퍼지 및 지능시스템학회 논문지 2007, Vol. 17, No. 1, pp. 86-93

Competitive Generation for Genetic Algorithms

Sung Hoon Jung[1]

Department of Information and Communication Engineering, Hansung University, Seoul 136-792, Korea

Abstract

A new operation termed competitive generation in the processes of genetic algorithms is proposed for accelerating the optimization speed of genetic algorithms. The competitive generation devised by considering the competition of sperms for fertilization provides a good opportunity for the genetic algorithms to approach global optimum without falling into local optimum. Experimental results with typical problems showed that the genetic algorithms with competitive generation are superior to those without the competitive generation.

Key words : genetic algorithms, competitive generation, premature convergence, optimization

1. Introduction

Genetic algorithms (GAs) based on the genetic processes of biological organisms have been successfully applied to a wide range of problems [1–11]. According as the applications increase various methods to enhancing the performance of genetic algorithms have also been introduced to date [9–16]. Most of such methods mainly modified existing operations such as selection, crossover, and mutation or controlled the parameters of operators such as the operator probabilities [9, 10, 14, 17, 18].

On the other hand, this paper introduces a new operation termed competitive generation for accelerating the evolution speed of GAs. In original GAs, offsprings are generated by crossover and mutation operations from individuals selected as parents. In the competitive generation, however, individuals selected as parents first produce a specific number of candidates and then the best candidate selected from the candidates takes part in generating offsprings. The main idea of this competitive generation is devised from the competition of sperms in the process of fertilization. That is, in nature only the best sperm from a great number of sperms in fertilization process can be fertilized with an egg cell. It is sure that the competition of sperm will make the offsprings better because relatively poor sperms will not be mated. From this point of view, we will call this competitive generation sperm-level evolution. Similar to nature, competitive generation in GAs will provide a good chance to produce better offsprings to GAs in that this generation provides more chances to evolve to GAs and helps the individuals keep the diversity that prevents GAs from falling into local optimum. As a result, competitive generation can accelerate the optimization speed of GAs. The competitive generation has been tested with one combinational problem and five typical function optimization problems. In order to analyze the effects of selection methods, we experimented the competitive generation with roulette wheel selection and rank selection. It was shown from the experiments that the competitive generation could be a core operation in the processes of GAs for enhancing the performances of genetic algorithms. Since the competitive generation is a new operation that existing GAs haven't, it can be easily incorporated into existing GAs for improving their performances.

This paper is organized as follows. Section 2 describes competitive generation for genetic algorithms. In section 3, experimental parameters and results are discussed. This paper concludes in section 4.

2. Competitive generation

The genetic algorithm with competitive generation is described in Algorithm 1.

접수일자 : 2006년 11월 18일
완료일자 : 2006년 12월 19일

Algorithm 1 Genetic algorithm with competitive generation

```
   // t : time //
   // n : population size //
   // P : populations //
   // S : temporary individuals (sperms) //
   // m : the number of sperms //
1  t ← 0
2  initialize P(t)
3  evaluate P(t)
4  while (not termination-condition)
5  do
6      t ← t + 1
7      select P(t) from P(t − 1)
8      competitive generation (*)
9          for i = 1 to n
10             make m sperms for parent i
11             select the best sperm for mating
12             replace the parent i with the best sperm
13         end for
14     recombine P(t)
15         do crossover
16         do mutation
17     evaluate P(t)
18 end
```

Except for the competitive generation, the algorithm is the same as the original genetic algorithm (OGA) [1]. The newly added competitive generation are marked by asterisk in Algorithm 1. In original GA, parents $P(t)$ are directly recombined by crossover and mutation operations for creating offsprings. While parents $P(t)$ in the proposed GA are modified by competitive generation and recombined. As shown in the lines 9 to 13 in Algorithm 1, the m number of sperms for each parent are generated by some methods. For simplicity, in this paper, we use the same method as the mutation of GA. However, it is necessary to device more effective methods considering how the competitive generation affects to the performances of GAs as a further work. After that, the best sperm from the m number of sperms generated from a parent is selected. This is why we call our method competitive generation. Finally, the best sperm will be used as a parent instead of the original parent for recombination as if the first arrived sperm (we can view this sperm as the best sperm) to an egg cell is fertilized with.

