Rank and Proximity Based Crossover (RPC) to Improve Convergence in Genetic Search

Goutam Chakraborty
Iwate Prefectural University
152-52 Azasugo, Takizawamura
Iwate, Japan 020-0193
goutam@soft.iwate-pu.ac.jp

Basabi Chakraborty
Iwate Prefectural University
152-52 Azasugo, Takizawamura
Iwate, Japan 020-0193
basabi@soft.iwate-pu.ac.jp

Abstract- A new crossover technique, we named Rank and Proximity Based Crossover (RPC), to improve the speed and quality of solutions in Genetic search, is proposed. In the proposed strategy, the probability of crossover is more when the rank of two chromosomes are both high, and they are closely located in the search space. This probability is again a function of the generation number. In the early stage of genetic search the crossover is independent of the rank and proximity of the partners. Thus, crossover takes place between any two chromosomes selected for crossover, to enable exploration of diverse locations of the problem space. With advancing generations, RPC probabilistically encourages crossover between chromosomes of higher ranks which are closely located in the search space. This ensures avoidance of disruption of good chromosomes by crossover, when prospective locations are found, and thus achieves much faster convergence. Different schemes of this probability function are tried and evaluated, and convergence efficiency is compared with other competitive algorithms.

Keywords:- Genetic Algorithm, Crossover, Fitness, Selection pressure

1 Introduction

"Genetic Algorithm (GA) is a search algorithm based on the mechanics of natural selection and natural genetics" [1]. They are superior because, (1) of wide applicability and make few assumption from the problem domain, (2) and are not biased towards local minimums. At the same time GAs are very efficient to direct the search towards relatively prospective regions of the search space.

In GA, one has to generate a pool of initial encoded solutions (also called chromosomes) of the problem. A fitness function has to be defined to measure the goodness of those encoded solutions. Then genetic operators *selection, crossover* and *mutation* operate on the population to generate new population (new set of solutions) from the old ones. Good solutions are *selected* with greater probability to next generation, in line with the idea of *survival of the fittest*. *Crossover* operation recombines selected solutions, by swapping part of them, producing divergent solutions to explore the search space. An occasional *mutation* is done by flipping the value at random position of the encoded chromosome, to facilitate jumping of solutions to new unexplored regions. As the algorithm continues and newer and newer generations evolve, the quality of solutions improve.

Many strategies for fitness calculation, selection, crossover and mutation are proposed. In Standard Genetic Algorithm (SGA), after selection, a subset of chromosomes are randomly chosen for crossover, and forms what is called crossover pool. Pairs of members from the crossover pool are randomly chosen and part of the encoded strings are swapped, until all members of crossover pool are crossed over. This new crossed over members along with the rest of those selected form the previous population form the new generation. Occasionally a bit from a randomly selected chromosome is flipped, with a probability p_m, to jump to yet unexplored regions of the search space.

For success of GA the two aspects of (1) population diversity i.e. to explore the different regions of the search space, and (2) selective pressure i.e. to get to the optimum point in a region, have to be properly taken care of. In SGA, the best few members of the initial population could predominate the whole population in a few cycles due to their much better fitnesses and therefore high chance of getting selected. This would result in poor exploration and premature convergence to suboptimal minimum. On the other hand, at the later stage of the search, when the high performance regions are identified, fine local tuning is necessary to get to the solution, especially for high precision problems. It is difficult to achieve this by SGA because of disruption of good chromosomes after crossover with the bad ones situated at distant locations of the search space. The idea proposed here is to improve this situation.

A number of strategies were proposed [2](chapter 4 and 6) to overcome this problem by setting a balance between diversity (during the beginning) and selection pressure (at the end). We briefly discuss this before introducing our ideas for improving selection pressure.

One of the early proposal was to scale the fitness function [1] (pp. 122-124) as we go from initial to final stage of genetic search. To sustain diversity in the beginning, the fitnesses are scaled down so that the influence of high fitness is diminished in the selection stage. At the later stage of search, when most of the chromosomes have similar and high fitnesses, a reverse scaling is done to accentuate the effect of higher fitness and thus facilitating selection of only best chromosomes for faster convergence.

