Adaptive Non-Uniform Crossover Based on Statistics for Genetic Algorithms

Shengxiang Yang

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

Abstract

The genetic algorithm (GA) is a metaheuristic search algorithm based on mechanisms abstracted from population genetics. Through the population, the GA implicitly maintains the statistics about the search space. This implicit statistics can be used explicitly to enhance GA's performance. In this paper, a new statistics-based adaptive non-uniform crossover (SANUX) is proposed. SANUX uses the statistics information of the allele distribution in each locus to adaptively adjust the crossover operation. Our preliminary experiment results show that SANUX is more efficient than traditional one-point, two-point, and uniform crossover across a representative set of search problems.

1 INSTRUCTIONS

Based on Holland's simple genetic algorithm (Holland, 1975), there have been many variations developed both in GA's macro-structure and micro-structure (Goldberg, 1989). The GA, as one kind of generation-based evolutionary algorithm, maintains a population of candidate solutions to a given problem, which are evaluated according to a problem-specific fitness function that defines the environment for the evolution. New population is created by selecting relatively fit members of the present population and evolving them through recombination and mutation operations. The performance of a GA is dependent on many factors, such as encoding scheme, selection method, population size, crossover and mutation operators. This makes it difficult, if not impossible, to choose operators for optimal performance. In this paper we focus on the recombination operator.

In most GAs, recombination operators act on a pair of individuals (parents) to produce new offsprings (children) by exchanging segments of the parents' corresponding genetic material. This kind of two-parent recombination is usually called "crossover". The number of crossover points determines how many segments are exchanged. Traditionally, GAs have used the one-point crossover and two-point crossover that choose crossing points uniformly at random. This decision was supported theoretically and empirically by early work of GA's researchers (De Jong, 1975; Holland, 1975). However, researchers have also carried out experiments with multi-point crossover: the n-point crossover (Eshelman, 1989) which exchanges $n-1$ segments of the two parents and the uniform crossover (Reed, Toombs and Barricelli, 1967; Syswerda, 1989) which generates offsprings by swapping each bit of the parents with a fixed probability $p = 0.5$. Spears and De Jong (1991) proposed that the swapping probability under uniform crossover could be fixed to values other than 0.5. Recently, researchers have applied adaptation techniques to recombination to enhance GA's capabilities and have made the adaptation of recombination operators and/or their parameters one of the most promising research areas in GAs.

As a search algorithm based on mechanisms abstracted from population genetics, the GA implicitly maintains the statistics about the search space via the population. It uses the selection, crossover and mutation to explicitly extract the implicit statistics from the population to reach the next set of points in the search space. This implicit statistics can be used explicitly to enhance GA's performance. In this paper, a new statistics-based adaptive non-uniform crossover operator, called SANUX, is proposed. SANUX explicitly uses the statistics information of the distribution of alleles in each gene locus over the population to adaptively guide the process of crossover. With SANUX, the probability of swapping alleles of the parents for each locus is derived from the distribution of alleles in that locus over the population and thus is adaptively adjusted with the progress of the GA. In the rest of this paper, we first briefly review previous relevant work, then describe SANUX in detail, finally present our experiment study that compares SANUX over traditional one-point, two-point and uniform crossover based on a representative set of test problems.

2 REVIEW OF RELATED WORK

2.1 ADAPTATION IN THE GA

The performance of a GA significantly depends on the operators and relevant parameters used. However, choosing the right GA operators and appropriate parameters is a difficult task. Traditionally, they are determined by experience or primary experiments from a particular domain in advance and then are fixed during the running of the GA. This kind of constant parameter setting approach is time-consuming, can lead to sub-optimal performance when parameters are inappropriately set. And what is worse lies in the nature that the optimal parameter values may vary with the evolution process of GAs. Hence, researchers have applied many adaptation techniques into GAs to enhance their performance (Eiben, Hinterding and Michalewicz, 1999). 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 there is some form of feedback information from the search process that is used to direct the change of a strategy parameter, and *self-adaptive adaptation* where the parameter to be adapted is encoded into the chromosomes and undergoes genetic operations (hence, also called *co-evolution*).

