A Precise Evolutionary Approach to Solve Multivariable Functional Optimization

Md. Robiul Islam, M.A.H. Akhand
Dept. of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna, Bangladesh
E-mail: robikuet@yahoo.com, akhand@cse.kuet.ac.bd

K. Murase
Dept. of Human and Artificial Intelligence Systems
University of Fukui
Fukui, Japan
E-mail: murase@synapse.his.u-fukui.ac.jp

Abstract— Genetic Algorithm (GA) is a stochastic search and optimization method imitating the metaphor of natural biological evolution. GA manages population of solutions instead of a single solution to find an optimal solution to a given problem. Although GA draws attention for functional optimization, it may search same point again due to its probabilistic operations that hinder its performance. In this study, we make a novel approach of standard Genetic Algorithm (sGA) to achieve better performance. The modification of sGA is investigated in selection and recombination stages and proposed Precise Genetic Algorithm (PGA). PGA searches the target space efficiently and it shows several potential advantages over the conventional GA when tested for solving functions having multiple independent variables.

Index Terms: Function Optimization, Genetic Operators and Fitness Evaluation Function.

I. INTRODUCTION

Optimization problems, in simple terms, are to find the best or close to the best solutions to the problems [1, 2, 5]. The term optimization mathematically refers to the study of problems that have the form:
Given: a function $f : A \to R$ from some set A to the real numbers. Sought: an element x_0 in A such that $f(x_0) \geq f(x)$ for all x in A ("maximization") or such that $f(x_0) \leq f(x)$ for all x in A ("minimization"). Function optimization problem exists both in single and high dimensional search space. The function having more than one independent variable is called multivariable function which is reasonably a complex study than that of single variable function. In general, optimization is a composite perceptual task and metaheuristics mimicking biologically motivated computing have become a very popular research topic in the last few years.

The biologically motivated computing activities have waxed and waned over the period of time. The evolutionary algorithm has become the most promising focus for the scientists and engineers especially in the area of simulation models, multi-objective and combinatorial optimization, mathematical problems, image processing, engineering design and control problems, fitting nonlinear curves to data, setting weights on neural networks and so on [1, 2, 15]. Generally evolutionary approach involves mechanisms (e.g. Evolutionary programming, Evolution strategy, Neuroevolution, Genetic algorithm, Genetic programming and so on) that are inspired by biological evolution such as reproduction, mutation, recombination, natural selection and survival of the fittest. GA is a well known method of these mechanisms and draws attention for solving functional optimization problems [1, 2, 15, 19].

Genetic algorithms are stochastic search methods that imitate the metaphor of natural biological evolution [1, 3, 4]. It operates on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution [2, 5, 14, 15]. GAs are the vigorous optimization techniques based on natural genetics that denote the ability of the GAs in finding the global optimum, or a near-optimal point, for any optimization problem [1, 2, 6,13].

Although GA is performed well in optimization problems, due to working with population of solutions it faces computation time and slow convergence through its basic steps such as selection, reproduction and replacement [8, 9, 20]. The random selection in GA causes duplication and repeated identical calculation which hampered its overall performance [3, 10, 11]. To enhance performance, a modification of GA is investigated in selection and recombination stages to maintain population diversity. The proposed method is called Precise Genetic Algorithm (PGA). The primary motivation for the proposed PGA is to ensure the successive convergence in optimization problems to reach optimal solution with a minimal time. Population diversity hinders premature convergence and helps to get global optimal points in the search space. PGA also eliminates the possibility to search in the same point that could be expensive in GA. Experimental results on a set of sample optimization functions reveal that PGA able to reply optimal solution within a less number of generation(s) than that of standard GA.

The rest of the paper is organized as follows. Section II explains GA and aspects of proposed PGA to solve several optimization problems. Section III presents experimental result and section IV concludes the paper with brief summary.

DOI: 10.5176_2010-2283_1.2.33

II. GA AND PGA TO SOLVE OPTIMIZATION PROBLEM

Since GA is the base of our proposed Precise Genetic Algorithm (PGA), this section first explains GA briefly and then describes PGA.