This competitive generation provides some effects to the GA as shown in Fig. 1. First, it can give the GA some diversity that prevents from falling into a premature convergence. In the figure 1, the case (a) shows that an original parent depicted by a dark circle generates three sperms depicted by white circles. The sperm (3) will replace with the parent because its fitness is the best and this will help GA escape the local optimum. Second, as shown in the case (b) competitive generation accelerates the the optimization speed of genetic algorithms. From this observation it can be said that the competitive generation increases the optimization speed of GAs without falling into local optimum. This results in enhancing the performances of GAs.

3. Experimental results and discussion

The competitive generation was tested on one combinational problem and five function optimization problems. In the problems, f_1 is one dimensional problem while f_3 to f_6 are two dimensional problems. All functions except for f_2 are function optimization problems. On the other hand, the function f_2, a bit pattern matching problem, is a combinatorial optimization problem between the target pattern T and an individual's pattern I. If all bits between T and I are same, then it is optimum. Therefore, optimum fitness is same as the number of bits h. Functions f_3 to f_6 used in many other papers are DeJong function 2 (f_3), DeJong function 5 (f_4), Mexican hat function (f_5), Shafer function 2 (f_6), respectively. Figure 2 shows the input-output relations of six functions. Function f_3 is relative simple in that it has a few local optimum; unlike function f_3, function f_4 has many local optimum in separate spaces; functions f_5 and f_6 have many local optimum in continuous spaces. The optimum value of f_3 is only one at $x_1 = -2.048$ and $x_2 = -2.048$ and optimum value is about 3905.9. Similarly, function f_4 has only one optimum value (about 498.9) at $x_1 = -32.509984$ and $x_2 = -32.509984$. On the other hands, function f_5, and f_6 has multiple optima. Multiple optima of function f_5 are at the smallest circle of Mexican hat and its value about 0.99. Function f_6 has multiple optima at four peaks near ($x_1 = -10$ and $x_2 = -10$, $x_1 = -10$ and $x_2 = 10$, $x_1 = 10$ and $x_2 = -10$, $x_1 = 10$ and $x_2 = 10$) and its value is about 14.3.

We experimented with these functions on the parameters of Table 1. Most of parameters are general and typical as the original GA. The number of sperms k and the mutation probability p_g for competitive generation are the parameters for testing the performances of GA with competitive generation in compared to the original GA.

Since the GA generally shows somewhat different results according to the initial individuals, we measured experimental results with average values of 10 runs using different random number seeds. When the GA finds the optimal solution, the number of generations at that generation is recorded and the results of 10 runs are averaged. This averaged value is an experimental result for one experiment. Table 2 shows total experimental results of six problems. Table 2 shows the average (avg.) values and standard deviation (dev.) values of 10 runs where the GA finds the optimal solution according to the number of sperms k and

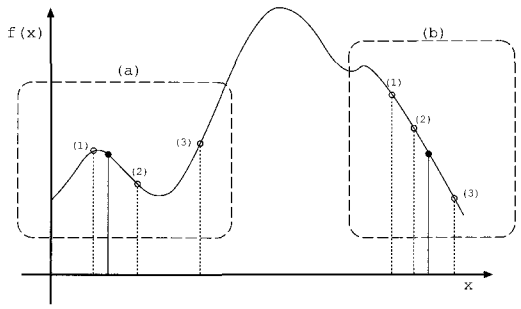


Figure 1: Effects of competitive generation

$$f_1 = 10x \cdot sgn(x)(sin(\pi x) + 1)(sin(10\pi x) + 1), \text{ where } -20 \leq x \leq 10 \quad (1)$$

$$f_2 = \sum_{j=1}^{h} m_j \begin{cases} m_j = 1 & \text{if } T_j = I_j \\ m_j = 0 & \text{if } T_j \neq I_j \end{cases} \quad (2)$$

$$f_3 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \text{ where } -2.048 \leq x_i \leq 2.048 \quad (3)$$

