Statistics-based Adaptive Non-Uniform Crossover for Genetic Algorithms

Shengxiang Yang
Department of Mathematics and Computer Science
University of Leicester
University Road, Leicester LE1 7RH
s.yang@mcs.le.ac.uk

Abstract

Through the population, genetic algorithm (GA) implicitly maintains the statistics about the search space. This implicit statistics can be used explicitly to enhance GA's performance. Inspired by this idea, a statistics-based adaptive non-uniform crossover, called SANUX, has been proposed. SANUX uses the statistics information of the alleles in each locus to adaptively calculate the swapping probability of that locus for crossover. A simple triangular function has been used to calculate the swapping probability. In this paper two different functions, the trapezoid and exponential functions, are investigated for SANUX instead of the triangular function. The experiment results show that both functions further improve the performance of SANUX across a typical set of GA's test problems.

1 Introduction

Based on natural evolution mechanisms researchers have developed several evolution-based stochastic optimum-seeking algorithms, summarily termed *Evolutionary Algorithms* (EAs). Of different classes of evolutionary algorithms, *Genetic Algorithms* (GAs) are the best known and have been widely studied [9]. GA emulates the natural evolution process and maintains a population of potential solutions to a given problem, which are evaluated according to a problem-specific fitness function that defines the evolution environment for the population. A new population is created by randomly selecting relatively fit individuals of the present population and evolving them through crossover and mutation operations. Through the population GA implicitly maintains the statistics about the search space. That is, useful materials permeate in the population. GA uses the selection, crossover and mutation operators to explicitly extract the statistics from the population to reach the next set of points in the search space.

In fact, this implicit statistics in the population can be used explicitly to enhance GA's performance. Inspired by this idea, a statistics-based adaptive non-uniform crossover, called SANUX, has been proposed by Yang [18]. SANUX explicitly uses the statistics information of the alleles in each locus to adaptively calculate the swapping probability of that locus for crossover operation. A simple triangular function has been used to calculate the swapping probability from the statistics of allele distribution. In this paper, two different functions, the trapezoid and exponential functions, are proposed for SANUX instead of the simple triangular function. These two functions make use of the statistics information more efficiently with respect to construction and protection of useful building blocks with the progress of the GA.

2 Related Work on Crossover

2.1 Traditional Crossover Operators

Traditionally, GAs have used the one-point crossover and the two-point crossover. This decision was supported theoretically and empirically by early work of GA's researchers [3, 10]. However, researchers have also carried out experiments with multi-point crossover: the n-point crossover [6] and the uniform crossover [16]. With the n-point crossover, n cut points are randomly chosen within the strings and the $n-1$ segments between the n points of the parents are exchanged. Uniform crossover is the generalization of n-point crossover. It creates offsprings by deciding, for each bit of the parent, whether to swap the allele of that bit with the other parent. The decision is made using an unbiased coin flip, i.e., the probability of swapping p_s is 0.5. Spears and De Jong [14] proposed the parameterized uniform crossover which dif-

fers from the traditional uniform crossover in that the decision for each locus is made by a biased coin flipping, i.e., the swapping probability p_s could be other than 0.5.

For traditional crossover operators, there exist two recombination biases: *positional bias* and *distributional bias* [1, 6]. Positional bias exists when the creation of a new schema is dependent upon the location of its defined bits. Distributional bias exists if the expected amount of material being exchanged is concentrated around a mean value. The two biases exist because traditional crossover operators are inherently all based on uniformly randomization mechanism, i.e., generating cut points (n-point crossover) or swapping points (uniform crossover) uniformly randomly over the loci of the chromosome. This situation is not very true with natural evolution. In nature, "hot spots" exist in the chromosome for crossover.

2.2 Adapting Crossover Operators

Natural evolution is intrinsically dynamic and adaptive. To emulate this, recently researchers have applied adaptation techniques to enhance GA's capabilities [5]. Based on the mechanism of change, adaptation in GAs can be classified into three categories: *deterministic adaptation* where the value of a strategy parameter is altered according to some deterministic rule, *adaptive adaptation* where feedback information from the search process is used to adjust a strategy parameter, and *self-adaptive adaptation* where the strategy parameter is encoded into and coevolves with the chromosomes.