A. GA to Solve an Optimization Problem

Standard GA (sGA) is a class of evolutionary algorithms that model natural processes, such as selection, recombination, mutation and migration [1, 2, 7, 15, 18]. Fig.1 shows the structure of a simple GA. It works on the population of individuals instead of single solution and it may works in a parallel manner [15, 16].

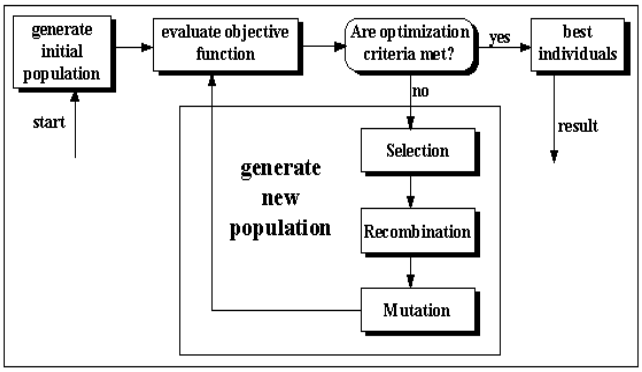

Fig. 1: Structure of a genetic algorithm

At the beginning, individuals (the population) are randomly initialized that stands as initial population. The objective function is then evaluated for these individuals. If the optimization criteria are not met, the creation of a new generation starts [12, 15, 17]. Individuals are selected according to their fitness for the production of offspring. Parents are recombined to produce offspring. All offspring will be mutated with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until the optimization criteria are reached [15, 18, 19].

Problems regarding GA are significantly related to the computation time and slow convergence [8, 9]. Due to its probabilistic nature, the duplicate selection and repetition of same searching points deteriorate the overall performance of GA. To overcome the weakness, we have investigated Precise Genetic Algorithm (PGA); the next section describes the proposed method in detail.

B. Precise Genetic Algorithm (PGA)

Precise Genetic Algorithm is an extension of the traditional genetic algorithm and modified by a search method to further improve individual's fitness that may keep high population diversity and reduce the likelihood premature convergence. This technique offers a very flexible and reliable tool able to search for a solution within a global context.

PGA effectively incorporates the global exploring ability of the genetic algorithm with the help of population diversity and the local convergent ability of the precise algorithm by adding new search points. Other techniques are also employed by PGA is to ensure outperformance over standard GA (sGA). Standard GA always accepts the newly produced individuals as offspring in the crossover and mutation. On the other hand, PGA does not directly allow two offspring like sGA. PGA always chooses the best chromosomes during the crossover and mutation process. In the crossover process, two parents are chosen to produce two offspring based on the classical multipoint crossover. The two parents and offspring compete with each other and PGA chooses two best chromosomes as offspring. Likewise, in the mutation process, the chromosome chosen to mutate and the altered chromosome compete with each other and PGA accepts the better one as offspring. With each new generation of individuals the overall fitness value of the population should increase. The process of creating offspring generations based on the former generation could be repeated until the optimum is reached. The coming sections explain steps of PGA in detail considering sGA as a base method dealing with function optimization.

B.1. Chromosomal representation

PGA uses the similar encoding scheme like sGA for function optimization. A binary vector is used as a chromosome to represent real values of x_i . For instance consider the following sample function should be optimized with PGA:

$$f_1(x_1, x_2) = x_1.\sin(10\pi x_1) + x_2.\cos(10\pi x_2) + 2$$

where $-1 \le x_1 \le 3$ and $-1 \le x_2 \le 2$. We wish to optimize the function f_1 with some required precision. The length of the vector depends on the required precision. In this case, it is considered that the desired output result should be 4 places after decimal point, i.e. the required precision is four decimal places for each variable. The domain of variable x_1 has length 4; the precision requirement implies that the range [-1, 3] should be divided at least 4×10000 equal size ranges. This means that 16 bits are required as the first part of the chromosome:

