International Journal of Fuzzy Logic and Intelligent Systems, vol.10, no. 2, June 2010, pp. 146−151

Rank-based Control of Mutation Probability for Genetic Algorithms

Sung Hoon Jung[1]

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

Abstract

This paper proposes a rank-based control method of mutation probability for improving the performances of genetic algorithms (GAs). In order to improve the performances of GAs, GAs should not fall into premature convergence phenomena and should also be able to easily get out of the phenomena when GAs fall into the phenomena without destroying good individuals. For this, it is important to keep diversity of individuals and to keep good individuals. If a method for keeping diversity, however, is not elaborately devised, then good individuals are also destroyed. We should devise a method that keeps diversity of individuals and also keeps good individuals at the same time. To achieve these two objectives, we introduce a rank-based control method of mutation probability in this paper. We set high mutation probabilities to lowly ranked individuals not to fall into premature convergence phenomena by keeping diversity and low mutation probabilities to highly ranked individuals not to destroy good individuals. We experimented our method with typical four function optimization problems in order to measure the performances of our method. It was found from extensive experiments that the proposed rank-based control method could accelerate the GAs considerably.

Key words : Genetic algorithms, premature convergence phenomena, function optimization, rank-based control

1. Introduction

Genetic Algorithms(GAs) have been applied to so many scientific and engineering problems especially for optimization problems [1–11]. As the number of application increases, the performances of GAs are also more and more important. One of the fatal problems of GAs is the premature convergence problem [1, 2, 4, 5, 8–10, 12, 13]. If GAs fall into premature convergence phenomena that most individuals are located within local optimum areas, their performances are significantly degraded [7, 8, 10, 14–17]. Crossover operations can not help GAs get out of the phenomena because they can make offsprings only near the local optimum areas and mutation operations can not also provide a solution because the mutation probability is very low. If we set the mutation probability to a high value, then it will help GAs escape the local optimum areas but GAs can not approach to the global optimum because they act like a random search.

In order to deal with this dilemma, various parameter control methods have been introduced [7, 10, 14, 15, 17–21]. These methods can be classified into three categories: deterministic methods, adaptive methods, and self-adaptive methods. Deterministic methods are to adjust the parameters using deterministic equations without considering the search process such as the fitness of individuals or distribution of individuals. In adaptive methods, the parameters are changed according to the fitness of individuals or distribution of individuals. The control parameters are encoded into the chromosomes as a part of chromosomes and evolved at the same time in self-adaptive methods. Deterministic methods are simple, but not effective because they don't care the search process and self-adaptive methods are somewhat effective, but they are considerably complex than the adaptive methods and result in sometimes making the GAs degrade. Adaptive methods are simple and effective, but additional parameters should be carefully selected. If not, the performances are sometimes degraded. Most existing adaptive methods have employed the fitness of individuals for control the mutation probability. Since the fitness has various ranges according to the optimized functions, additional parameters are necessary. If the parameters are inadequately set, then the performances are not good.

In this paper, we introduce a new control method of mutation probability based on the ranks, not fitness of individuals. In order to keep diversity of individuals and to keep good individuals, we set high mutation probabilities to lowly ranked individuals and low mutation probabilities to highly ranked individuals at every generations. The best individual has zero mutation probability and the worst individual has the highest mutation probability. In order to

Manuscript received Jan. 17, 2010; revised Apr. 30, 2010;
Accepted Apr. 30, 2010.

Corresponding Author : shjung@hansung.ac.kr (Sung Hoon Jung)

This research was financially supported by Hansung University
in the year of 2009.

measure the performances of our algorithm, we extensively experimented our method with typical four function optimization problems under various parameters. From these experiments we found that our method was superior to the existing GAs. Our method can be easily employed to many types of existing GAs to increase their performances without many modifications.

This paper is organized as follows. Section 2 describes previous control methods and our rank-based control method of mutation probability. Experimental results and discussion are provided in section 3. We conclude our paper in section 4.

2. Rank-based Mutation Probability Control

This section first introduces the premature convergence problem and previous works for control of mutation probabilities not to fall into premature convergence and next describes our rank-based mutation control method. Although the other methods to solve the premature convergence problems exist, we only focus on the control methods of mutation probability.