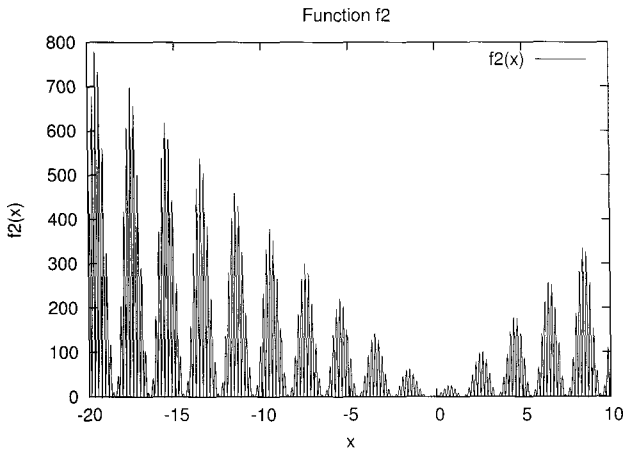
$$f_4 = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a[i][j])^6}, \text{ where } -65.536 \leq x_i \leq 65.536 \quad (4)$$

$$f_5 = 0.5 - \frac{sin(\sqrt{x_1^2 + x_2^2})sin(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))(1.0 + 0.001(x_1^2 + x_2^2))}, \text{ where } -10 \leq x_i \leq -10 \quad (5)$$

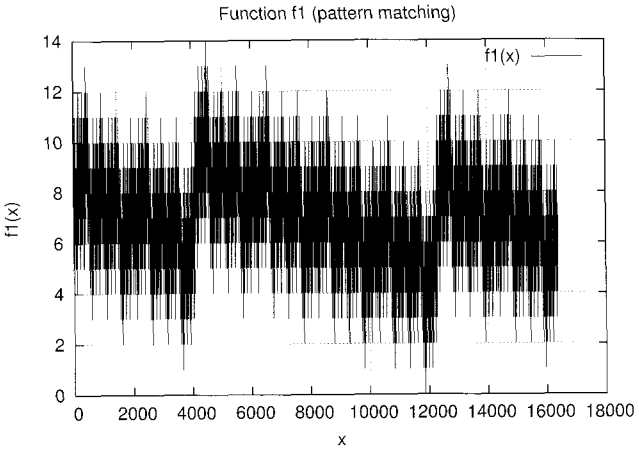
$$f_6 = (x_1^2 + x_2^2)^{0.25} sin(50(x_1^2 + x_2^2)^{0.1} + 1)^2, \text{ where } -10 \leq x_i \leq -10 \quad (6)$$

Table 1: Parameters for experiments

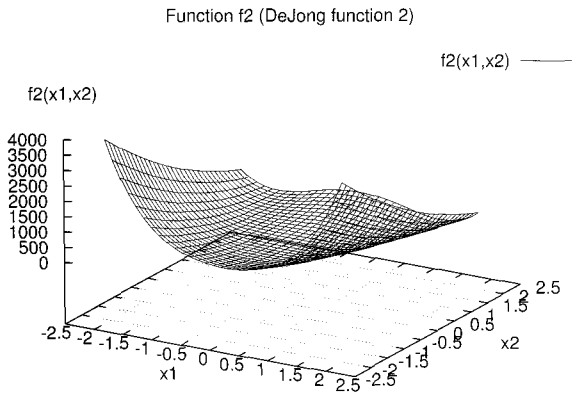
Parameters	Values
Selection method	roulette wheel selection
Crossover probability (p_c)	0.6
Mutation probability (p_m)	0.05
Population size	10
Individual length	24 bits
The number of sperms (k)	2, 6, 10
Competitive generation mutation probability (p_g)	0.02, 0.05, 0.07, 0.1, 0.2