$32768 = 2^{15} < 40000 \le 2^{16} = 65536$. Similarly, the domain of variable x_2 has length 3; the precision requirement implies that the range [-1, 2] should be divided at least 3×10000 equal size ranges. This means that 15 bits are required as the second part of the chromosome:

$16384 = 2^{14} < 30000 \le 2^{15} = 32768$. The total length of a chromosome (solution vector) is then $m = (16+15) = 31$ bits; the first 16 bits code x_1 and rest 15 bits |17-31| code x_2. The binary string $<b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}\ldots b_0>$ and $<b_{30}b_{29}b_{28}b_{27}\ldots b_{16}>$ map into a real number x_i from

the range [-1…3] and [-1…2] respectively is completed in two steps:

- Convert the binary string from the base 2 to base 10
  $(<b_{15}b_{14}b_{13}b_{12}...b_0>)_2 = (\sum b_i.2^i)_{10} = x_1'$
  $(<b_{30}b_{29}b_{28}b_{27}...b_{16}>)_2 = (\sum b_i.2^i)_{10} = x_2'$
- Find a corresponding real number x_i .

The chromosomes (0000000000000000) and (1111111111111111) represent boundaries of the domain [-1, 3]. Each chromosome is a binary vector of several bits and converts it into corresponding real number to evaluate function.

The minimum/maximum of a function $(y = f(x_i))$ is found based on a variation of x_i beginning with one or more starting points. The basic element of a GA is the artificial individual consists of a chromosome and a fitness value. The every changing of the chromosome leads to a changing of the individual fitness. In this case (searching a maximum of a function), an artificial individual only consists of a value of x_i and $y = f(x_i)$. x_i plays the role of a chromosome and y plays the role of the fitness.

The remarkable problems regarding GA encoding are duplication selection and searching same points again which significantly affects the performance and makes slow convergence. The next sub sections explain PGA as a remedy which evolved with a set of search point generation.

B.2. Search Points Generation

It is given the attention with the review of sGA and the following type optimization problems: Maximize $f(x_1, x_2, ..x_m)$ where each x_i is a real parameter subject to $a_i \le x_i \le b_i$ for some constants a_i and b_i . The formula $x_i = left\ value + x_i' \times (right\ value - left\ value) \div (m_i - 1)$ is used to generate new search points within specific ranges by means of chromosomes avoiding duplication for better convergence. A representation having each variable x_i coded as a binary string of length m_i clearly satisfies the precision requirement. Additionally, the following formula interprets each such string:

$x_i = a_i + decimal(00100\cdots1001_2) \times (b_i - a_i) \div (m_i - 1)$

where m = no. of used bit in chromosome. As for example, if the bit string size is 16 maps into a real number in the range [-1…3] then the search point is generated by $x_1 = -1 + x_1' \times 4 \div (16 - 1)$ where x_1' is the decimal value of the corresponding bit string. Similarly $x_2 = -1 + x_2' \times 3 \div (15 - 1)$ where x_2' is the decimal value of the corresponding bit string.

B.3. Precise Crossover

Crossover is the process of creating a new offspring by combination of parental individuals [2, 12]. The bits between the numbers *pos1* and *pos2* indicate the position of the crossing points. From two chromosomes
$v_1 = (b_1; ..... ; b_{pos1}; b_{pos1+1}; ..... ; b_{pos2}; ..... ; b_m)$ *and*
$v_2 = (c_1; ..... ; c_{pos1}; c_{pos1+1}; ..... ; c_{pos2}; ..... ; c_m)$
two new chromosomes are generated through exchanging the corresponding bits between positions *pos1* and *pos2*:
$v_1' = (b_1; ..... ; c_{pos1}; c_{pos1+1}; ..... ; c_{pos2}; ..... ; b_m)$ *and*
$v_2' = (c_1; ..... ; b_{pos1}; b_{pos1+1}; ..... ; b_{pos2}; ..... ; c_m)$
PGA does not directly accept two offspring v_1' and v_2' as sGA does. We compute all the fitness of $\{ v_1; v_2;\ v_1'; v_2' \}$. Then we choose two best chromosomes from these four as the offspring according to their fitness values.