Ranking of the chromosomes according to their fitnesses, and not using the exact values of their fitnesses for selection, is another way of scaling of fitnesses throughout all generations to achieve the same goal.

In the approach named non-uniform mutation [4], at later stage of generations mutations are probabilistically performed more towards the tail part of the coded chromosomes. This is to avoid disrupting a good chromosome from its present location by changing bits at the head part (assuming that they are more significant bits). Another class of proposals is to adaptively change the crossover (p_c) and mutation (p_m) probabilities [3] [5] [6] [7]. In [6] the best few chromosomes are disrupted with much less probability than those with weak fitnesses. Thus the weak chromosomes are used for exploration of different regions, and the good ones to find the optimum locations.

We here propose a new crossover strategy and named it rank and proximity based crossover (RPC). In our previous work [9], we proposed only rank based crossover - where the results were not much better than fitness-scaling. In this RPC strategy, though two chromosomes are randomly chosen for crossover, the probability that the crossover actually takes place, depends on how far they are in the fitness scale, and their relative proximity in the search space. The nearer they are, more is the probability that they will be crossed over. Initially this range of nearness covers the whole of the fitness scale and search area, so that all randomly chosen pairs are crossed over, to allow exploration of different regions of the search space. Slowly when good regions are discovered, we allow crossover between chromosomes of similar rank only when their fitnesses are high. Then good solutions recombine fast to find the optimum location. The probability function changes with generations. Under this scheme, every time two chromosomes are picked up for crossover, the actual crossover may not be executed. The number of crossovers tried (X_t) and the number of crossovers executed (X_e) are different. During initial generations $X_t \approx X_e$, but at the end $X_t \gg X_e$. In the simulation we set $X_e = p_c \times P$, where p_c is the probability of crossover and P is the population size.

As we see, RPC does not improve over SGA for exploration. It improves the speed of convergence and thereby quality of solution due to stronger selection pressure at the end generations. RPC also facilitates tuning this range very easily. It has to be kept in mind that though the motivation of scaling, adaptive p_c and the proposed RPC are same, these three strategies are not competitive and actually all could be simultaneously used to improve the performance of genetic search.

In the next section 2 we describe in detail our proposed *rank and proximity based crossover* (RPC). In section 3 we present experimental results to show that when RPC strategy is used genetic search could deliver better results more efficiently, compared to standard genetic algorithm (SGA), SGA with fitness scaling. Conclusions and future research plans are discussed in section 4.

2 Rank Based Crossover

2.1 The RPC algorithm

The basic steps of the algorithm of genetic search using RPC is described in pseudocode below. First the notations are explained.

g **:** generation number.
G **:** maximum generation.
P **:** population size.
m_i **:** i^{th} member of the population.
f_i, f_i^{nor} **:** fitness and normalized fitness of m_i.
$\Pi(g)$ **:** set of chromosomes at generation g.
$\Pi''(g)$ **:** set of chromosomes after selection.
$\Pi'(g)$ **:** set of chromosomes after crossover.
$p_c,\ p_m$ **:** probabilities of crossover and mutation.
X_e **:** number of crossovers executed.
$\varphi^1()$, $\varphi^2()$, and $|\quad|$: functions are explained below.

```
Algorithm **RPC** ( $g, G, \Pi(g), P, p_c, p_m$ )
01 **begin**
02   $g = 0$;
03   Create initial population $\Pi(0)$;
04   Fitness evaluation of $m_i \in \Pi(0)$;
05   **while** ( $g \leq G$ )
06     $g := g + 1$;
07     $\Pi''(g) \xleftarrow{selection} \Pi(g-1)$;
08     **while** ( $X_e < p_c \times P$ )
09       Select $m_i(g),\ m_j(g)$ randomly from $\Pi''(g)$;
10       **if**($f_i^{nor}(g)$ *is high*)
11         **if**(($\varphi^1(f_i^{nor}(g) - f_j^{nor}(g)) > rand(0,1)) \wedge$
           $(\varphi^2\ (|\ m_i(g) - m_j(g)\ |) > rand(0,1)))$
12           $m_i(g)$ and $m_j(g)$ are crossed over;
13       **else if**(($\varphi^1(f_i^{nor}(g) - f_j^{nor}(g)) > rand(0,1))$
14           $m_i(g)$ and $m_j(g)$ are crossed over;
15         **endif**
16       **endif**
17     **endwhile**
18     $\Pi(g) \xleftarrow{mutation} \Pi'(g)$;
19   **endwhile**
20 **end**
```