According to Yang [18], adaptation in crossover happens in three levels from top to bottom. In the top level, crossover operators are themself adapted during a run of the GA. Davis [2] proposed that the GA selects operators from a set of operators, each with a fixed probability. Spears [13] appended to each individual one *tag bit* that co-evolves with the individual and is used to switch between two-point and uniform crossover. In the medium level, the rate or probability of crossover is altered during a run of the GA. Julstrom [11] proposed an adaptive mechanism that regulates the ratio between crossover and mutation based on their performance. Tuson and Ross [17] encoded into each individual the crossover and mutation probabilities as one-normalized real numbers that are used by and co-evolve with the individual. In the bottom level, the position of crossing or swapping probability in each locus is adapted during a run of the GA. Fraser [8] associated for each locus of an individual a swapping probability that co-evolves with the individual. Levenick [12] inserted a metabit before each bit of the individual. If the metabit was "1" in both parents swapping occurred with base probability P_b, otherwise with reduced probability P_r.

3 Statistics-based Adaptive Non-Uniform Crossover

3.1 Motivation of SANUX

Holland's building block hypothesis states that GA works by juxtaposition of short, low-order schemas into fitter schemas [9, 10]. However, it doesn't state how crossover, the major source of the search power of GA, works to recombine highly fit schemas from short, low-order schemas. Through theoretical analysis and experimental study researchers found that crossover operators are disruptive as well as constructive with respect to schemas [15]. Hence, it may be beneficial to take a single gene instead of schema as the fundamental functionality unit and study the behavior of a single gene. Anyway, schemas consist of genes.

For the convenience of description and analysis, Yang [18] has introduced the concepts of intrinsic attribute and extrinsic tendency of allele valuing for a gene locus. In the optimal binary-encoded solution(s) of a given problem, for a gene locus if its allele is 1 it is called *1-intrinsic*, if its allele is 0 it is called *0-intrinsic*, otherwise if its allele can be either 0 or 1 it is called *neutral*. Whether a locus is 1-intrinsic, 0-intrinsic, or neutral depends on the problem being solved and the encoding scheme, e.g., whether introns are inserted [12]. During the running of a GA, for a gene locus, if the frequency of 1's in its alleles over the population tends to increase (to the limit of 1.0) with time (generation), it is called *1-inclined*; if the frequency of 1's tends to decrease (to the limit of 0.0), it is called *0-inclined*; otherwise, if there is no tendency of increasing or decreasing, it is called *non-inclined*. Whether a locus is 1-inclined, 0-inclined, or non-inclined depends on the problem being solved, encoding scheme, genetic operators and initial conditions.

Usually and hopefully as the GA progresses, those gene loci that are 1-intrinsic (or 0-intrinsic) will appear to be 1-inclined (or 0-inclined), i.e., the frequency of 1's in the alleles of these loci will eventually converge to 1 (or 0). This convergent information is implicit in

the population. Traditional crossover operators make no use of this implicit convergent information and consistently generate cut points or swapping points randomly but uniformly across the chromosome. Obviously it will be beneficial to link these partially or fully converged genes into groups or building blocks and co-evolve them during crossover operations. *Gene linkage* is the property of grouping interactive genes to evolve them together. Of course, we need some mechanism that makes use of this implicit convergent information to link genes. In fact, interaction between genes could be addressed by varying their swapping probabilities and forming linkage groups based on their swapping probabilities. Those genes with small-valued probabilities (i.e., close to zero) form a linked group because it is unlikely for crossover to disrupt that group. This is realized in SANUX. SANUX explicitly uses the implicit convergence information as feedback information to guide the crossover by adaptively adjusting the swapping probability for each locus based on the statistics of alleles in that locus.