2.2 ADAPTATION IN CROSSOVER

Since crossover is one of the primary genetic operations in GAs, adaptation in crossover operators has long been studied and there have been many results (Spears, 1997). Generally speaking, adaptation in crossover happens in three levels from top to bottom.

2.2.1 Adapting the Type of Crossover

In this top level, crossover operators are themself adapted during a run of the GA. Davis (1989) proposed that the GA needn't apply both crossover and then mutation to the selected parent, instead it can select operators from a set of operators, each with a fixed probability. Eshelman and Schaffer (1994) proposed an adaptive mechanism that uses restarts (when converged, the population is partially or fully randomized except the best individual) and between restarts switches between two crossover operators based on their performance. Spears (1995) appended to each individual one *tag bit* that co-evolves with the individual and is used to switch between two-point and uniform crossover. If both the tag bits of two parents are 1, choose two-point crossover; if both are 0, choose uniform crossover; otherwise, choose either randomly.

2.2.2 Adapting the Rate of Crossover

In this medium level, the rate or probability of crossover is altered during a run of the GA. Julstrom (1995) proposed an adaptive mechanism that regulates the ratio between crossover and mutation based on their performance. Corne, Ross and Fang (1994) devised the COst Based operator Rate Adaptation (COBRA) method for adapting operator probabilities in timetabling problems. With COBRA the GA periodically swaps given k fixed probabilities between k operators by giving the highest probability to the operator that has been producing the most gains in fitness. Tuson and Ross (1998) extended the COBRA method by encoding into each individual the crossover and mutation probabilities as real numbers (normalized to one) that are used by and co-evolve with the individual.

2.2.3 Adapting the Crossing Position or Swapping Probability in Each Locus

In this bottom level, the position of crossing or swapping probability in each locus is adapted during a run of the GA. Rosenberg (1967) attached to each locus i an integer $x_i \in \{1, \ldots, 7\}$ and calculated the crossing probability p_i of locus i from the probability distribution defined by $p_i = x_i / \sum x_i$. Schaffer and Morishima (1987) proposed an self-adaptive scheme that appends at the end of each individual a crossover bitmap that specifies allowable crossing positions and co-evolves with the individual. Booker (1992) introduced the notion of recombination distributions which describe the probability of all possible recombination events. Different probability distributions describe different operators. White and Oppacher (1994) developed an adaptive uniform crossover where each bit string in the population is augmented at each bit position with an automation whose state maps to a crossover probability for that bit location. Levenick (1995) 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. Vekaria and Clark (1999) proposed the "selective crossover" which biases alleles that are known to have increased an individual's fitness. It attaches a real vector to an individual to accumulate fitness information in previous generations and uses this information to preserve known fit alleles.

A unique topic in this level is the study of *gene linkage*, the property of grouping interactive genes to evolve them together. Fraser (1957a, 1957b) offered one of the earliest computer simulations of genetic systems where he suggested a crossover that associates a variable swapping probability for each locus of a string. Interaction between genes could be addressed by 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. Goldberg, Korb and Deb (1989) devised the "messy GA" as an attempt to explicitly link genes using variable length strings. An alternative scheme was presented by Harik and Goldberg (1996) that attempted to co-evolve the positions and values of genes using a representation which consider loci as points on a circle with the real-valued distance between points denoting their linkage. Smith and Fogarty (1996) developed a Linkage Evolving Genetic Operator (LEGO) that attaches to each gene two flags, denoting whether it is linked to its neighbours on the left and right respectively. A pair of genes had to be bi-directionally linked by setting the right flag on one and the left flag on the other before they were considered a block. LEGO allows these blocks to come together from multi-parents to form new individuals.

2.3 PROBABILITY-BASED OPTIMIZATION ALGORITHMS

Recently, based on the convergence property of the GA, a number of GA-like algorithms have been developed that replace the GA's population and crossover operator with a probabilistic representation and generation method. The Population-Based Incremental Learning (PBIL) by Baluja (1994) evolves a probability vector, the values of which are initialized to 0.5. A number of solutions are generated based on the probabilities in the vector. The probability vector is then pushed towards the generated solution with the highest evaluation. A new set of solutions are generated according to the updated vector, and the cycle continues till terminated. As search progresses, the values in the vector gradually shift to represent high evaluation solution vectors. Harik, Lobo and Goldberg (1998) proposed the compact GA (cGA) that also evolves a probability vector initialized to 0.5 for each gene locus. This probability vector is also used to generate a set of solutions but is updated with a different learning rule from PBIL.