B.4. Precise Mutation

The probability of mutation (p_m) normally sets in smaller range e.g., 0.1. For each chromosome in the current (i.e. after crossover) population and for each bit within the chromosome: For each integer i in $[1, m]$, generate a random number r_i in the range $[0; 1]$. If $r_i < p_m$, then mutate the ith bit of $v = (b_1; ..... ; b_i; ..... ; b_m)$ to generate a new chromosome $v' = (b_1; ..... ; 1 - b_i; ..... ; b_m)$. Then we compute the fitness of v and v' and PGA choose the better chromosome as the offspring.

B.5. Precise Selection

Selection is the process of picking out a suitable individual from the population in order to create a new individual. During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where suitable solutions (as measured by a fitness function) are typically more likely to be selected. Suitable individuals are individuals with a good fitness [7, 13, 16]. Here we use precise elitist selection scheme to select an elitist chromosome with the highest fitness value, which is copied directly into the new population of next generation. It is important to prevent promising individuals from being eliminated from the population during the application of genetic operators. The other chromosomes are selected by roulette-wheel selection process, where the selection probability of each individual is proportional to its fitness value. Selection operator is the implementation of the principle "survival of the fittest". Suitable parental individuals are such individuals with a high y value because the maximum of the function has to be found.

B.6. Evaluation function and fitness

Roulette Wheel Selection: Let $f_1, f_2, ...f_\mu$ be fitness values of individuals 1, 2, . . . , μ. Then the selection probability for individual *i* is:

$$p_i = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$

For the selection process (selection of a new population with respects to the probability distribution based on fitness values), a roulette wheel with slots sized according to fitness is used:

Calculate the fitness value $eval(v_i)$ for chromosome v_i

$i = 1, \ldots\ldots, pop\_size:$

Find the total fitness of the population

$$F = \sum_{i=1}^{pop\_size} eval(v_i).$$

Calculate the probability of a selection p_i for each chromosome v_i *(i = 1,.....,pop_size):*

$$p_i = eval(v_i) / F$$

Calculate a cumulative probability q_i for each chromosome v_i *(i = 1,.....,pop_size) :*

$$q_i = \sum_{j=1}^{i} p_j$$

The selection process is based on spinning the roulette wheel *pop_size* times; each time it selects a single chromosome for a new population in the following way:

1. Generate a random number *r* from the range [0….1].

2. If $r < q_1$ then select the first chromosome (v_1); otherwise select the *ith* chromosome v_i *(2≤ i ≤ pop_size)* such that $q_{i-1} < r \le q_i$.

III. Experimental Studies

This section evaluates PGA on several optimization problems. We have implemented and tested PGA on a set of test functions and compare its performance with sGA. Table 1 shows the test functions of this study. Both sGA and PGA are tested for the following functions with same encoding scheme. Other parameters are as follows: Population size = 50, No. of bits in each individual = 31, Probability of mutation $p_m = 0.1$, Probability of crossover $p_c = 0.6$, Total Generation = 100 and the results are the average of 50 independent runs. The aim is to find the maximum value of the test function. For instance, the maximal value of the function f_1 is at $x_1 = 2.850340$, $x_2 = 2.000000$ and the value is 6.850171.

In our proposed Precise Genetic Algorithm, the best chromosome $v_{\max} = (11110110011010111111111111111111)$ was found after 70 generation for a sample runs which corresponds to the value $x_{i\max} = [2.850340, 2.000000]$ for function f_1. Table 2 shows detail particulars of that point. On the other hand, sGA return the maximum value 6.760506 after 80 generation. Table 3 shows the comparison between maximum value of PGA and sGA for the test functions.

Table 1: Test Functions with range.