3.2 Description of SANUX

We use the frequency of 1's in the alleles in a locus over the population (equivalently we can also use the frequency of 0's as the argument) to calculate corresponding swapping probability of that locus. The frequency of 1's in the alleles of a locus can be looked as the degree of convergence to "1" for that locus. Let L be the length of binary strings, $f_1(i,t)$ $(i = 1,\ldots,L)$ denote the frequency of 1's in the alleles in locus i over the population at time (generation) t and $p_s(i,t)$ $(i = 1,\ldots,L)$ denote the swapping probability of locus i at time t. In [18], a simple triangular function, as shown in Figure 1, is used to calculate $p_s(i,t)$ from $f_1(i,t)$, defined as:

$$p_s(i,t) = P_{max} - 2*|f_1(i,t) - 0.5|*(P_{max} - P_{min}) \quad (1)$$

where $|.|$ is an absolute function, P_{max} and P_{min} are the maximum and minimum allowable swapping probabilities for a locus respectively. The triangular function is symmetric with respect to the line $f_1(i,t) = 0.5$ due to the symmetric property of allele valuing in a locus.

Now during the evolution of the GA, after a new population has been generated, we first calculate the distribution of 1's $f_1(i,t)$ for each locus i over the population t and from this obtain the swapping probability $p_s(i,t)$ of locus i. Then we can perform SANUX operations

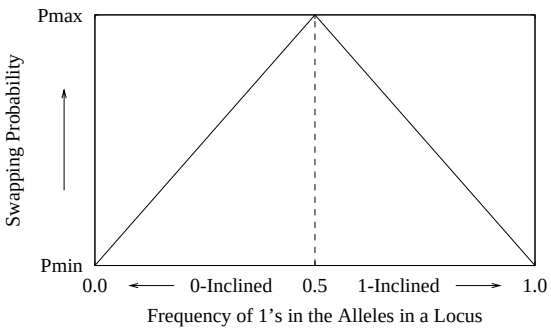

Figure 1: Triangular function for calculating the swapping probability of a locus.

similarly as traditional parameterized uniform crossover. Figure 2 shows an example operation of applying SANUX. When performing SANUX on two parents individuals P_1 and P_2, we first generate a mask bit by bit by flipping a coin biasedly, i.e., generating a "1" with the swapping probability $p_s(i,t)$ for each locus i. The generated mask is then used to guide the crossover by exchanging those bits of P_1 and P_2 that correspond to the positions where there are a "1" in the mask while leaving other loci unchanged.

1's Freq. in loci:	0.9	0.4	0.2	0.9	0.6
Calculating:	⇓	⇓	⇓	⇓	⇓
Swapping Prob.:	0.1	0.4	0.2	0.1	0.4
Biased Flipping:	⇓	⇓	⇓	⇓	⇓
Created Mask:	0	1	0	0	1
Applying Mask:		⇓			⇓
Parent P_1:	1	1	0	1	0
Parent P_2:	1	0	0	1	1
Swapping:		⇓			⇓
Child C_1:	1	0	0	1	1
Child C_2:	1	1	0	1	0

Figure 2: An example operation of SANUX with the triangular calculating function where $P_{max} = 0.5$ and $P_{min} = 0$.

From above descriptions, it can be seen that SANUX is much simpler than those adaptation mechanisms that add extra tag bit [12] or value [8] per genetic bit and co-evolve these tag bits or values with each individual. With SANUX, what we add to traditional uniform crossover are spatially only one real vector of the chromosome length dimension that records the swapping probability for each locus, and computationally only one statistics per genera-

tion that calculates the frequency of ones (hence the swapping probability) for each locus. This simple extra statistics is well rewarded in the sense of computational efficiency. First, for each crossover operation at generation t, the number of swappings on strings of length L is on average $L/2$ with uniform crossover and $\sum_{i=1}^{i=L} p_s(i,t)$ with SANUX. This number is the same for both uniform crossover and SANUX when the population is randomly initialized, assuming $P_{max} = 0.5$. However, with the progress of the GA, 1- and 0-intrinsic genes tend to converge to 1 and 0 respectively and their swapping probabilities decrease according to Equation (1). This results in reduced number of swappings with SANUX, i.e., $\sum_{i=1}^{i=L} p_s(i,t) < L/2$. Second, more importantly, SANUX can reduce invalid crossover operations and hence fitness evaluations. As the population converges, with uniform crossover, many crossover operations are wasted on those converged loci since generated offsprings with found building blocks destroyed during crossover are easily kicked off by selection operators because they are usually less fit. Most of these invalid crossover operations can be saved by SANUX through decreasing swapping probabilities for those converged loci.