The basic steps of RPC strategy differ from SGA from steps 08 to 17, in the process of crossover decision. First the fitness of all members in $\Pi''(g)$ are normalized to a value from 0 to 1. If $f_i(g)$ be the fitness of member $m_i(g)$, the normalized fitness denoted by $f_i^{nor}(g)$ is,

$$f_i^{nor}(g) = \frac{f_i(g) - f^{min}(g)}{f^{max}(g) - f^{min}(g)}$$

where, $f^{max}(g)$ and $f^{min}(g)$ are the maximum and the minimum fitness of all the members of $\Pi''(g)$.

We denote the normalized distance $|\ m_i(g) - m_j(g)\ |$ between two members of the population, m_i and m_j, as follows. Suppose the search space is d-dimensional. We represent the chromosomes m_i and m_j as:

$m_i = \langle l_{1i}\ \ l_{2i}\ \ l_{3i}\ \ \ldots\ \ l_{ki}\ \ \ldots\ \ l_{di}\ \rangle$
$m_j = \langle l_{1j}\ \ l_{2j}\ \ l_{3j}\ \ \ldots\ \ l_{kj}\ \ \ldots\ \ l_{dj}\ \rangle$

and

$$|\ m_i(g) - m_j(g)\ | = \sum_{k=1}^{d} \frac{(l_{ki} - l_{kj})}{r_k} / d$$

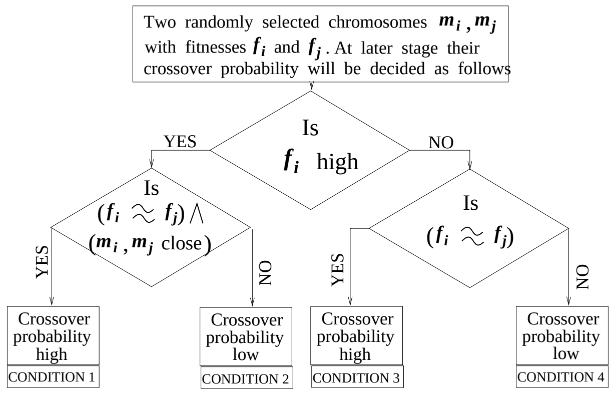
Two randomly selected chromosomes m_i, m_j with fitnesses f_i and f_j. At later stage their crossover probability will be decided as follows
Is f_i high
YES
NO
Is $(f_i \approx f_j) \wedge (m_i, m_j$ close)
Is $(f_i \approx f_j)$
YES
NO
YES
NO
Crossover probability high
CONDITION 1
Crossover probability low
CONDITION 2
Crossover probability high
CONDITION 3
Crossover probability low
CONDITION 4

Figure 1: Crossover probability with RPC: effective only at end part of the genetic search. CONDITION 1 is satisfied when both chromosomes are of high fitness and are close in the search space, whereas CONDITION 2 is when they are not close. CONDITION 3 is satisfied when both the chromosomes are of low fitness, and CONDITION 4 when only chromosome-i is of low fitness.

where, r_k is the range of the search space in the k^{th} dimension. We used Manhattan distance between the locations of two chromosomes, which is normalized by dividing the range in the corresponding dimension. Finally it is divided by d, so that the value of $\mid m_i(g) - m_j(g) \mid$ always lies between 0 to 1.

The condition part of the **while** statement in line 08 of the Algorithm RPC ensures that number of crossover is taken place, in spite of the fact that all crossovers tried are not successful. Two members from $\Pi''(g)$ are selected at random (line 09) for crossover. In line 10, the condition part of the **if** statement is ($f_i^{nor}(g)$ *is high*). The value of *high* is set at 0.7 during our experiments. Depending on whether we still want more explorations or we need more selective pressure during the end generations, we can set *high* to lower or higher values. We can even change its value from low to high as generation progresses.