Test Function	Range
$f_1(x_1, x_2) =$ $x_1.\sin(4\pi x_1) + x_2.\cos(4\pi x_2) + 2$	$-1 \le x_1 \le 3$ $-1 \le x_2 \le 2$
$f_2(x_1, x_2) =$ $x_1^2 + x_2^2 + 25(\sin^2 x_1 + \sin^2 x_2)$	$-1 \le x_i \le 3$
$f_3(x_1, x_2) =$ $5.5 + x_1.\sin(4\pi x_1) + x_2.\sin(20\pi x_2)$	$-3 \le x_1 \le 12.1$ $4.1 \le x_2 \le 5.8$
$f_4(x_1, x_2) =$ $20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$	$-1 \le x_1 \le 3$ $-1 \le x_2 \le 2$
$f_5(x_1, x_2) = 100(x_1^2 + x_2)^2 + (1 - x_1)^2$	$-2.048 \le$ $x_i \le 2.048$
$f_6(x) = x.\sin(4\pi x) + 2$	$-1 \le x \le 3$

Table 2: Sample result of PGA for function f_1.

Generation (P)	No. of individuals (n)	Best individual v_{max}	Value of x_{imax}	Eval $(v_{max}) = f(x_{imax})$	T. fitness of the pop. $\sum_{i=1}^{pop\_size} eval(v_i).$
2...100	50	11110110011010111111111111111111	[2.850340, 2.000000]	6.850171	303.608427

Table 3: Comparison between PGA and sGA.

Test Function	Max. value for PGA	Max. value for sGA
f_1	6.850171	6.760506
f_2	55.140726	55.140643
f_3	32.850254	32.380927
f_4	48.54326	48.535985
f_5	3897.734227	3897.734227
f_6	4.850151	4.850151

In every run of the PGA makes the better or equal result to obtain successive convergence than that of sGA without a notable increase in the computational complexity. For both single and multivariable functional optimization, the experimental results show that the PGA converges to the global maxima accurately and much faster than that of sGA. Fig. 2 compares total fitness and Max. value of $f_1(x_1, x_2)$ in

between sGA and PGA. The Fig. 3 for test function $f_5(x_1, x_2)$ also clearly indicates the successive convergence of PGA. Most importantly, PGA converges rapidly in comparison with standard GAs. It is obvious that the PGA helps to solve optimization problems without depending on some profound mathematical and statistical optimization theories. From the result it is found that PGA is shown better than sGA.

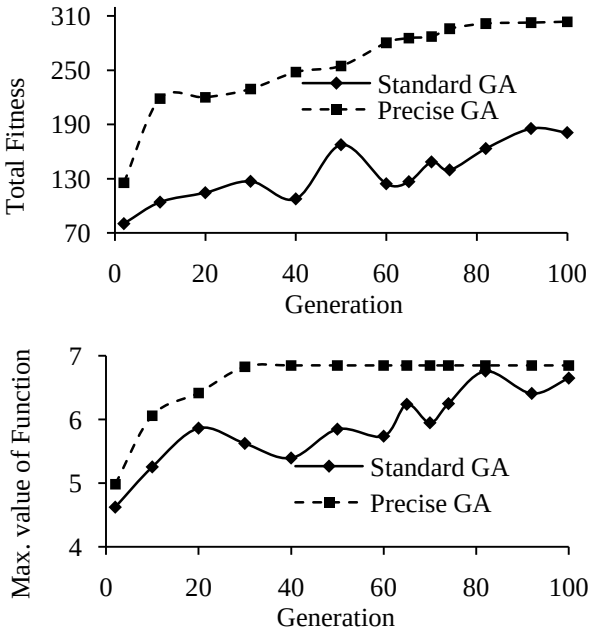

Fig. 2: Fitness curve and convergence comparison of f_1.