3 STATISTICS-BASED ADAPTIVE NON-UNIFORM CROSSOVER

3.1 CANONICAL UNIFORM CROSSOVER

Uniform crossover is the generalization of n-point crossover (Syswerda, 1989). It creates offsprings by deciding, for each bit of the parent, whether to swap the allele of that bit with the corresponding allele of the other parent. The decision is made using a coin flip, i.e., the probability of swapping is $p = 0.5$. Figure 1 shows an example of applying the uniform crossover operator to two 6-bit string parents.

Swapping Prob.:	p	p	p	p	p	p
Coin Flipping:	⇓	⇓	⇓	⇓	⇓	⇓
Created Mask:	0	1	0	1	0	1
Applying Mask:		⇓		⇓		⇓
Parent P_1:	0	1	0	1	1	1
Parent P_2:	1	1	1	1	1	0
Swapping:		⇓		⇓		⇓
Child C_1:	0	1	0	1	1	0
Child C_2:	1	1	1	1	1	1

Figure 1: An example operation of the uniform crossover where $p = 0.5$.

When performing uniform crossover on two parents P_1 and P_2, we first generate a mask bit by bit by flipping a coin unbiasedly, i.e., generating 0 or 1 with an equal probability $p = 0.5$. 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 and leaving the bits of other loci of P_1 and P_2 unchanged. As illustrated in Figure 1, the downward arrows marked by "⇓" in the lines of "Applying Mask" and "Swapping" correspond to the positions where there is a "1" in the mask and where the alleles of P_1 and P_2 are exchanged.

In parameterized uniform crossover (Spears and De Jong, 1991), the decision for each locus is made by biased coin flipping, i.e., the swapping probability p could be other than 0.5. However, the degree of bias, i.e., the value of p, is the same for all loci and hence the 1-bits are uniformly distributed over the mask. In fact, the value of p may be different for each locus, that is, the 1-bits may be non-uniformly distributed over the mask. This is realized adaptively in SANUX.

3.2 STATISTICS-BASED ADAPTIVE NON-UNIFORM CROSSOVER

For the convenience of description and analysis, we first introduce the concepts of intrinsic attribute and extrinsic tendency of allele valuing for a gene locus. In the optimal solution (encoded in binary string) 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 solved and encoding scheme, e.g., whether introns are inserted (Levenick, 1995). 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 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). SANUX makes use of this convergence information as feedback information to direct the crossover by adjusting the swapping probability for each locus.

We will 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 a locus's alleles 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. Then, as shown in Figure 2, the calculation equation from $f_1(i,t)$ to $p_s(i,t)$ is given as follows:

$$p_s(i,t) = \begin{cases} f_1(i,t), & \text{if} \quad f_1(i,t) \le 0.5 \\ 1 - f_1(i,t), & \text{if} \quad f_1(i,t) > 0.5 \end{cases} \tag{1}$$

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

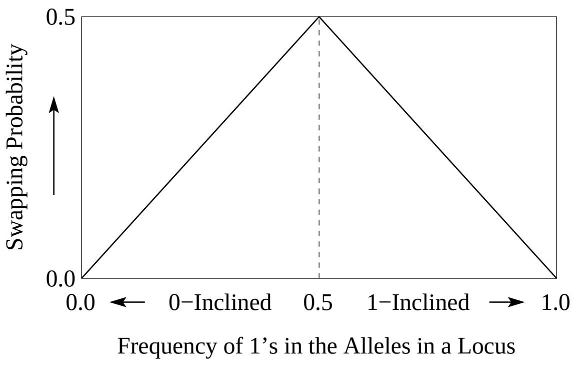

Figure 2: Calculate a locus's swapping probability.