If $f_i^{nor}(g)$ is high, and the fitnesses and locations of the two chromosomes are close (condition of the **if** statement in line 11), then there is a high probability (only after sufficient number of generations of search is over) that the crossover will take place (line 12). Else, if $f_i^{nor}(g)$ is low, and the fitnesses of the two selected chromosomes, m_i and m_j are close (condition of the **if** statement in line 13), the crossover will take place with high probability. If the fitnesses are not close, i.e., the fitness of m_j is high, the probability of crossover is low (to avoid disrupting a good chromosome m_j). This algorithm is described in Fig. 1.

2.2 How RPC is made effective only when $g \approx G$?

Let their normalized fitnesses of m_i and m_j are $f_i^{nor}(g)$ and $f_j^{nor}(g)$. Function $\varphi^1(f_i^{nor}(g) - f_j^{nor}(g))$, first part of the **if** condition in line 11, is of the following form,

$$\varphi^1(f_i^{nor}(g) - f_j^{nor}(g)) =$$

$$exp\left[-\frac{1}{2}\left(\frac{f_i^{nor}(g) - f_j^{nor}(g)}{\sigma(g)}\right)^2\right]$$

The shape of φ^1 is same as the normal function with maximum = 1 at $f_i^{nor}(g) = f_j^{nor}(g)$. The value of φ^1 decreases as the difference $(f_i^{nor}(g) - f_j^{nor}(g))$ increases. Finally, as written in line 11 of the Algorithm RPC, the crossover is possible only when $\varphi^1 > rand(0,1)$ (ANDing with the other part of the **if** condition). Here $rand(0,1)$ is a random real number between 0 and 1. The closer are $f_i^{nor}(g)$ and $f_j^{nor}(g)$, more probable is their crossover. $\varphi^1(f_i^{nor}(g) - f_j^{nor}(g))$ is shown graphically in Fig. 2, for $g = 1000$ and $g = 9000$. Here, G is set to 10,000. It is easy to see that RPC is effective only when $g \approx G$. How this change of φ^1 is implemented, is explained in section 2.3.

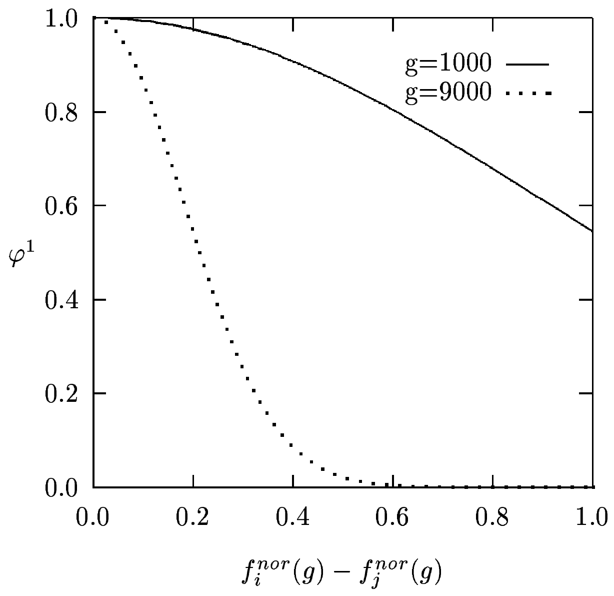

Figure 2: Function $\varphi^1(f_i^{nor}(g)f_j^{nor}(g))$, when $G = 10,000, \alpha = 2$

Exactly similar is the φ^2 function, which is the other part of the **if** condition in line 11,

$$\varphi^2(\mid m_i(g) - m_j(g) \mid) =$$

$$exp\left[-\frac{1}{2}\left(\frac{\mid m_i(g) - m_j(g) \mid}{\sigma(g)}\right)^2\right]$$

Thus, more closely the two chromosomes are in the search space, greater will be the value of φ^2 function.