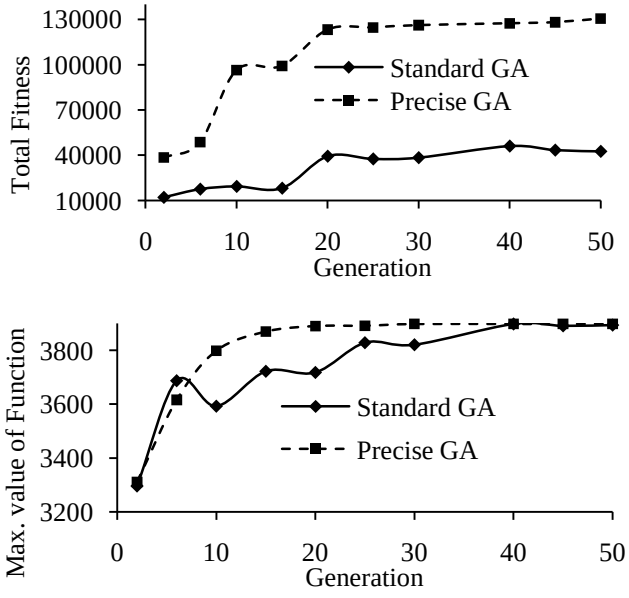

Fig. 3: Fitness curve and convergence comparison of f_5.

IV. CONCLUSIONS

In this study, we modified standard Genetic Algorithm (sGA) for better performance and the new technique called Precise Genetic Algorithm (PGA) is presented. The new method performs better and gradually increases the convergence without much cost of speed than that of standard GA (sGA) when tested for multivariable function optimization. The experimental results indicate that the proposed method succeeds in avoiding premature convergence by maintaining a diverse population. This algorithm uses a precise mutation, crossover and selection techniques to produce a legal offspring and avoid the permutation and duplication problem of sGA. Precise elitism technique is also implemented in PGA to decrease simulations needed to optimize a test function. PGA is shown to give better results in the context of the quality and the time needed to reach the optimal solutions. Modifications of the standard GA to save previously computed fitness and functional values provide significant performance improvement. Hence, the findings and experimental results instruct us to tell that the PGA is excellent and awfully efficient for successively finding an approximate global maximum in high dimension search space.

REFERENCES

[1] Abdullah Konak, David W. Coit, Alice E. Smith (2006), Multi-objective optimization using genetic algorithms: A tutorial, Information Sciences and Technology, Penn State Berks, USA.

[2] Kulvinder Singh and Rakesh Kumar (2010), Optimization of Functional Testing using Genetic Algorithms. International Journal of Innovation, Management and Technology, Vol. 1, No. 1, April 2010 ISSN: 2010-0248.

[3] David Beasley, David R. Bully and Ralph R. Martinz (1993), An Overview of Genetic Algorithms, University of Cardi, university computing 15(2),58-69.

[4] Bhupinder Kaur and Urvashi Mittal (2010), Optimization of TSP using Genetic Algorithm. Advances in Computational Sciences and Technology. ISSN 0973-6107 Volume 3 Number 2 (2010) pp. 119–125, © Research India Publications.

[5] B.V. Babu and Rakesh Angira, Optimization of non-linear functions using evolutionary computation. Birla Institute of Technology & Science, Pilani 333 031.

[6] Matti Palonen, Ala Hasan, Kai Siren (2009), A genetic algorithm for optimization of building envelope and hvac system parameters, Eleventh international ibpsa conference, Glasgow, Scotland, July 27-30, 2009.

[7] Enrique Alba and Carlos Cotta (2004), Evolutionary Algorithms, Dept. Lenguajes y Ciencias de la Computacion, ETSI Informatica, Universidad de Malaga, Campus de Teatinos, Malaga – Spain, 2004.

[8] Z.-Q. Meng (2007), Autonomous genetic algorithm for functional optimization, Progress In Electromagnetics Research, PIER 72, 253–268, 2007.

[9] Yang Chen, Jinglu Hu, Kotaro Hirasawa and Songnian Yu (2007), GARS: An Improved Genetic Algorithm with Reserve Selection for Global Optimization. GECCO'07, July 7–11, 2007, London, England.