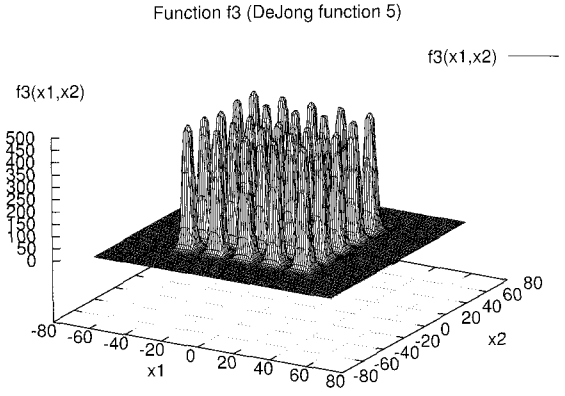
(a)



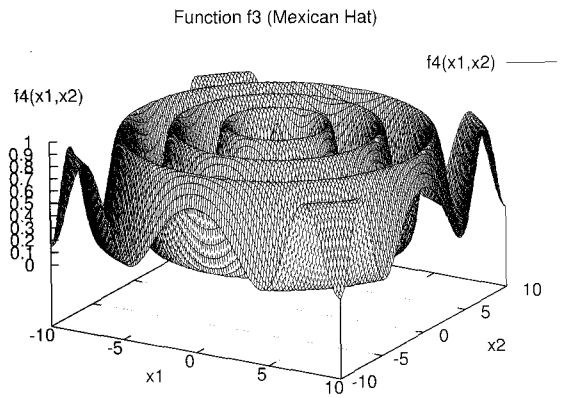
(b)



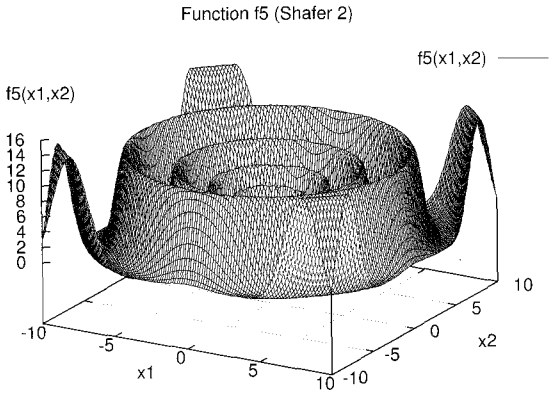
(c)



(d)



(e)



(f)

Figure 2: Experimental functions (a) f_1 (b) f_2 (c) f_3 (d) f_4 (e) f_5 (f) f_6)

Table 2: Experimental results

		f_1		f_2		f_3	
		avg.	dev.	avg.	dev.	avg.	dev.
OGA		47378.4	26575.9	64201.3	40495.1	59078.5	55933.1
2	0.02	25152.4	22779.8	1599.1	1383.6	3346.8	3882.1
	0.05	58970.4	63311.0	3197.6	2857.8	4532.4	4434.6
	0.07	63310.2	62717.2	7558.3	3949.9	25430.1	25178.1
	0.1	306679.9	311347.1	7449.3	9404.3	45448.3	52526.1
	0.2	810483.7	946069.9	82396.2	54927.1	284938.4	161217.7
6	0.02	567.0	388.3	46.7	41.5	1640.0	1108.6
	0.05	1265.0	987.5	35.1	29.2	357.3	338.7
	0.07	1860.1	1197.7	30.8	30.8	374.3	297.8
	0.1	3849.3	3820.5	103.8	81.4	905.9	511.7
	0.2	68172.1	44341.3	1594.9	983.4	20810.2	12345.9
10	0.02	699.3	887.7	26.9	10.3	536.2	531.2
	0.05	362.0	338.2	(*)12.5	3.7	(*)220.6	250.9
	0.07	(*)205.4	143.1	14.4	7.1	284.8	174.3
	0.1	762.7	600.0	26.7	18.1	650.1	918.1
	0.2	20784.3	22361.7	327.4	263.3	3876.5	5626.6
OGA / best		47378.4/205.4 = 230.6		64201.3/12.5 = 5136.1		59078.5/220.6 = 267.8	