Another more important point of SANUX is its property of implicit gene linkage. For example, in Figure 2 loci 1 and 4 are more convergent and implicitly linked though they are not adjacent because the probability for them to coevolve via crossover is $0.1 * 0.1 + 0.9 * 0.9 = 0.82$. SANUX differs from Fraser's crossover in that SANUX adapts one probability vector for all individuals based on one simple statistics per generation while Fraser's crossover modifies one probability vector for each individual per crossover based on a random learning rule [8].

3.3 New Calculating Functions

From above discussions, we can see that there is a close-loop control in SANUX in the sense of constructing and preserving building blocks. The GA is first used to construct useful building blocks. Then after some time when some useful building blocks are built up the statistics information is used to protect these found useful building blocks from being destroyed by the crossover and hence divert the crossover to search unconverged loci for potential more unknown useful building blocks because unconverged loci have relatively higher swapping probabilities. With this understanding, we can further enhance this close-loop control by introducing new functions instead of the above triangular function to calculate the swapping probability of each locus from its allele distribution statistics information. As mentioned previously, in this paper we propose two such functions.

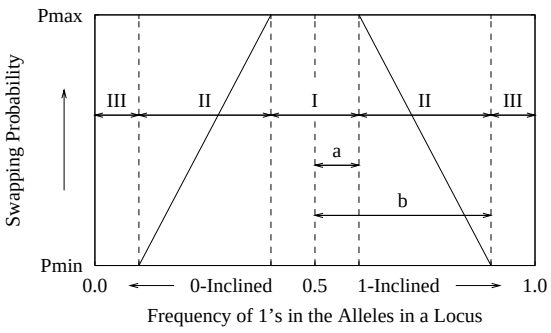

Figure 3: Trapezoid function for calculating the swapping probability of a locus.

The first function is a trapezoid function, as shown in Figure 3, defined by the following linear segmented equation.

$$p_s(i,t) = \begin{cases} P_{max}, & \text{if } |f_1(i,t) - 0.5| \le a \\ \frac{b - |f_1(i,t) - 0.5|}{b-a} * (P_{max} - P_{min}) + & \\ \quad P_{min}, & \text{if } a < |f_1(i,t) - 0.5| < b \\ P_{min}, & \text{if } |f_1(i,t) - 0.5| \ge b \end{cases} \tag{2}$$

where a and b are parameters that satisfy the relation of $0 \le a \le b \le 0.5$. The trapezoid function is symmetric with respect to the line $f_1(i,t) = 0.5$. From Figure 3 it can be seen that the triangular function in Figure 1 is an extreme example of the trapezoid function when $a = 0$ and $b = 0.5$ in Equation (2).

As shown in Figure 3, the parameters a and b split the whole range of $f_1(i,t)$ into three zones: I) $|f_1(i,t) - 0.5| \le a$; II) $a < |f_1(i,t) - 0.5| < b$; III) $|f_1(i,t) - 0.5| \ge b$. Zone I is called *constructive zone* where useful building blocks are mainly constructed by SANUX. Zone II is called *transient zone* where building blocks are constructed as well as destroyed. And Zone III is called *protective zone* where building blocks found so far are protected from being destroyed by SANUX while exploring unknown useful building blocks is still carried out by SANUX on unconverged loci. From Figure 3 it can be seen that the larger the value of a, the wider the region that allows the GA to construct building blocks, and that the smaller the value of b, the stronger the protection to converged genes or found building blocks from being destroyed by SANUX.

The second function we introduce here is an

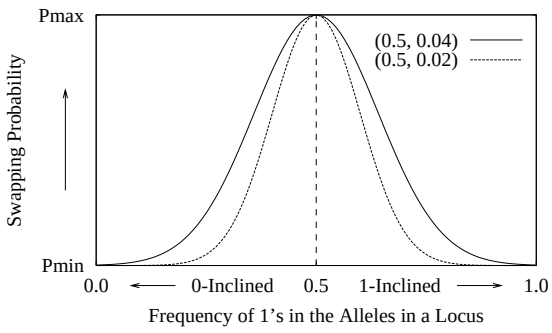

Figure 4: Exponential function for calculating the swapping probability of a locus.