2.3 Function $\sigma(g)$ that tunes φ^1 and φ^2 and thus RPC

The tuning of the effectiveness of RPC is done by introducing another function $\sigma(g)$, a function of g that controls the variance of φ. In the beginning, when g is low, $\sigma(g) \approx 1$, and thus allowing crossover for any randomly selected pair from $\Pi''(g)$. When g is large and nearing G (the maximum generation), $\sigma(g)$ becomes small. The way $\sigma(g)$ changes with generations is shown in Fig. 3. When $g \approx G$, φ^1 and φ^2 become sharper, as shown in Fig. 2, allowing only pairs with closer fitness and proximity to cross. Function $\sigma(g)$ is defined as follows:

$$\sigma(g) = 1 - \left[\frac{g}{\lambda G}\right]^{\alpha}$$

where α and λ are the controlling parameters. Fig. 3 is drawn with $\lambda = 1.1$ and $\alpha = 2, 1, 0.5$. In our experiments too we used these three different values of α.

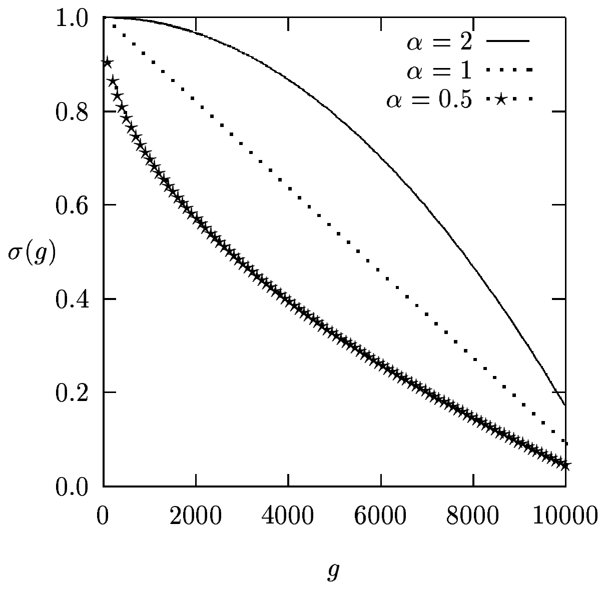

Figure 3: Function $\sigma(g)$ for different α

When $\alpha > 1$, $\sigma(g)$ changes slowly in the beginning (when g is low) and rapidly afterwards. Thus lots of exploration is done in the beginning, whereas rapid convergence is forced at the end. On the other hand, when $\alpha < 1$, the selection pressure is high almost from the beginning. For simple search problem, this would find the optimum result fast, but may miss the global maximum for complex multimodal functions. When $\alpha = 1$, $\sigma(g)$ decreases linearly. The efficiency and success of the algorithm depends somewhat on the complexity of the search space and corresponding proper choice of α. We will see that $\alpha = 1$, i.e. linear decrement of $\sigma(g)$ is a good choice for all situations. In general we will show by several experiments that RPC strategy works much better compared to SGA as well as SGA with "linear fitness scaling".

3 Experimental Set up and Results

The effectiveness of our algorithm is demonstrated by solving maximization problem for several univariate and multivariate multimodal functions. Due to space constraint, here we discuss results with only four functions, f_1 to f_4, as shown in Fig. 4 and listed in Table. 1. All are well known benchmark functions.

3.1 Simulation Parameters

As mentioned earlier, we implemented Standard Genetic Algorithm (SGA), SGA with liner fitness scaling, and our