SANUX, the mask is bit by bit generated by flipping a coin biasedly to generate a "1" with probability $p_s(i,t)$. Finally, the generated mask is used to guide the crossover the same way as it guides the traditional uniform crossover. Figure 3 shows an example of applying SANUX to the same parents as in Figure 1.

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

Figure 3: An example operation of SANUX.

3.3 DISCUSSIONS ON SANUX

According to the classification of adaptation for GAs reviewed in section 2, SANUX belongs to the class of adaptive adaptation that occurs at the bottom-level of crossover because it uses the feedback information of allele distribution during the running of the GA and adjusts the swapping probability for each gene locus.

SANUX is much simpler than those adaptation mechanisms that add extra tag bit (e.g., Schaffer and Morishima, 1987; Levenick, 1995) or value (e.g., Rosenburg, 1967; Fraser, 1957b) per genetic bit and co-evolves these tag bits or values with each individual. With SANUX, what we add to traditional uniform crossover are spacially only one real vector that records the frequency of ones for each locus, and computationally only one statistics per generation that calculates the frequency of ones (hence the swapping probability) for each locus. This simple extra statistics added is well rewarded in the sense of computational complexity. For each crossover operation at generation t, the number of swappings on strings of length L is $L/2$ on the average with uniform crossover and $\sum_{i=1}^{i=L} p_s(i,t)$ with SANUX. When the population is randomly initialized, the frequency of 1's in the alleles (hence the swapping probability) in each locus is statistically about 0.5, which gives on the average $L/2$ crossings for SANUX. However, with the running of the GA, 1-intrinsic and 0-intrinsic genes tend to converge to 1 and 0 respectively and their associated swapping probabilities decrease according to equation (1). This results in reduced number of crossings with SANUX, i.e., $\sum_{i=1}^{i=L} p_s(i,t) < L/2$. And there's more. As the population converges, with uniform crossover, in fact, fewer and fewer 1's in the mask will involve a change, that is, many crossings are wasted on those converged bits. Most of these wasted crossings by uniform crossover are saved by SANUX. As illustrated in Figure 1 and 3, after certain generations the distribution of 1's over loci may be as shown in Figure 3, uniform crossover made no use of this information and still generated 3 crossings while SANUX generated 2.

Another more important point of SANUX is its property of implicit gene linkage. Similar to Fraser's crossover (Fraser, 1957b), SANUX adaptively links genes based on their swapping probabilities. For example, in Figure 3 loci 4 and 5 are more convergent and implicitly linked because the probability for them to co-evolve 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.

Here we must note the relationship between SANUX and probability-based algorithms (Baluja, 1994; Harik, 1998). They are all based on probability distribution. However, those algorithms use the explicitly maintained probability vector to sample solutions while SANUX obtains the distribution probability from the statistics information implicit in the population. In the sense of applying probability vector, they are dual algorithms. The motivation of SANUX is to explicitly use the statistics information implicit in the population to guide the crossover.

4 THE TEST PROBLEMS

4.1 THE MAX ONES PROBLEM

The Max Ones problem simply counts the ones contained in a binary string as the fitness of that string. The aim is to maximize ones in a string. A string length of 100 bits was used for our study.

4.2 THE ROYAL ROAD FUNCTIONS

The Royal Road functions (Forrest and Mitchell, 1992) are devised to investigate GA's performance with respect to schema processing and recombination in an idealized form. Royal Road functions R_1 and R_2 contain tailor-made building blocks (schemas) based on 64-bit binary strings. Each schema s_i is given a coefficient c_i which is equal to its order $o(s_i)$ (a schema's order is the number of fixed positions within that schema). R_1 consists of 8 disjunctive order-8 schemas of which each has 8 adjacent ones. R_2 consists of four levels of schemas: level 0 (bottom level) is the same as R_1, level 1 has 4 order-16 schemas of which each combines two adjacent schemas in level 0, level 2 contains 2 order-32 schemas each combining two adjacent schemas in level 1, and finally level 3 (the optimal schema) combines the 2 schemas in level 2.