exponential function, as shown in Figure 4, defined as follows:

$$p_s(i,t) = \alpha * exp(-(f_1(i,t) - 0.5)^2/\beta) \qquad (3)$$

where α is the parameter with the same functionality as P_{max} in Equation (1) and Equation (2), and β is the parameter that combines the functionalities of parameters P_{min}, a and b in Equation (2) and controls the falling speed of $p_s(i,t)$ when $f_1(i,t)$ diverts away from 0.5. The smaller the value of β, the faster the falling speed of $p_s(i,t)$, and the stronger the protection to converged genes or building blocks found so far. As shown in Figure 4, the solid curve shows the function with $(\alpha, \beta) = (0.5, 0.04)$ while the dashed curve is when $(\alpha, \beta) = (0.5, 0.02)$.

From above discussions, we can see that both the trapezoid and exponential functions loose the region for the GA to construct building blocks and enhance the protection to found building blocks at the same time. Both sides are expected to be beneficial for GA's performance.

4 The Test Problems

4.1 The Max Ones Problem

The Max Ones problem is simply to maximize ones in a binary string. The fitness of a string is the number of ones it contains. A string length of 100 bits will be used for this study.

4.2 Royal Road Functions

The Royal Road functions R_1 and R_2 contain tailor-made building blocks and are devised to investigate GA's performance with respect to schema processing and recombination by Forrest and Mitchell [7]. They are defined using a list of schemas. Each schema is given a coefficient which is equal to its defining order. The fitness of a bit string x for both R_1 and R_2 is computed by summing the coefficients corresponding to each of the given schema of which x is an instance. The optimal solutions for R_1 and R_2 have a fitness of 64 and 192 respectively.

4.3 The L-SAT Problem Generator

The random L-SAT problem generator [4] is a boolean satisfiability problem generator devised to investigate the effects of epistasis on the performance of GAs. It generates random boolean expressions in conjunctive normal form of clauses subject to three parameters V (number of boolean variables), C (number of disjunctive or conjunctive clauses) and L (the length of the clauses). Each clause is created by selecting L of V variables uniformly randomly and negating each variable with probability 0.5. For each generated boolean expression, the aim is to find an assignment of truth values to the V variables that makes the entire expression true. Since the boolean expression is randomly generated, there is no guarantee that such an assignment exists. The fitness function for the L-SAT problem is as follows:

$$f(chrom) = \frac{1}{C}\sum_{i=1}^{C} f(clause_i)$$

Where *chrom* consists of C clauses and the fitness contribution of clause i, $f(clause_i)$, is 1 if the clause is satisfied or 0 otherwise.

As in [4], in our experiments we will fix V to 100 and L to 3 and vary the value of C from 200 (low epistasis) to 1200 (medium epistasis) to 2400 (high epistasis).

5 Experimental Study

5.1 Design of Basic Experiment

In our basic experiment study we compare the SANUX with the trapezoid and exponential functions over SANUX with triangular function and traditional 2-point, 0.5 uniform and 0.2 uniform crossover on the chosen test problems. The parameters for different calculating functions of SANUX are set as follows: $P_{max} = 0.5$ and $P_{min} = 0.0$ for triangular and trapezoid functions, $(a, b) = (0.1, 0.4)$ for trapezoid function, and $(\alpha, \beta) = (0.5, 0.04)$ for exponential function.

For each experiment of combining different crossover operators and test problems, 100 independent runs were executed under the same

100 different random seeds. In all the experiments, the GA uses the fitness proportionate selection with stochastic universal sampling and elitist model, and bit flip mutation. And typically the probabilities of crossover and mutation were fixed to 0.6 and 0.001 respectively and the population size was set to 100 for each run. For each run, we recorded the best-so-far fitness every 100 evaluations. Here, only those chromosomes changed by crossover and mutation operations are evaluated and counted into the number of evaluations. Each experiment result is averaged over the 100 independent runs.