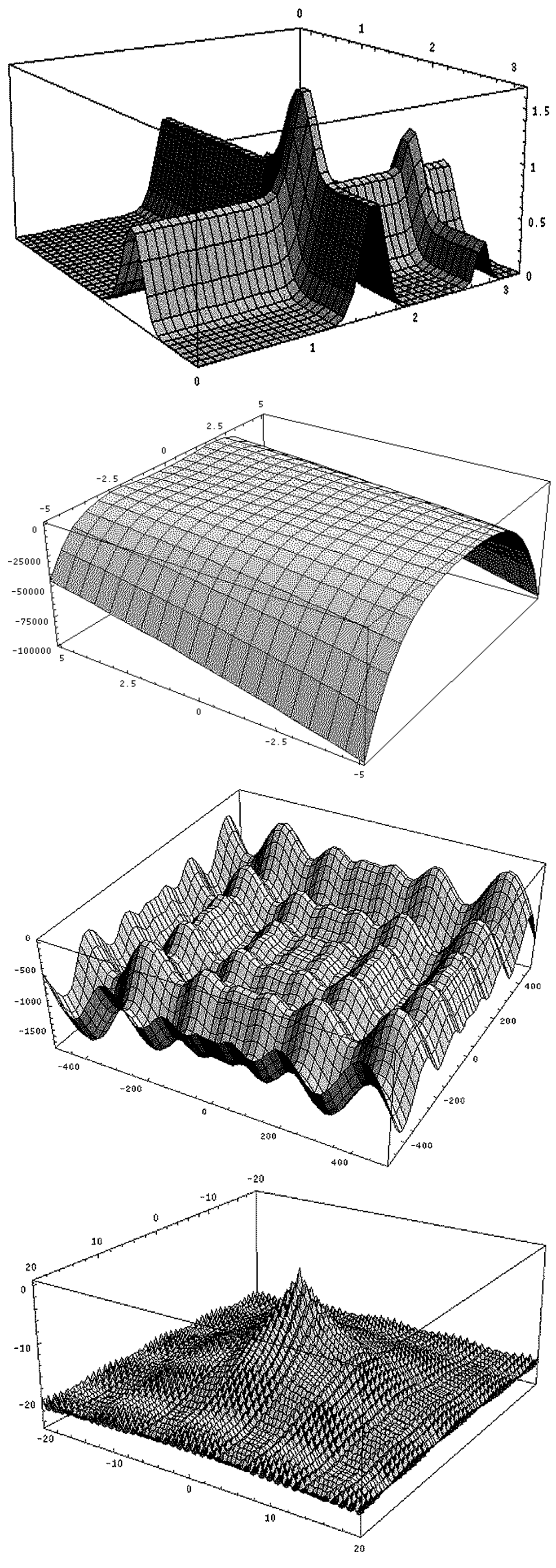

Figure 4: Graphs of functions f_1 and f_2 and experimental results

Epistatic Michalewicz's func:	$f_1 = \sum_{i=1}^{n} sin(x_i)\, sin^{20}(\frac{ix_i^2}{\pi})$	$n = 5, 0 \leq x_i \leq \pi, max = 3.698857$
Generalized Rosenbrock func:	$f_2 = -\sum_{i=1}^{n-1}((1-x_i)^2 + 100\,(x_{i+1} - {x_i}^2)^2)$	$n = 5, 5.12 \leq x_i \leq 5.12, max = 0$
Schwefel's function:	$f_3 = -\left(418.9829\,n + \sum_{i=1}^{n} -x_i\, sin\,(\sqrt{\lvert x_i \rvert})\right)$	$n = 5, 500 \leq x_i \leq 500, max = 0$
Generalized Ackley's func:		
	$f_4 = -\left(20 + e - 20\, e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} e^{\frac{1}{n}\sum_{i-1}^{n} cos(2\pi x_i)}\right)$	$n = 5, -20 \leq x_i \leq 20, max = 0$

Table 1: Functions used in the experiment

proposed GA with RPC. For RPC, we used three different functions for $\sigma(g)$ with $\alpha = 0.5$, $\alpha = 1.0$ and $\alpha = 2.0$. Therefore, in total we compare results of 5 different implementations. The following parameters are held constant for all runs.

Probability of crossover p_c = 0.6
Probability of mutation p_m = 0.03
Population size = 100

One point crossover method is performed. Because we used the maximum number of generations as the stopping criterion, we have performed our experiments with three different maximum generation numbers (G) as shown in Table. 2. Independent 50 runs with different random initial populations were done for each 3 cases of maximum number of generations. Initial population is same for all the five different algorithms, but are different in the 50 different runs. The following results in the next section are averages over 50 independent runs.

3.2 Analysis of results

Of the different analysis we did, we present here only two most significant results.

1. The average (over 50 runs) of best fitness value up to a certain number of generations (Fig. 5), and

2. The number of times a certain algorithm could hit its reachable maximum fitness and that value (which may be much less than actual maximum) at the end of G generations (Table. 2).