2.1 Premature Convergence Problem

Premature convergence is the problem that relatively good individuals within local optimum areas at initial generations are selected as parents and repeatedly regenerated with small modifications, as a result do not approach global optimum areas [1, 2, 4, 5, 8–10, 12, 13, 20, 21]. If GAs fall into premature convergence phenomena, then it is very difficult for individuals to get out of the local optimum areas [7, 8, 10, 14–17]. Under the premature convergence, most individuals are very similar each other because they are located nearly within local optimum areas. Crossover operations of GAs can not provide a lot of affects to get out the problem because offsprings after crossover operations are very similar to their parents. Whereas mutation operations can change the individuals very much, so it helps GAs get out of local optima. But it is not possible because the mutation probability is generally very low, typically 0.05 or less than that. If we set the mutation probability to a high value to overcome this, then GAs are hard to approach global optima because most building blocks are also destroyed. Actually GAs act as a random search as the mutation probability increases. It is important to keep the diversity not to fall into premature convergences, but it is also important to keep good individuals. If we use a constant mutation probability as the original GA, this is a dilemma because we cannot set the mutation probability to low values and also cannot set to high values. From this, control methods of mutation probability during run have been introduced [2, 8–10, 12, 13, 20, 21].

2.2 Previous Works

Existing methods can be classified into three categories: deterministic methods, adaptive methods, and self-adaptive methods. One of the simple deterministic methods is introduced by Ho *et al.* [17]. As we already mentioned above, premature convergence occurs at initial generations. From this fact, they employed a very simple equation as follows.

$$p_m = 0.1 - 0.09 \times \frac{g}{G},$$

where p_m is the mutation probability, g is the number of generation, and G is a user setting generation. If $g = 0$, then the $p_m = 0.1$ and the p_m is continuously decreased as the generation goes, and if $g = G$, then the $p_m = 0.01$. This method is very simple, so computationally effective, but its performance is not good because they don't care the search situation how individuals are converged into local optimum areas.

From this observation, adaptive methods have been introduced [2, 8–10, 13]. Srinivas *et al.* [2] proposed a control method of mutation probability using some measures of fitness as:

$$p_m = \begin{cases} k_2 \frac{(f_{max} - f)}{(f_{max} - \bar{f})}, & f \geq \bar{f} \\ k_4, & f < \bar{f} \end{cases}$$

where k_2 and k_4 are the user selected parameters and f, $\bar{f}$, f_{max} are the fitness of an individual, average fitness, and maximum fitness, respectively. When GAs fall into premature convergence, the average fitness $\bar{f}$ will be closed to the maximum fitness f_{max} and as a result, p_m is increased. If a good individual with good fitness closed to maximum fitness f_{max}, then the p_m becomes small, otherwise the p_m is large. Even if the p_m becomes to high when an individual has low fitness, it can not be enough to get out of the premature convergence. For this, they introduced the parameter k_4. The individuals with low fitness than the average fitness can be strongly mutated by k_4. Unlike the first method, this method showed relatively good performances because they used a measure of premature convergence using fitness. Since this method used fitness as the measure of premature convergence, the p_m variation is large according to the optimized functions. This is one of disadvantage of this method.

Another adaptive method relatively recently introduced used another measure of premature convergence [21]. They first calculated all distances between individuals as follows.

$$d(i,j) = \sqrt{(g_i(1) - g_j(1))^2 + \cdots + (g_i(N) - g_j(N))^2}$$

where the $d(i,j)$ is the distance between individuals i and j, $g_i(1)$ is the first gene of individual i and so on, and N is

the number of genes. Using this distance, the diversity div can be obtained as follows.

$$div = \frac{\sum_{i=1}^{N}\sum_{j=i+1}^{N} 1_{d(i,j)>D}}{N^2}$$

where the D is a threshold value user selects and N is the number of genes. From this equation, we can know the diversity of individuals. If GAs fall into premature convergence, then the diversity div will be small. With this equation, the mutation probability is controlled as follows.

$$p_m = p_m^0 - div \cdot p_m^0$$

where p_m^0 is the constant mutation probability. If all individuals are same, then the diversity div becomes to zero and $p_m = p_m^0$. If div is large, then the p_m becomes to small. Since this method directly measured the diversity using all individuals, a lot of computation $O(n^2)$ is necessary especially for the large N cases. This is a main drawback of this method.

In self-adaptive methods, the mutation probability is also encoded into chromosomes as part of chromosomes. When an individual is mutated, then the encoded mutation probability is applied. This method is based on the assumption that if the mutation probability is proper, then the offsprings are good and alive at next generation. This method is somewhat complex and since the quality of offsprings does not depend on only the mutation, so the good offsprings alive don't have always good mutation probability.

In this paper, we introduce rank-based control method of mutation probability. Our method is simple, but effective in the viewpoints of computation and measure of premature convergence.

2.3 Proposed Rank-based Control Method

In order to control mutation probability based on ranks of individuals, we first evaluate ranks of individuals and then calculate new mutation probability of each individual according to its rank. Algorithm 1 shows the GAs with rank-based mutation probability control.