5.2 Basic Experiment Results

The results of the basic experiment on different test problems are shown in Figure 5 to Figure 7 respectively. From these figures it can be seen that SANUX with different calculating functions has outperformed traditional crossover operators on all test problems except the L-SAT problems with low epistasis. On most test problems lowering the swapping probability p from 0.5 to 0.2 improves the performance of uniform crossover (0.2 uniform crossover outperforms 2-point crossover and 0.5 uniform crossover), however 0.2 uniform crossover is beaten by SANUX. The relative performance of different crossover seems quite consistent on all of the test problems. The approximate performance order from best to worst is from SANUX-Exponent $\Rightarrow$ SANUX-Trapezoid $\Rightarrow$ SANUX-Triangle $\Rightarrow$ 0.2 uniform crossover $\Rightarrow$ 0.5 uniform crossover $\Rightarrow$ 2-point crossover. Within SANUX both the trapezoid function and exponential function have outperformed the triangular function. This confirmed our prediction.

From Figure 5 to Figure 7 we can also see that the performance of different crossover operators differs most heavily on royal road functions, especially on R_2. This happens because, as mentioned before, royal road functions contain tailor-made building blocks and thus are good test problems to investigate GA's performance with respect to schema processing and recombination. In other words, they provide the chance to test GA's performance with respect to crossover operators instead of other genetic operators. Hence, crossover operators that win on royal road functions much more strongly prove themselves to be more powerful than other crossover operators than on other problems. SANUX-Exponent is obvious such a winner. This means as a crossover operator SANUX-Exponent is efficient.

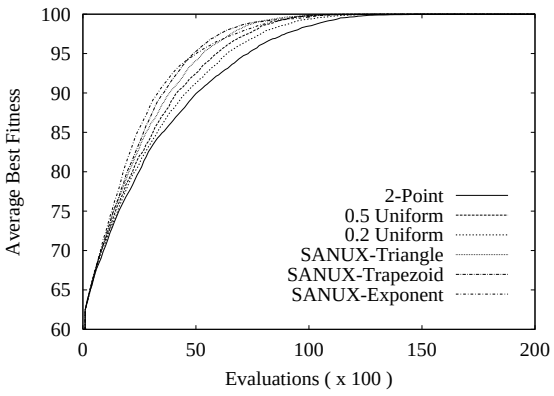

Figure 5: Comparison of GAs with different crossover on Max Ones problem.

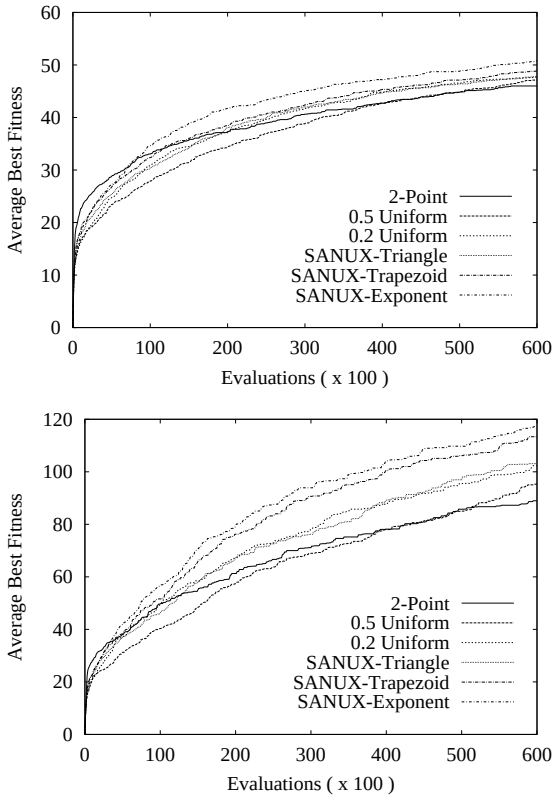

Figure 6: Comparison of GAs with different crossover on (Top) R_1 and (Bottom) R_2.