By (1) we could see how fast the convergence is achieved and by (2) we can judge the probability of reaching the target.

In Fig. 5, the best fitness values found until that generation is plotted against the number of generations, for functions f_1 to f_4 respectively (result for f_1 at the top, and f_4 at the bottom). The actual maximum value calculated analytically is mentioned in Table. 1.

From the results it is evident that the proposed RPC strategy could reach better quality of results faster, compared to SGA as well as SGA with fitness scaling. Results obtained using RPC, though almost similar for different α values, from various results (all are not shown here) we conclude that $\alpha = 1.0$ is a good choice for every occasion.

Finally it is also seen that using SGA or linear scaling of fitness, only very few of the run could reach the target maximum. With RPC the target maximum is reached more often as shown in Table. 2. In Table. 2, the average is the

		SGA	Scale	α=0.5	α=1	α=2
	Average⇒ *no. of gen*⇓	3.69714	3.69266	3.69885	3.69874	3.69885
f_1	100	0	0	6	6	5
	5000	1	4	7	11	3
	10000	5	13	12	8	14
f_2	Average⇒	-26.218	-3.734	-2.7588	-2.2554	-2.4564
	100	0	0	1	0	0
	5000	0	0	4	5	0
	10000	0	4	3	4	4
f_3	Average⇒	-45.724	-2.2102	-0.1749	-0.1999	-0.173
	100	0	0	0	0	0
	5000	0	0	2	0	0
	10000	0	0	1	2	3
f_4	Average⇒	-2.98	-0.439	-0.0945	-0.104	-0.202
	10000	0	0	3	0	0
	20000	0	2	3	3	3
	30000	0	0	2	3	4

Table 2: Average fitness and no. of times reaching optimum

average of best value of 50 runs with highest G (for f_1 to f_3 $G = 10000$, and for f_4 $G = 30000$). The other entries in the table shows the number of times the genetic search could reach global optimum out of 50 trials.

4 Conclusions

A new partner selection technique for crossover operation has been introduced for genetic algorithm searches. Crossover partners randomly selected from population are allowed to perform crossover only probabilistically and is controlled by a function. The probability function which controls this permission is very wide in the beginning, allowing any two randomly selected members to be crossed. But slowly, with progressing generations, this function becomes narrow, allowing members only with close fitnesses and proximities to be crossed. Simulation results with functions of different complexity show that the best fitness chromosomes are created with higher probabilities using RPC strategy.

The motivation of more exploration in the beginning of the search and high selective pressure at the end of the search was achieved in a number of previous researches [3] [4] [5] [6] by adaptive changes of crossover and mutation probabilities allowing more crossover and mutation in the beginning and less at the end. In all previous works, selection pressure is controlled in the selection stage, by manipulating fitnesses. We improve convergence by selecting who should crossed over with whom. It should be emphasized that our RPC strategy has no better exploration capability than standard GA. But it could be used in conjunction with