Algorithm 1 Proposed Genetic Algorithm

// t : time //
// n : population size //
// p_m^o : constant original mutation probability //
// p_m^i : mutation probability of ith individual //
// ζ : mutation scale factor //
// r_i : rank of ith individual //
// P : populations //

t ← 0
initialize $P(t)$
evaluate $P(t)$
while (**not** termination-condition)
do
    t ← t + 1
    select $P(t)$ from $P(t-1)$
    recombine $P(t)$
        do crossover
        evaluate $P(t)$ (*)
        rank $P(t)$ (*)
        calculate mutation probability (*)
            for $i = 1$ **to** n
                $p_m^i = (p_m^o + p_m^o) * \zeta * r^i/(n-1)$
            end for
        do mutation with p_m^i instead of p_m^o (*)
    evaluate $P(t)$
end

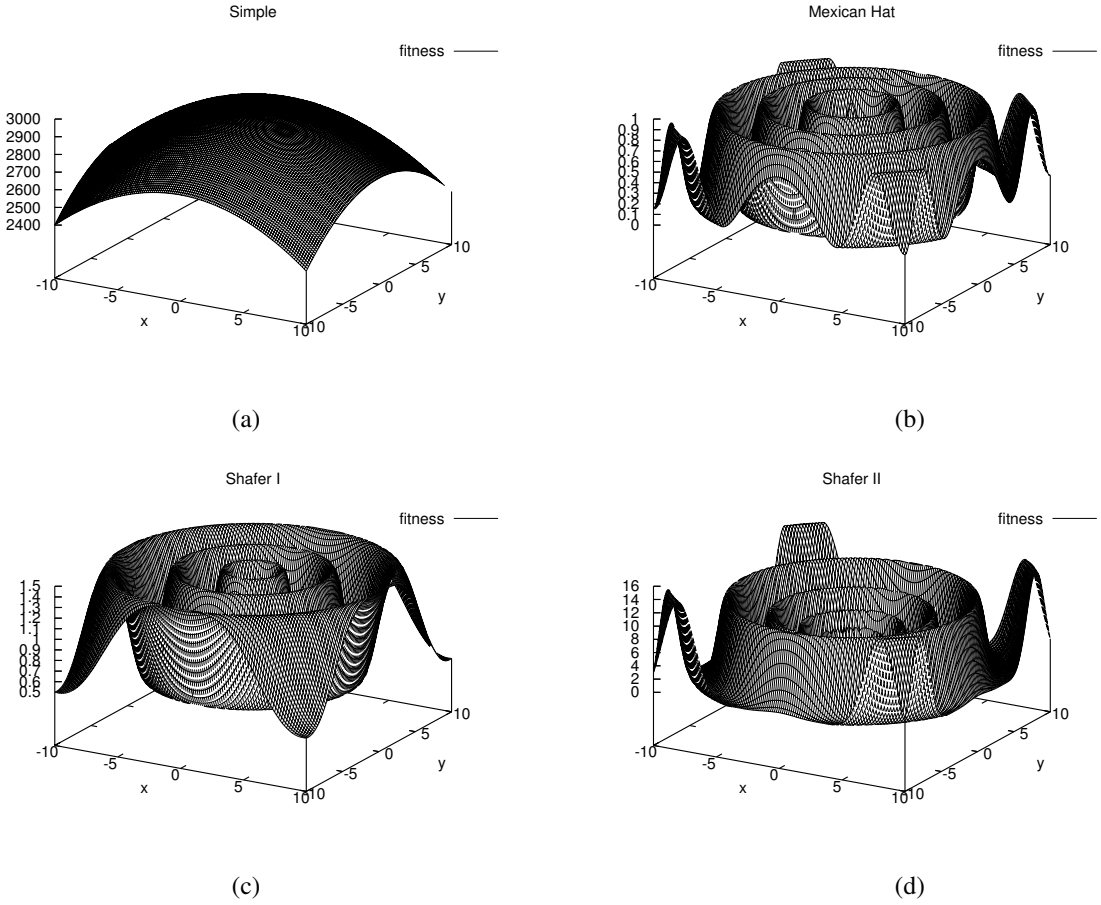
In algorithm 1, (*) indicates additional operations for rank-based mutation probability control. Except for those, the others are the same as the original GA. After the crossover operation, evaluation should first be done in order to decide ranks of individuals and then individuals are ranked using their fitness. After that, new mutation probability of each individual is calculated with the original mutation probability p_m^o, the mutation scale factor ζ, and the rank of the individual r^i. Since the rank is provided from zero to $n-1$, the new mutation probability is scaled up from 0 to $2p_m^o * \zeta$. If $\zeta = 1$, then the new probabilities of the highest ranked individual, the middle ranked individual, and the lowest ranked individual becomes 0, p_m^o, $2p_m^o$, respectively. According to the mutation scale factor ζ, the mutation probability is scaled up.