5.3 Experiment on Effect of Parameters

As mentioned previously in this paper, the trapezoid and exponential functions have some parameters that may affect the performance of SANUX. Hence, suitably adjusting these parameters may further improve the performance of GAs with SANUX. To test this thought, we fur-

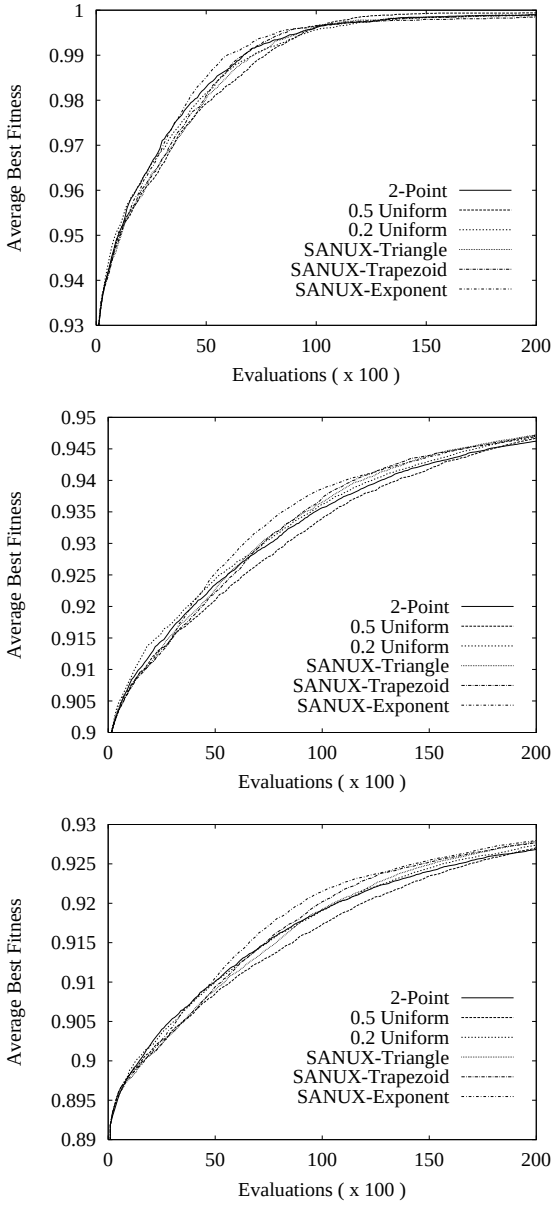

Figure 7: Comparison of GAs with different crossover on L-SAT problems with (Top) low, (Middle) medium, and (Bottom) high epistasis.

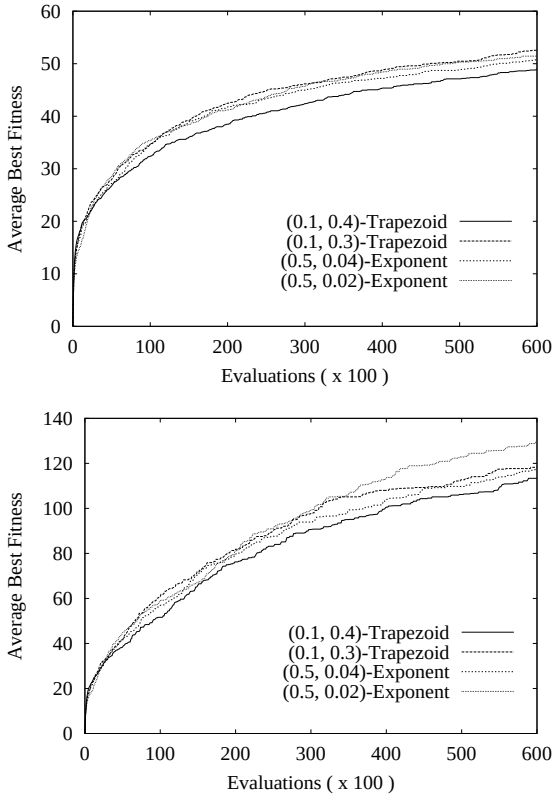

Figure 8: Comparison of GAs with SANUX with different parameters in calculating functions on (Top) R_1 and (Bottom) R_2.

ther carried out experiments with different parameter settings for the two calculating functions of SANUX on R_1 and R_2 because our basic experiments show that the performance disparity of using different crossover happens most heavily on them. We decreased the value of parameter b from 0.4 to 0.3 for the trapezoid function and the value of parameter β from 0.04 to 0.02 for the exponential function (see Figure 4 for the two exponential curves). The other parameters and conditions are kept the same as above basic experiments. Both changes are aimed to enhance the protection to building blocks found so far from being destroyed by crossover. Hence the performance might be improved if the changes are appreciate.

