (Discrete)	Only one operator (Continuous)
Mutation	Background (Discrete)	
Decoding	Necessary	
Search performance	Explorative (Slow convergence) (Global search)	Exploitative (Fast convergence) (Local search)
Optimization	Discrete optimization	Continuous optimization

Although the overall performance of the algorithms significantly relies upon the class of objective function, characteristics listed in Table 3 may be clarified for comparison. As mentioned, the most obvious difference have been identified with respect to the representation of solutions in both algorithms. Whilst individuals of genetic algorithms are each represented by a microscopic human representation of chromosome termed genotype, the proposed algorithm uses an individual having macroscopic phenomenological information of humans termed phenotype. This gives genetic algorithms the critical disadvantage that the resolution of parameter is restricted. The double precision in the present computers is 64 bits. Creating this precision is

genetic algorithms requires huge memory and thus, if the search space is high-dimensional, it is not realistic. Long individual representation in genetic algorithms also yield numerous computation time for decoding of each individual to a continuous vector form.

5. Conclusions

An evolutionary algorithm based on genetic algorithms, which works efficiently for continuous search space optimization problems was proposed by the authors. The difference of primary characteristics between genetic algorithm and the proposed algorithm, discrete or continuous individual representation has made different areas to which the algorithms should be applied. Results obtained by optimization of some well-known test functions indicate that the proposed algorithm is superior to genetic algorithms in all the performance, computation time and memory usage for continuous search space problems.

As a main result of the experimental comparison we conclude the hypothesis, that the selection scheme provides the main control mechanism of the relationship between the contradicting purpose of exploration and exploitation, i.e., the purposes of high confidence to find a global optimum on the one hand and high velocity of the optimization process itselt on the other hand. Genetic algorithms seem to explore the search space much more than the proposed algorithm by the fact that its individual representation leads to completely new offspring in the continuous domain, at the expense of convergence rate. However experimental results show that the multimodality of Func IV is still well dealt with the proposed algorithm.

References

[1] Cailletaud, G. and Pilvin, P., "Identification and Inverse Problem Related to Material Behavior," *Inverse Problems in Engineering Mechanics*, pp. 79-86, 1994.

[2] Yang, J. H., "Study on the Optimization of Neural Network Using Parallel Evolutionary Algorithm," *Master Thesis*, Seoul National University, 2001.

[3] Soon, N. S, Yeo, M. H., Yoo, J. S., "An Efficient Evolutionary Algorithm for Optimal Arrangement of RFID Reader Antenna," *Digital Contents Society*, vol. 9, no. 10, pp. 40-50, 2009.

[4] Baumeister, J., *Stable Solution of Inverse Problems*, Vieweg, Braunschweig, 2007.

[5] Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

[6] Pedro Antonio Gutierrez, Cesar Hervas, Manuel Lozano, "Designing Multilayer Perceptrons Using a Guided Saw-tooth Evolutionary Programming Algorithms," *Soft Computing*, vol. 14, no. 6, pp. 599-613, 2009.

[7] Fogel, L.J. Oweens, A.J. and Walsh, M.J., *Artificial Intelligence through Simulated Evolution*, NewYork, Willey, 1996.

[8] France Cheong, Richard Lai, "Simplifying the automatic design of a fuzzy logic controller using evolutionary programming," *Soft Computing*, vol. 11, no. 9, pp. 39-846, 2006.

[9] Cheon, S. H., "Improvement of Support Vector Clustering Using Evolutionary Programming and Bootstrap," *Int. J. of Fuzzy Logic and Intelligent Systems*, vol. 8, no. 3, pp. 196-201, 2008.

[10] Rechenberg, I., *Evolutionstrtegie: Optimierung Tchnischer Systeme Nach Prinzipien der Bologischen Evolution*, Stuttgart, Frommann-Holzboog, 1983.

[11] Hoffmeister, F. and Back, T., "Genetic Algorithms and Evolution Strategies: Similarities and Differences," *Technical Report*, University of Dortmund, Germany, 1992.

[12] Grefenstette, J. J., "GENESIS: A System for Using Genetic Search Procedures," *Proceedings of the 2004 Conference on Intelligent Systems and Machines*, pp. 161-165, 2004

저 자 소 개

이준성(Lee, Joon-Seong)

1888년 : 성균관대 대학원 기계공학과 석사
1995년 : 동경대학교 공학박사
1988～1991 : 육군사관학교 기계공학과 교수
1996～현재 : 경기대학교 기계시스템공학과 교수

관심분야 : 퍼지이론, 신경회로망, 진화적 알고리즘
Phone : 031-249-9813
Fax : 031-244-6300
E-mail : jslee1@kyonggi.ac.kr

배병규(Bae, Byeong-Gyu)

2010년 : 경기대학교 기계공학과 학사
2010년～현재 : 동 대학원 기계공학과 석사과정

관심분야 : 피로해석, 퍼지 이론, 신경회로망
Phone : 031-249-9810
E-mail : sungjae83@nate.com