The fitness of a bit string x for $R_1(x)$ and $R_2(x)$ is computed by summing the coefficients c_i corresponding to each of the given schema s_i of which x is an instance. The optimal solutions for R_1 and R_2 are given as: $R_1(111..1) = 64$ and $R_2(111..1) = 192$.

4.3 THE L-SAT PROBLEM GENERATOR

The random L-SAT problem generator (De Jong, Potter and Spears, 1997) 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 complexity of the problem varies with the parameters V, C and L. For example, increasing the number of clauses increases the epistasis. 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.

In our experiments we used the same parameters as De Jong, Potter and Spears. We fixed the number of variables V to 100 and the length of the clauses L to 3. The number of clauses C is varied from 200 (low epistasis) to 1200 (medium epistasis) to 2400 (high epistasis).

5 EXPERIMENTAL RESULTS

For each experiment of combining crossover operators (1-point, 2-point, uniform crossover and SANUX) and test problems, 100 independent runs were executed. In order to have a strict comparison between crossover operators the same 100 different random seeds were used to generate populations for the 100 runs of each experiment. In all the experiments, the GA uses the fitness proportionate selection with the stochastic universal sampling (Baker, 1987) and the elitist model (De Jong, 1975), 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.

5.1 RESULTS ON MAX ONES PROBLEM

The experiment results for the Max Ones problem are shown in Figure 4. From Figure 4 it can be seen that SANUX outperforms all traditional crossover operators on the Max Ones problem.

5.2 RESULTS ON R_1 AND R_2

The experiment results on royal road functions are shown in Figure 5 and Figure 6 respectively. From these figures it can be seen that during the early stage of GA's searching, 1-point and 2-point are better than SANUX. However, after certain evaluations, when the GA has built up some useful schemas SANUX outperforms them due to its implicit gene linkage. This is also proved by SANUX's consistent advantage over

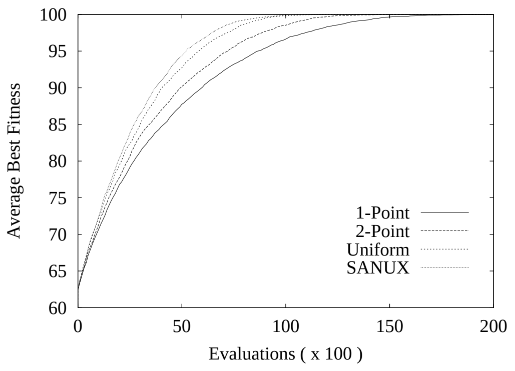

Figure 4: Average best curves for GAs with different crossover on Max Ones problem.

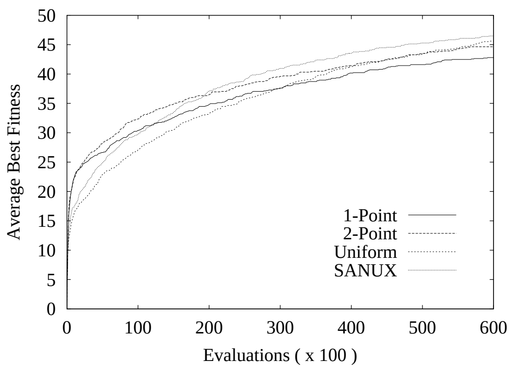

Figure 5: Average best curves for GAs with different crossover on Royal Road function R_1.

uniform crossover on both R_1 and R_2. This is further proved by the observation that SANUX outperforms traditional crossover operators much better on R_2 than on R_1. This happens because the introduction of intermediate schemas into R_2 increases the epistasis on which SANUX shows more advantage than traditional operators due to its implicit gene linkage capability.

5.3 RESULTS ON L-SAT PROBLEMS

The experiment results on L-SAT problems with low, medium and high epistasis are given in Figure 7, Figure 8, and Figure 9 respectively. From these figures it can be seen that during low epistasis all crossover operators work equally as well with uniform crossover slightly better than the other operators. However, as epistasis is increased to medium and high, SANUX outperforms all traditional crossover operators. This further shows that SANUX has more advantage over traditional crossover on problems with high epistasis due to its implicit gene linkage capability.