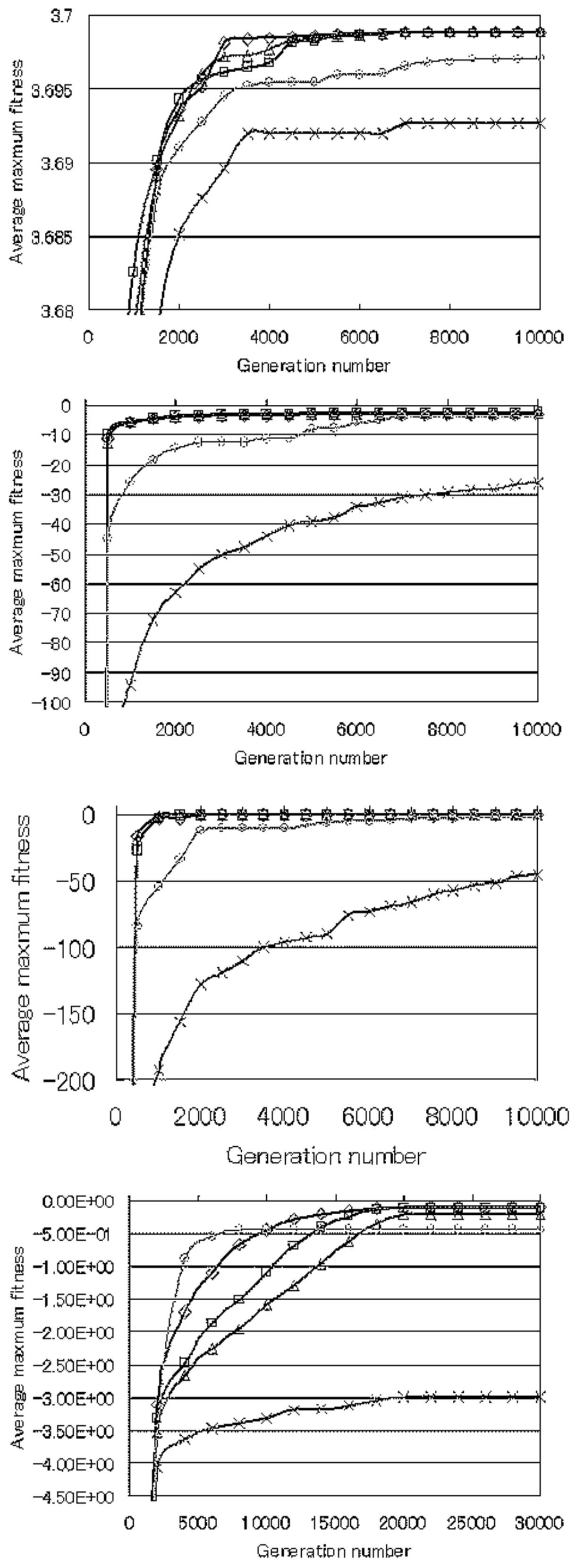

Figure 5: Average of best fitness values versus number of generations, results for functions f_1 to f_4 in order, f_1 at top. Here, symbols *circle* (O) for SGA with scaling, *cross* ($\times$) for SGA, *triangle* ($\triangle$) for $\alpha = 2.0$, *square* ($\square$) for $\alpha = 1.0$, and *diamond* ($\Diamond$) for $\alpha = 0.5$ are used to indicate results obtained using five different implementations.

other ideas where explorations are emphasized and premature convergence is avoided, and to see whether the results are further improved.

Depending on the nature of the search space, there would be an optimum choice for α. It is evident that for complex search problems, a higher value of $\alpha > 1.0$ is a better choice for allowing longer exploration in the beginning. On the other hand, for simple problems, $\alpha < 1$ will work faster. We are also working on finding a way to adaptively set the value of α with growing generations. The initial value of α would be set to 1, and from the analysis of the nature of changes of fitness values of the different members, it would take to its optimum value.

Acknowledgement: Part of this research is supported by Japan Government Somushou Project 042302005.

Bibliography

[1] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[2] Zbigniew Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, 1995.

[3] J. D. Schaffer and A. Morishima, "An adaptive crossover mechanism for genetic algorithms," in *Proceedings of Second International Conference on Genetic Algorithms*, pp. 36-40, 1987.

[4] T. C. Fogarty, "Varying the probability of mutation in genetic algorithms," in *Proceedings of Third International Conference on Genetic Algorithms*, pp. 104-109, 1987.

[5] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-16, No. 1, pp.122-128, Jan./Feb. 1986.

[6] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of crossover and mutation in Genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-24, No. 4, pp.656-666, April 1994.

[7] Shengxian Yang, "Adaptive Crossover in Genetic Algorithms Using Statistics Mechanism," *Artificial Life VIII, Standish, Abbass Bedau (eds)*, pp.182-185, MIT Press, 2002.

[8] Sankar K. Pal, Paul P. Wang, "Fitness Evaluation in Genetic Algorithms with Ancestors' Influence,", *Genetic algorithms for pattern recognition*, pp. 1-23, CRC Press, 1996.

[9] Goutam Chakraborty, and Kayo Hoshi, "Rank Based Crossover - A new technique to improve the speed and quality of convergence in GA", *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington, USA, pp.1595-1602, July 6-9, 1999.