Adaptive Genetic Algorithm with Mutation and Crossover Matrices

Nga Lam Law and K.Y. Szeto
Hong Kong University of Science and Technology
Department of Physics
Clear Water Bay, Hong Kong SAR, China
Corresponding author: phszeto@ust.hk

Abstract

A matrix formulation for an adaptive genetic algorithm is developed using mutation matrix and crossover matrix. Selection, mutation, and crossover are all parameter-free in the sense that the problem at a particular stage of evolution will choose the parameters automatically. This time dependent selection process was first developed in MOGA (mutation only genetic algorithm) [Szeto and Zhang, 2005] and now is extended to include crossover. The remaining parameters needed are population size and chromosome length. The adaptive behavior is based on locus statistics and fitness ranking of chromosomes. In crossover, two methods are introduced: Long Hamming Distance Crossover (LHDC) and Short Hamming Distance Crossover (SHDC). LHDC emphasizes exploration of solution space. SHDC emphasizes exploitation of local search process. The one-dimensional random coupling Ising Spin Glass problem, which is similar to a knapsack problem, is used as a benchmark test for the comparison of various realizations of the adaptive genetic algorithms. Our results show that LHDC is better than SHDC, but both are superior to MOGA, which has been shown to be better than many traditional methods.

1 Introduction

The notion of mutation matrix is used in the development of adaptive genetic algorithm so that the selection process does not require any external input parameter. Numerical examples on knapsack problems indicate that this new adaptive genetic algorithm, called MOGA (mutation only genetic algorithm) is superior to many traditional methods [Ma and Szeto, 2004; Szeto and Zhang, 2005; Zhang and Szeto, 2005]. The idea behind MOGA is that mutation probability is a function of time, fitness ranking of the chromosome, and locus. The dependence is automatically determined by the statistics of the locus and the fitness ranking of the chromosome at the given time. The traditional genetic algorithm is a special case of MOGA in which all the mutation probabilities are fixed. In this paper, we extend the basic idea behind MOGA to include the crossover process. Section 2 reviews the MOGA formalism without crossover. Section 3 focuses on the implementation of the crossover matrix. A brief explanation of the test problem - Ising spin glass is given in section 4, and the test results on Ising spin glass are discussed in section 5.

2 Mutation Matrix

Traditional genetic algorithm can be considered as a special case of the mutation only genetic algorithm (MOGA). For a population of N_P chromosomes, each encoded by L locus, can be arranged into a population matrix A with N_P rows and L columns. Each element of the matrix A has a probability p $(0 < p < 1)$ to undergo mutation. These mutation probabilities $M_{ij}(i = 1, ..., N_P; j = 1, ..., L)$ form a mutation matrix M: M_{ij} is the mutation probability of A_{ij}. In matrix A, each row is one chromosome. The rows are listed in decreasing order of the fitness: $f_i \geq f_j$ if $i < j$.

In traditional genetic algorithm, the best $N_1 = c_1 \times N_P$ chromosomes survive $(0 < c_1 < 1)$. These survivors produce $N_2 = c_2 \times N_P$ children to replace the less fit ones through crossover and/or mutation $(0 < c_2 < 1 - c_1)$. The remaining $N_3 = (1 - c_1 - c_2) \times N_P$ unfit chromosomes are replaced by the same number of randomly generated candidates in order to ensure population diversity. Thus the traditional genetic algorithm can be described by a mutation matrix which has $M_{ij} = 0$ for the first N_1 rows; $M_{ij} = m(0 < m < 1)$ for the next N_2 rows; $M_{ij} = 1$ for all remaining rows. In traditional genetic algorithm, we must input three time independent parameters: c1, c2 and m.

MOGA makes use of the information of the fitness ranking and loci statistics to determine M_{ij} dynamically. It outperforms traditional genetic algorithm in terms of both speed and quality of solution. In this formalism, the mutation matrix is nearly the same as the population matrix A, but with the loci arranged according to the standard deviation $\sigma(j)$ for the allele distribution. If there are V allele values possible for the locus, (in ordinary binary string representation of genetic algorithm, the possible value of an allele is either 0 or 1, so V = 2), then we define the standard deviation $\sigma(j)$ for the allele distribution as:

$$\sigma(j) = \sqrt{\frac{\sum_{i=1}^{N_P}(A_{ij} - \overline{h(j)})^2 \times C(f(i))}{\sum_{i=1}^{N_P} C(f(i))}} \qquad (1)$$

where

$$\overline{h(j)} = \frac{1}{N_P} \sum_{i=1}^{N_P} A_{ij}. \quad (2)$$

Here, C(f(i)) is the fitness cumulative probability of chromosome i, which, as well as fitness, can be considered as an informative measure of chromosome i. High fitness cumulative probability indicates high informative measure. The allele standard deviation $\sigma(j)$ for each loci can also be regarded as an informative measure. A locus which has a smaller allele standard deviation contains more useful information than the other loci.