[10] G. A. Jayalakshmi, K. Srinivasan, R. Rajaram (2005), Performance Analysis of a Multi-phase Genetic Algorithm in Function Optimization, Thiagarajar College of Engineering, Madurai 625 015.

[11] Kinjo, Oshiro, Kurata, et al. (2006), Improvement of searching performance of real-coded genetic algorithms by the use of biased probability distribution function and mutation (in Japanese). Trans SICE 42:581–590

[12] Rasheed (1999), Guided crossover: a new operator for a genetic algorithm based optimization. Proceedings Congress on Evolutionary Computation, CEC '99, IEEE, USA, vol 2, pp 1535–1541

[13] Bäck, T. Hoffmeister, and Schwefel (1991), A survey of evolution strategies. In Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kauffman, San Mateo, CA.

[14] Chi-Ming Lin and Mitsuo Gen (2007), An Effective Decision-Based Genetic Algorithm Approach to Multiobjective Portfolio Optimization Problem Applied Mathematical Sciences, Vol. 1, 2007, no. 5, 201 – 210.

[15] Michalewicz Z. (1996), Genetic Algorithms + Data structures = Evolution Programs. Springer-Verlag, Heidelberg. ISBN 3-540-60676-9.

[16] Mitchell, M. (1998), An Introduction to Genetic Algorithm. Prentice-Hall of India Private Limited, New Delhi-110001, (Eastern Economy Edition).

[17] JH. Holland (1995), Adaptation in natural and artificial systems. MIT Press, Cambridge.

[18] Joe-Ming Yang and Cheng-Neng Hwang (2002), Optimization of corrugated bulkhead forms by genetic algorithm, Journal of Marine Science and Technology, Vol. 10, No. 2, pp. 146-153 (2002)

[19] D. E. Goldberg, Genetic Algorithms, Addison-wesley, 1998.

[20] Yasuhito Sano and Hajime Kita (2002), Optimization of Noisy Fitness functions by means of Genetic Algorithms using History of Search with Test of Estimation, Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI , USA, pp. 360-365.

Md. Robiul Islam is a postgraduate student in the Dept. of Computer Science and Engineering (CSE) at Khulna University of Engineering and Technology (KUET), Bangladesh. He also acquired the B.Sc. Engineering degree in CSE in 2004 and PGD in Computer Networking in 2008. His research interests include evolutionary computation and other Bio-inspired Computing Techniques, Machine Translation, Data Warehousing and Mining, Soft Computing, and Computer Networking. He is a member of International Association of Computer Science and Information Technology (IACSIT) and Bangladesh Open Source Network (BDOSN).

M. A. H. Akhand received the B.E. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET), Bangladesh in 1999, the M.E. degree in Human and Artificial Intelligent Systems in 2006, and the Doctoral degree in System Design Engineering in 2009 from University of Fukui, Japan. He joined as a lecturer at the Dept. of Computer Science and Engineering at KUET in 2001, and is now an Assistant Professor. His research interest includes artificial neural networks and the applications to classification and on-line business systems.

K. Murase is a Professor at the dept. of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, Fukui, Japan, since 1999. He received M.E. in Electrical Engineering from Nagoya University in 1978, Ph.D. in Biomedical Engineering from Iowa State University in 1983. He Joined as a Research Associate at Dept. of Information Science of Toyohashi University of Technology in 1984, as an Associate Professor at the Dept. of Information Science of Fukui University in 1988, and became the Professor in 1992. He is a member of The Institute of Electronics, Information and Communication Engineers (IEICE), The Japanese Society for Medical and Biological Engineering (JSMBE), The Japan Neuroscience Society (JSN), The International Neural Network Society (INNS), and The Society for Neuroscience (SFN). He serves as a Board of Directors in Japan Neural Network Society (JNNS), a Councilor of Physiological Society of Japan (PSJ) and a Councilor of Japanese Association for the Study of Pain (JASP).