		f_4		f_5		f_6	
		avg.	dev.	avg.	dev.	avg.	dev.
OGA		37401.3	37294.3	776085.3	624688.8	250111.1	228965.8
2	0.02	18719.0	17736.5	434323.2	475359.9	104297.6	90802.6
	0.05	22806.6	13873.4	781462.0	905776.8	264896.3	191795.2
	0.07	67024.3	46242.7	1527011.4	925622.6	424570.2	287383.2
	0.1	261524.5	200214.0	1361112.6	1739700.4	543164.6	271498.0
	0.2	421060.2	364690.1	630308.0	432581.1	961098.9	668125.2
6	0.02	3336.9	2601.6	32586.0	42511.3	17180.6	18253.9
	0.05	732.6	554.8	25389.4	25237.5	32139.3	34039.6
	0.07	1856.0	1227.0	31952.1	14917.9	15570.6	15681.7
	0.1	2731.2	2379.8	161460.6	200738.0	58941.6	38902.7
	0.2	84312.3	61745.9	301596.2	288467.5	262624.9	195144.5
10	0.02	1371.0	1646.6	6904.9	7783.8	17636.6	14668.9
	0.05	760.7	1026.3	(*)5124.4	4130.1	24674.5	15967.1
	0.07	524.0	377.9	9959.7	8877.3	(*)8347.9	6180.5
	0.1	(*)330.1	236.7	26276.4	30116.9	17570.5	16820.1
	0.2	29293.0	25479.4	304400.6	145141.8	95332.3	127174.5
OGA / best		37401.3/330.1 = 113.3		776085.3/5124.4 = 151.4		250111.1/8347.9 = 29.9	

Table 3: Experimental results

		f_1		f_2		f_3	
		avg.	dev.	avg.	dev.	avg.	dev.
OGA		46669.8	40810.1	206.9	185.2	15475.4	12953.2
2	0.02	25295.5	20438.7	83.6	53.4	1346.1	1102.0
	0.05	162601.0	138169.7	228.2	202.5	4251.0	2561.7
	0.07	156916.6	185730.7	1303.5	844.8	13860.0	11303.0
	0.1	267582.2	244769.9	4675.1	4220.7	71951.4	65032.7
	0.2	901772.4	587805.6	82381.1	49139.2	435488.7	428062.3
6	0.02	1037.2	854.7	15.5	6.6	765.5	692.9
	0.05	839.1	815.9	13.9	8.5	633.8	824.9
	0.07	1372.7	995.9	19.5	11.2	593.0	845.8
	0.1	5153.7	2461.9	48.9	39.9	479.7	353.2
	0.2	105438.3	53092.9	746.7	791.2	14734.5	26468.6
10	0.02	515.1	438.9	11.7	3.9	454.8	509.7
	0.05	385.6	293.1	(*)7.7	2.6	(*)332.0	410.5
	0.07	(*)189.6	134.0	9.6	2.6	412.0	444.0
	0.1	559.1	374.6	14.7	10.0	403.3	398.8
	0.2	26796.5	20985.8	349.1	354.0	7523.8	10573.3
OGA / best		46669.8/189.6 = 246.1		206.9/7.7 = 26.8		15475.4/332.0 = 46.6	