At the beginning of the algorithm, chromosomes are randomly generated. Alleles are randomly distributed. Thus the allele standard deviation is not so informative. After generations of evolution, chromosomes acquire higher fitness, good chromosome structures emerge. Then more chromosomes evolve to acquire the good structures. Therefore, a locus with high structural information has a small allele standard deviation.

The presence of the factor of $\frac{C(f(i))}{\sum_{i=1}^{N_P} C(f(i))}$ in the allele standard deviation is to take account of the informative usefulness of each chromosome. Loci in chromosomes with high fitness generally are more informative than those in chromosomes with low fitness. With this factor, the allele standard deviation more appropriately reflect the amount of information contained in a particular locus. Our cumulative probability $C(f(i))$ is defined by

$$C(f(i)) = \frac{1}{N_P} \sum_{g \le f} N(g). \quad (3)$$

N(g) is the number of chromosomes with fitness value g. Note that C(f) is a non-decreasing function of f where $C(f_{max}) = 1$ and $(\frac{1}{N_P} \le C(f) \le 1)$.

Therefore, the mutation matrix can be viewed as an arrangement of genes based on the informative measures: the fitness cumulative probability $C(f(i))$ and the standard deviation $\sigma(j)$ for the allele distribution. The higher the row is, the more useful information the loci in this row contains. Similarly, the closer to the right side the locus is, the more information it contains. Our mutation is to make use of these two informative measures through the mutation matrix.

In each generation, rows (i.e. chromosomes) in the mutation matrix were first selected for mutation based on the fitness cumulative probability. The probability $\alpha(i)$ to choose the ith row for mutation is defined as,

$$\alpha(i) = 1 - C(f(i)). \quad (4)$$

Therefore, a chromosome with higher fitness has a lower probability to participate in mutation, or in other words, it has higher ability to survive. As the chromosomes are kept in a decreasing order of the fitness: $f_i \ge f_j$ if $i < j$, in the matrix A; $\alpha(i)$ is a non-decreasing function of the row index (chromosome index) i. Thus, $\alpha(i)$ is higher for less fit rows (chromosome). The fittest chromosome, i.e. the first row of the population matrix A, $\alpha(i) = 0$. It is, hence, never selected for mutation. The worst chromosome, i.e. the last row of A, will mutate with a probability close to unity for large enough N_P: $\alpha(N_P) = 1 - \frac{1}{N_P} \approx 1$, e.g. $\alpha(N_P) = 0.95$ for $N_P = 20$.

Next, we employ the fitness cumulative probability again to determine the number N_{mg} of loci to undergo mutation for the selected chromosomes, each with L loci. Stating the formula for N_{mg},

$$N_{mg} = \alpha(i) \times L. \quad (5)$$

It is obvious that N_{mg} is also a function of the row index i, or in GA language, the fitness. If a selected chromosome has a high fitness, it has fewer loci to undergo mutation. If a selected chromosome has a low fitness, it has more loci to mutate. Finally, we mutate the N_{mg} leftmost loci located in the selected rows in the mutation matrix, i.e. the N_{mg} least informative loci of each chromosome. In this way, most of the informative loci remain and guide the evolution process to the next generation, while most of the less informative loci mutate until they contain more information.

3 Crossover Based on Hamming Distance

With the implementation of the mutation matrix, the crossover is operated implicitly on the population. The implicit crossover deploys the chromosome fitness ranking and the loci statistics in search of solution. Besides these two kinds of information, the Hamming distance can also be used to search the solution. Through the introduction of a crossover matrix, two different explicit crossover methods which employ the information of the Hamming distance are implemented and compared.

The distance matrix H is a square $N_P \times N_P$ matrix with matrix element $H_{ii'}$ defined by a distance measure between the i-th chromosome, represented by $\overrightarrow{R}_i$ which is the i-th row vector of the population matrix A, and the i'-th chromosome $\overrightarrow{R}_{i'}$: $H_{ii'} = D(\overrightarrow{R}_i, \overrightarrow{R}_{i'})$. In this paper, we have two different measures for H. The first one is called long Hamming distance crossover (LHDC) in which $H_{ii'} = H^L_{ii'} = D_{ii'}$, where $D_{ii'}$ is equal to the number of different alleles in these two rows. The second measure is called short Hamming distance crossover (SHDC), with $H_{ii'} = H^S_{ii'} = L - D_{ii'}$. We see that in LHDC, the smaller the distance between two chromosomes i and i' is, the more similar they are. While in SHDC, it is just the opposite: smaller distance indicates low similarity. Note that the matrix H is a symmetric matrix. The diagonal entries of H are zero in LHDC, but equal L in SHDC.