The experiment results are shown in Figure 8 where the labels inside the figure show relevant parameters in the form of (a, b)-Trapezoid and (α, β)-Exponent. From Figure 8, it can be seen that, as expected, for both changes GA's performance is really improved.

6 Conclusions and Future Work

In this paper, we investigate two different functions, the trapezoid and exponential functions, which are used with SANUX instead of the triangular function to calculate the swapping probability for a locus from its allele distribution statistics. The motivation of introducing these two functions is to make more efficient use of the statistics information implicit in the population to explicitly guide the crossover operation. Both these two functions widen the region for the GA to construct building blocks and enhance

the protection to searched building blocks at the same time, hence are expected to be beneficial for the performance of GAs.

The experiment results of this study show that SANUX with the trapezoid and exponential functions performs better than SANUX with the triangular function and traditional two-point and uniform crossover (parameterized or not) on a set of typical GA's test problems. Our experiment results indicate that the exponential function is a good choice to be used in SANUX for its performance and simplicity, and that SANUX with (0.5, 0.02)-exponential function is a good candidate as a crossover operator.

Since SANUX works at the bottom-level of crossover, it can be easily combined into other adaptation techniques for crossover and can act as the basis for designing and analyzing new algorithms, which is one future work on SANUX. Comparing obtained SANUX with other adaptation techniques for crossover is a second future work on SANUX. Formally analyzing SANUX with different calculating functions is a third important future work on SANUX.

References

[1] L. B. Booker. Recombination distributions for genetic algorithms. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 29–44. Morgan Kaufmann, 1992.

[2] L. Davis. Adapting operator probabilities in genetic algorithms. In D. Schaffer, editor, *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, pages 60–69. Morgan Kaufmann, 1989.

[3] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD Thesis, University of Michigan, Ann Abor, MI, 1975.

[4] K. A. De Jong, M. A. Potter, and W. M. Spears. Using problem generators to explore the effects of epistasis. In T. Bäck, editor, *Proc. of the 7th Int. Conf. on Genetic Algorithms*, pages 338–345. Morgan Kaufmann, 1997.

[5] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 3:124–141, 1999.

[6] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer. Biases in the crossover landscape. In J. D. Schaffer, editor, *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, pages 10–19. Morgan Kaufmann, 1989.

[7] S. Forrest and M. Mitchell. Biases in the crossover landscape. In D. SWhitley, editor, *Foundations of Genetic Algorithms 2*, pages 10–19. Morgan Kaufmann, 1993.

[8] A. S. Fraser. Simulation of genetic systems by automatic digital computers. II. effects of linkage or rates of advance under selection. *Australian Journal of Biological Sciences*, 10:492–499, 1957.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

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

[11] B. Julstrom. What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In L. J. Eshelman, editor, *Proc. of the 6th Int. Conf. on Genetic Algorithms*, pages 81–87. Morgan Kaufmann, 1995.

[12] J. Levenick. Metabits: genetic endogenous crossover control. In L. J. Eshelman, editor, *Proc. of the 6th Int. Conf. on Genetic Algorithms*, pages 88–95. Morgan Kaufmann, 1995.

[13] W. M. Spears. Adapting crossover in evolutionary algorithms. In *Proc. of the 4th Annual Evolutionary Programming Conference*, pages 367–384, 1995.

[14] W. M. Spears and K. A. De Jong. On the virtues of parameterized uniform crossover. In R. K. Belew and L. B. Booker, editors, *Proc. 4th Int. Conf. on Genetic Algorithms*, pages 230–236. Morgan Kaufmann, 1991.

[15] W. M. Spears and K. A. De Jong. Dining with gas: operator lunch theorems. In *Foundations of Genetic Algorithms 5*, pages 85–101. Morgan Kaufmann, 1998.

[16] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1989.

[17] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6: 161–184, 1998.

[18] S. Yang. Adaptive non-uniform crossover based on statistics for genetic algorithms. In W. B. Langdon, *et al*, editors, *Proc. of the Genetic and Evolutionary Computation Conference, GECCO 2002*, pages 650–657. Morgan Kaufmann, 2002.