From this rank-based control of mutation probability, we can keep the diversity of individuals not to fall into premature convergence by the strong mutation of lowly ranked individuals and can ensure the highly ranked individuals against destroying their good building blocks by the zero or small mutation. Our algorithm is relatively simple than the previous methods because our method uses ranks instead of fitness and doesn't need large additional parameters except for the mutation scale factor ζ. The mutation scale factor ζ is not essential to our algorithm, it can be simply set to one because it doesn't affect to the performances of GAs largely. The original mutation probability p_m^o is generally set to 0.05, but it can be set by empirically or the other parameter tuning methods.

3. Experimental Results

We experimented our algorithm with four function optimization problems. The four functions are given in Equation 1.

$$\begin{aligned} f_1 &= 3000 - 3(x^2+y^2) \\ f_2 &= 0.5 - \frac{sin(\sqrt{x^2+y^2})sin(\sqrt{x^2+y^2})-0.5}{(1.0+0.001(x^2+y^2))(1.0+0.001(x^2+y^2))} \\ f_3 &= 0.5 + sin^2(\frac{(\sqrt{x^2+y^2}-0.5)}{(1+0.001(x^2+y^2))^2}) \\ f_4 &= (x^2+y^2)^{0.25}sin^2(50(x^2+y^2)^{0.1}+1) \end{aligned} \quad (1)$$



(a) (b)

(c) (d)

Fig. 1 Experimental functions (a) f_1 (b) f_2 (c) f_3 (d) f_4)

Table 1. Parameters for experiments

Parameters	Values
Selection method	roulette wheel selection (normalized fitness: 1000)
Crossover probability (p_c)	0.6
Mutation probability (p_m)	0.01, 0.05, 0.1, 0.2
Population size	20
Individual length	22 bits
Mutation scale factor (ζ)	0.5, 1, 2, 4, 8, 16

Table 2. Experimental results

fn.	p_m	OGA	DGA	PGA (ζ)						OGA/best
				0.5	1	2	4	8	16	
f_1	0.01	57382.8	94807.8	75198.3	35448.9	19908.5	12600.6	8961.2	11107.6	6.4
	0.05	32033.1	32050.2	17930.3	9144.7	16443.3	11141.5	9810.7	20141.9	3.5
	0.1	49476.1	58760.6	9144.7	16443.3	11141.5	9810.7	20141.9	47161.3	5.4
	0.2	55405.3	76840.2	16443.3	11141.5	9810.7	20141.9	47161.3	126467.8	5.6
f_2	0.01	88394.5	132789.9	84454.2	25273.8	9553.0	6598.0	7068.2	6927.1	13.4
	0.05	36609.9	28308.7	15167.3	7406.5	10713.2	9964.4	10086.2	15324.6	4.9
	0.1	45267.4	26433.0	7406.5	10713.2	9964.4	10086.2	15324.6	23622.1	6.1
	0.2	53482.2	40627.1	10713.2	9964.4	10086.2	15324.6	23622.1	100911.1	5.4
f_3	0.01	204.8	215.1	264.3	120.8	113.9	72.3	41.3	39.6	5.2
	0.05	96.3	230.3	175.9	23.5	26.1	76.8	74.1	89.5	4.1
	0.1	249.9	259.0	23.5	26.1	76.8	74.1	89.5	211.2	10.6
	0.2	141.5	252.6	26.1	76.8	74.1	89.5	211.2	348.6	5.4
f_4	0.01	1360.4	2600.5	5203.7	1533.6	679.3	615.2	427.8	330.3	4.1
	0.05	1124.2	1543.9	1175.4	565.3	522.3	405.0	328.8	917.1	3.4
	0.1	2383.9	3156.4	565.3	522.3	405.0	328.8	917.1	4033.5	7.3
	0.2	8605.9	4449.9	522.3	405.0	328.8	917.1	4033.5	11600.6	26.2