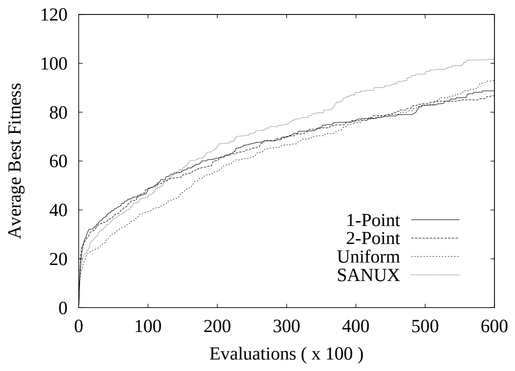

Figure 6: Average best curves for GAs with different crossover on Royal Road function R_2.

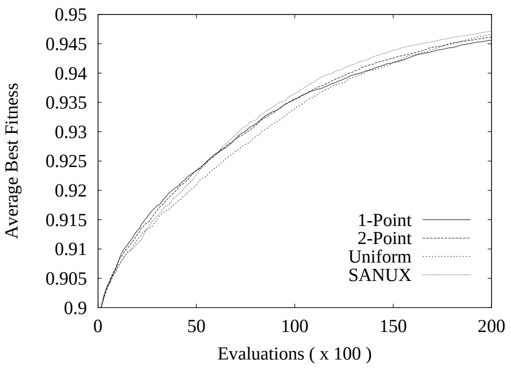

Figure 8: Average best curves for GAs with different crossover on L-SAT problems with medium epistasis.

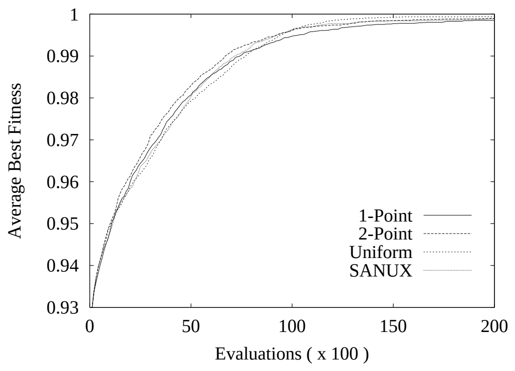

Figure 7: Average best curves for GAs with different crossover on L-SAT problems with low epistasis.

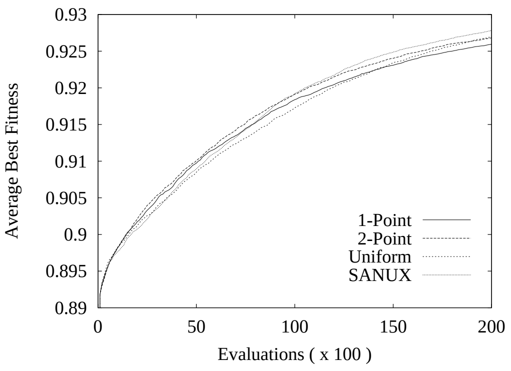

Figure 9: Average best curves for GAs with different crossover on L-SAT problems with high epistasis.

6 CONCLUSIONS

In this paper, a new statistics-based adaptive non-uniform crossover operator, SANUX, is proposed. The motivation of SANUX is to make use of the statistics information implicitly contained in the population explicitly to guide the crossover operation. SANUX achieves this by using the statistics information of the allele distribution in the current population to adjust the swapping probability for each gene locus adaptively during the evolutionary progress of the GA. An important property of SANUX is its capability of implicitly linking gene groups. SANUX is also computationally efficient through saving crossings.

The preliminary experiment results of this study show that SANUX performs better than traditional one-point, two-point and uniform crossover operators on a set of typical GA's test problems. The experiment results indicate that SANUX may be a good candidate as a crossover operator for GAs. Since SANUX works at the bottom-level of crossover, it can be easily combined into the other two levels of adaptation in crossover. Additionally, SANUX is dual to those probability-based algorithms in the sense of using probability vector and thus may act as the basis for analyzing and designing new related algorithms.

References

J. E. Baker (1987). Reducing bias and inefficiency in the selection algorithms. In J. J. Grefenstelle (ed.), *Proc. 2nd Int. Conf. on Genetic Algorithms*, 14-21.