		f_4		f_5		f_6	
		avg.	dev.	avg.	dev.	avg.	dev.
OGA		55783.5	56581.1	857647.7	511527.9	204548.9	127878.3
2	0.02	18208.4	18611.3	184743.1	185260.8	196577.2	118380.5
	0.05	62863.1	48757.0	529270.8	859380.7	236654.9	189777.8
	0.07	108673.2	96464.8	731790.1	407055.2	431686.6	368738.3
	0.1	244618.3	321800.5	1365798.3	1613145.9	411378.0	380211.9
	0.2	893343.2	846905.2	1302722.6	913991.6	951899.6	640898.1
6	0.02	537.3	314.0	7999.6	5645.1	27244.8	27458.2
	0.05	499.1	327.0	7326.6	4683.6	27480.3	17884.0
	0.07	533.3	466.3	16489.4	14760.7	24153.3	17078.9
	0.1	3662.3	2946.6	96366.9	118033.3	33553.4	23951.0
	0.2	135573.7	92780.1	469603.2	487382.1	193434.7	177412.3
10	0.02	(*)172.7	91.6	6578.5	6199.3	30022.6	28821.5
	0.05	294.1	245.5	(*)2543.6	2359.1	8740.9	7496.7
	0.07	202.8	139.2	3099.6	2353.0	9822.9	12297.2
	0.1	308.2	284.6	14681.9	13563.0	(*)8355.1	6334.4
	0.2	25823.8	25952.6	249402.6	376404.8	99181.4	79940.5
OGA / best		55783.5/172.7 = 323.0		857647.7/2543.6 = 337.1		204548.9/8355.1 = 24.4	

Table 4: Summary of results

	roulette wheel selection		rank selection	
function	OGA	best	OGA	best
f_1	47378.4	205.4 (10/0.07)	46669.8	189.6 (10/0.07)
f_2	64201.3	12.5 (10/0.05)	206.9	7.7 (10/0.05)
f_3	59078.5	220.6 (10/0.05)	15475.4	332.0 (10/0.05)
f_4	37401.3	330.1 (10/0.10)	55783.5	172.7 (10/0.02)
f_5	776085.3	5124.4(10/0.05)	857647.7	2543.6(10/0.05)
f_6	250111.1	8347.9(10/0.07)	204548.9	8355.1(10/0.10)

the mutation probability of competitive generation p_g. In table 2, OGA means the original genetic algorithm and its performance is compared with the best result (marked by asterisk) of our method (show OGA / best). Note that the performance ratio between OGA and the best result is very large ranged from 29.9 to 5136.1 It is very natural that the performances of our method (the GA with competitive generation) are better and better as the number of sperms k increase intuitively because it gives more evolution chances to GAs. Its effect, however, is greatly larger than the additive computational efforts. Although we simply regard the additive computational efforts as k times when the number of sperms is k, the performances of our method increase about 3 times to 500 times than that of the original GA. Of course, practically additive computational efforts are not so large. In this experiment, our method shows the best performance at the f_2 combinatorial problem and medium performance at the f_1 and f_3 relatively simple problems. In considerably complex problems $f_4 \sim f_6$, the performance enhancement is little down.

In order to investigate the effect of selection method, we also test our method with rank selection unlike previous experiment using roulette wheel selection. Experimental results with rank selection are shown in Table 3. As shown in Table 3, most results are similar to those of the previous experiments. At functions f_2 and f_3, however, OGA with rank selection shows greatly better performances than that with roulette wheel selection. According to the selection method, the performances of GA are changed generally. However, the best performances of GA with competitive generation are very similar in the two experiments. This indicates that the GA with competitive generation is less sensitive to the selection method than OGA. This is one of the good features of competitive generation. We summarized the two results in Table 4 for clarity. The results of f_1 and f_6 are very similar to previous results. On the other hand, the performance improvements at function f_4 and f_5 are increased than those of the previous experiment. In most cases, the GA with competitive generation shows better results than OGA. From this we can conclude that the competitive generation makes GAs find global optimum more quickly.

4. Conclusion

In this paper, we proposed a new operator termed competitive generation for improving the performances of genetic algorithms. Competitive generation similar to nature provides more changes to evolve to the individuals in population pool. From this GAs employing competitive generation find the optimal solutions more quickly. We experimented the GA with competitive generation with one combinatorial problem and five function optimization problems. In order to show the effects of selection method, two selection methods, one is roulette wheel selection and another rank selection, are used. It was found from experiments that the GA with competitive generation could considerably increase the performances of genetic algorithms by fast evolution. Moreover the GA with competitive generation showed relatively steady performances with respect to the selection method. This is a good feature of our method. Proposed competitive generation can be easily applied to the other types of genetic algorithms for improving their performances.