Functions $f_1 \sim f_4$ are a simple function, a Mexican hat function, a Schafer function 1, and Schafer function 2, respectively. Fig. 1 shows the input-output relations of four functions. Function f_1 is relative simple, but the others has a lot of local optima. We experimented with typical parameters as shown in Table 1. Since the roulette wheel selection uses the fitness of individuals as the measure of goodness, the selection pressure of good individuals depends on the values of fitness of test functions. In order to eliminate this effect, we normalized the fitness to 1000. That is, the best individual in each generation has the fitness of 1000. We tested original genetic algorithm(OGA) [1], the existing genetic algorithm using diversity (DGA) [21], and our proposed genetic algorithm(PGA). Since the performances of GA depends on the initial individuals, we tested 10 runs and averaged the results. At each run, if GA finds a global optimum, then the number of generations at that time is recorded and these numbers are averaged. Table 2 showed the experimental results. For simplicity, we describe only average values without standard deviation values. In Table 2, the underbars indicate the best results on the parameters and the numbers at the eleventh column mean the performance ratio between OGA and the best of PGA.

As shown in Table 2, our algorithm is superior to the OGA and DGA. The DGA showed better results than the OGA in some cases especially for the function f_2, but it had worse results than the OGA for the other functions except for only one case of f_4 with $p_m = 0.2$. It was shown from this experiment that the DGA method is not effective to increase the performances of GAs. We experimented the DGA method with the diversity threshold value $D = 1/3 * l$ where l is the individual length. The other value of D may show different results. However, the other value $D = 1/2 * l$ also showed similar results, but not included in the table. The mutation scale factor ζ had broad ranges for the best results. For the relatively simple function f_1, the ζ showing best results had relatively small values, but it had large values for the complex and multimodal function f_4. Unlike the functions of f_2 and f_3 that have only one global optimum at the center of areas, global optimum areas of function f_4 are distributed at four areas. We can guess from this that the strong mutation scale factor provides good effects to the function of f_4. Although the results of PGA according to the mutation scale factors are somewhat different, the differences of results are not large and they are better than the OGA and DGA.

These results are because our algorithm can search more broad ranges using bad individuals and focus on the best places using good individuals. From this, our algorithm rarely falls into premature convergence and easily gets out of the premature convergence if it falls into without destroying good individuals. These effects increase the performances of our algorithm.

4. Conclusion

In this paper, we introduced a new rank-based control method of mutation probability. This rank-based control method makes bad individuals with low fitness to keep the diversity of population and also makes good individuals with high fitness not to be destroyed. As a result, rank-based control of mutation probability can decrease the

probability of falling into premature convergence without any negative side effects and results in increasing the performances of GAs. It was proved with four function optimization problems, our genetic algorithm was superior to the original genetic algorithm and the previous genetic algorithm using a diversity measure. Our method is very simple, so it can easily applied to the other genetic algorithms.

References

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

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

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

[4] 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.

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

[6] 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.

[7] S. M. Libelli and P. Alba, "Adaptive mutation in genetic algorithms," *Soft Computing*, vol. 4, pp. 76–80, 2000.

[8] 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.

[9] 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.

[10] A. E. Eiben, Z. Michalewicz, m. Schoenauer, and J. E. Smith, "Parameter Control in Evolutionary Algorithms," *Studies in Computational Intelligence*, vol. 54, pp. 19–46, 2007.

[11] Z. Jinhua, Z. Jian, D. Haifeng, and W. Sun'an, "Self-organizing genetic algorithm based tuning of PID controllers," *Information Sciences*, vol. 179, pp. 1007–1018, 2009.

[12] J. A. Vasconcelos, J. A. Ramirez, R. H. C. Takahashi, and R. R. Saldanha, "Improvements in Genetic Algorithms," *IEEE Transactions on Magnetics*, vol. 37, pp. 3414–3417, Sept. 2001.

[13] 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.

[14] 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.

[15] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, pp. 656–667, Apr. 1994.

[16] A. Tuson, "Adapting Operator Probabilities in Genetic Algorithms," master thesis, Dept. of Artificial Intelligence, University of Edinburgh, UK, 1995.

[17] 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.

[18] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft computing : a fusion of foundations, methodologies and applications*, vol. 92, no. 2, pp. 81–87, 1997.

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

[20] S. Meyer-Nieberg and H.-G. Beyer, "Self-Adaptation in Evolutionary Algorithms," *Studies in Computational Intelligence*, vol. 54, pp. 47–75, 2007.

[21] Z. Tang, Y. Zhu, G. Wei, and J. Zhu, "An Elitist Selection Adaptive Genetic Algorithm for Resource Allocation in Multiuser Packet-based OFDM Systems," *Journal of Communications*, vol. 3, pp. 27–32, July 2008.

Sung Hoon Jung
Professor of Hansung University
Research Area: Evolutionary Computation, Neural Network, Fuzzy, Systems Biology
E-mail : shjung@hansung.ac.kr