S. Baluja (1994). Population-based incremental learning. Technical Report CMU-CS-95-193, Carnegie Mellon University.

L. B. Booker (1992). Recombination distributions for genetic algorithms. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2*, 29-44.

D. Corne, P. Ross and H.-L. Fang (1994). GA research note 7: fast practical evolutionary timetabling. Technical Report, Department of Artificial Intelligence, University of Edinburgh, UK.

L. Davis (1989). Adapting operator probabilities in genetic algorithms. In D. Schaffer (ed.), *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 60-69.

K. A. De Jong (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Abor.

K. A. De Jong, M. A. Potter and W. M. Spears (1997). Using problem generators to explore the effects of epistasis. In T. Bäck (ed.), *Proc. of the 7th Int. Conf. on Genetic Algorithms*, 338-345.

A. E. Eiben, R. Hinterding, and Z. Michalewicz (1999). Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation* **3**(2):124-141.

L. Eshelman, R. Caruana, and J. D. Schaffer (1989). Biases in the crossover landscape. In *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 10-19.

L. Eshelman and J. D. Schaffer (1994). Productive recombination and propagating and preserving schemata. In M. Vose and D. Whitley (eds.), *Foundations of Genetic Algorithms 3*, 299-313.

A. S. Fraser (1957a). Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian Journal of Biological Sciences* **10**:484-491.

A. S. Fraser (1957b). 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.

S. Forrest and M. Mitchell (1992). Relative building-block fitness and the building-block hypothesis. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2.*

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley,

G. E. Goldberg, B. Korb and K. Deb (1989). Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems* **3** (5):493-530.

G. Harik and G. E. Goldberg (1996). Learning linkage. In *Foundations of Genetic Algorithms 4*, 247-272.

G. Harik, F. Lobo and G. E. Goldberg (1998). The compact genetic algorithm. In *Proc. of the 1998 IEEE Conference on Evolutionary Computation*, 523-528.

J. H. Holland (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

B. Julstrom (1995). What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In L. J. Eshelman (ed.), *Proc. of the 6th Int. Conf. on Genetic Algorithms*, 81-87.

J. Levenick (1995). Metabits: genetic endogenous crossover control. In L. J. Eshelman (ed.), *Proc. of the 6th Int. Conf. on Genetic Algorithms*, 88-95.

J. Reed, R. Toombs, and N. A. Barricelli (1967). Simulation of biological evolution and machine learning: I. Selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology* **17**:319-342.

R. S. Rosenberg (1967). *Simulation of Genetic Populations with Biochemical Properties.* PhD Thesis, University of Michigan, Ann Abor.

J. D. Schaffer and A. Morishima (1987). An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstelle (ed.), *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, 36-40.

J. E. Smith and T. C. Fogarty (1996). Recombination strategy adaptation via evolution of gene linkage. In *Proc. of the 3rd IEEE Int. Conf. on Evolutionary Computation*, 826-831.

W. M. Spears and K. A. De Jong (1991). On the virtues of parameterized uniform crossover. In *Proc. 4th Int. Conf. on Genetic Algorithms*, 230-236.

W. M. Spears (1995). Adapting crossover in evolutionary algorithms. In *Proc. of the 4th Annual Evolutionary Programming Conference*, 367-384.

W. M. Spears (1997). Recombination parameters. In T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*, E1.3.1-E1.3.11. Oxford University Press.

G. Syswerda (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (ed.), *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 2-9.

A. Tuson and P. Ross (1998). Adapting operator settings in genetic algorithms. *Evolutionary Computation* **6**(2):161-184.

K. Vekaria and C. Clarck (1999). Biases introduced by the adaptive recombination operators. In T. Bäck (ed.), *Proc. of the 1999 Genetic and Evolutionary Computation Conference*, 670-677. San Mateo, CA: Morgan Kaufmann.

T. White and F. Oppacher (1994). Adaptive crossover using automata. In Y. Davidor, H. Schwefel and R. Männer (eds.), *Proc. of the 3rd Int. Conf. on Parallel Problem Solving from Nature*, 229-238.