Acknowledgement. This research was financially supported by Hansung University in the year of 2006.

References

[1] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison–Wesley, 1989.

[2] C. L. Karr and E. J. Gentry, "Fuzzy Control of pH Using Genetic Algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 1, pp. 46–53, Jan. 1993.

[3] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *IEEE Computer Magazine*, pp. 17–26, June 1994.

[4] J. L. R. Filho and P. C. Treleaven, "Genetic-Algorithm Programming Environments," *IEEE Computer Magazine*, pp. 28–43, June 1994.

[5] D. Beasley, D. R. Bull, and R. R. Martin, "An Overview of Genetic Algorithms: Part 1, Fundamentals," Technical Report obtained from http://home.ifi.uio.no/ jimtoer/GA_Overview1.pdf.

[6] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Transactions on Neural Networks*, vol. 5, pp. 3–14, Jan. 1994.

[7] H. Szczerbicka and M. Becker, "Genetic Algorithms: A Tool for Modelling, Simulation, and Optimization of Complex Systems," *Cybernetics and Systems: An International Journal*, vol. 29, pp. 639–659, Aug. 1998.

[8] R. Yang and I. Douglas, "Simple Genetic Algorithm with Local Tuning: Efficient Global Optimizing Technique," *Journal of Optimization Theory and Applications*, vol. 98, pp. 449–465, Aug. 1998.

[9] C. Xudong, Q. Jingen, N. Guangzheng, Y. Shiyou, and Z. Mingliu, "An Improved Genetic Algorithm for Global Optimization of Electromagnetic Problems," *IEEE Transactions on Magnetics*, vol. 37, pp. 3579–3583, Sept. 2001.

[10] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, pp. 126–142, Apr. 2005.

[11] V. K. Koumousis and C. Katsaras, "A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 19–28, Feb. 2006.

[12] L. Davis, "Adapting Operator Probabilities in Genetic Algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms and their Applications*, pp. 61–69, 1989.

[13] R. Hinterding, Z. Michalewicz, and A. E. Eiben, "Adaptation in Evolutionary Computation: A Survey," in *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pp. 65–69, 1997.

[14] A. Tuson and P. Ross, "Adapting Operator Settings In Genetic Algorithms," *Evolutionary Computation*, vol. 6, no. 2, pp. 161–184, 1998.

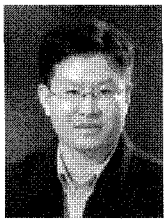
[15] C. W. Ho, K. H. Lee, and K. S. Leung, "A Genetic Algorithm Based on Mutation and Crossover with Adaptive Probabilities," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1, pp. 768–775, 1999.

[16] S. H. Jung, "Self-tuning of Operator Probabilities in Genetic Algorithms," *전자공학회 논문지*, vol. 37, pp. 29–44, Sept. 2000.

[17] R. Hinterding, "Gaussian Mutation and Self-adaption in Numeric Genetic Algorithms," in *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, pp. 384–389, 1995.

[18] J. Andre, P. Siarry, and T. Dognon, "An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization," *Advances in engineering software*, vol. 32, no. 1, pp. 49–60, 2001.

저 자 소 개



Sung Hoon Jung

1991년~1995년 : 한국과학기술원 전기 및 전자공학과(공학박사)
1995년~1996년 : 한국과학 기술원 전기 및 전자공학과 위촉연구원
1996년~1998년 : 한성대학교 정보통신 공학과 전임강사

1998년~2002년 : 한성대학교 정보통신공학과 조교수
2002년~현재 : 한성대학교 정보통신공학과 부교수

관심분야 : 진화연산, 신경망, 퍼지, 시스템생물학, 생물지능
E-mail : shjung@hansung.ac.kr