After defining the matrix H, we have to select two chromosomes for crossover. The probabilities to choose the two chromosomes are different, but they are both related to H. One probability is just the fitness cumulative probability, and we call it the first crossover probability $P_{CI}(i)$.

$$P_{CI}(i) = C(f(i)). \quad (6)$$

A chromosome, which we call the first chromosome, is first chosen with this first crossover probability. As the row index in H is also the row index in A, chromosome represented by smaller H row index has higher probability to be selected for crossover. Then, the second chromosome is selected with another probability called the second crossover probability. The

second crossover probability depends on the chromosome already chosen. If chromosome i (i.e. chromosome with H row index i), is the first chromosome, then the second crossover probability $P_{CII}(i')$ to choose chromosome i' as the second chromosome is:

$$P_{CII}(i') = \frac{H_{ii'}}{\sum_{k=1}^{N_P} H_{ik}}. \quad (7)$$

If the first and the second chromosome are the same, the second one is chosen again until they are both different. Both LHDC and SHDC are single-point crossovers with the crossing points chosen randomly once in a generation. In each generation, a total of N_P chromosomes are selected to participate in the crossover. There is no limitation on the number of times a chromosome can take part in crossover in one generation. Later, the fittest N_P chromosomes are selected to form a new generation from the pool of N_P original chromosomes and the N_P children chromosomes produced by crossover.

In each generation, we first carry out crossover and then mutation. At the end of crossover, i.e. when the fittest N_P chromosomes are selected, as well as at the end of the mutation, we update the fitness and cumulative fitness probability for all the chromosomes in the population.

4 Application: One-Dimensional Ising Spin Glass

The algorithm is applied to find the minimum energy of a 1D periodic Ising model consisting of N_{loci} spins (N_{loci} = 10, 11, 15, 20, 25 and 30). Each spin can be either up or down, represented by '1', or '-1' respectively. The interaction energy between a pair of neighbouring Ising spin is randomly generated in the range [-1,1]. The problem is to find the minimum of the energy

$$E = \sum_{j=1}^{N_{loci}-1} J_{j,j+1} s_j s_{j+1} \quad (8)$$

for a given set of interaction energy . For each value of N_{loci}, two different sets of interaction energy were generated. For each set of interaction energy, the exact minimum energy E_{min} was first computed by exhaustive search over $2^{N_{loci}}$ spin configurations. The program is then executed 400 times with a fixed population size of $N_P = 20$ chromosomes. Each execution is regarded as one sample. An execution stops when an acceptable solution is found or the maximum allowed generations $N_{MA} = 250000$ is reached. An acceptable solution is a candidate solution which is within 1% tolerance of the exact minimum energy E_{min}. When no solution can be found within N_{MA} generations in an execution, the execution is considered as a failure sample.

5 Results

We have compared the efficiency of three adaptive genetic algorithms (MOGA, LHDC and SHDC) on the spin glass problem. Table 1 summarizes the average generation to reach solution and the number of failure. Both GAs with crossover, LHDC and SHDC, outperform MOGA in terms of average generation and number of successful searches. Moreover LHDC gives the best performance.

N_{loci}	sample	MOGA	LHDC	SHDC
10	1	20 (0)	6 (0)	6 (0)
	2	20 (0)	7 (0)	7 (0)
15	1	146 (0)	14 (0)	14 (0)
	2	318 (0)	49 (0)	65 (0)
20	1	2456 (0)	52 (0)	47 (0)
	2	1483 (0)	159 (0)	197 (0)
25	1	41274 (1)	171 (0)	194 (0)
	2	31144 (0)	586 (0)	669 (0)
30	1	123621 (0)	1724 (0)	1956 (0)
	2	123460 (280)	7610 (0)	11863 (0)

Table 1: Average generation and number of failure in 400 samples to reach solution using the MOGA, LHDC and SHDC. The number of failure is written in the round brackets.

The two GAs with crossover is better due to the information exchange between chromosomes. In LHDC, fit chromosome is selected and crossover with a very different chromosome to generate two new chromosomes. This expands the exploration space and minimizes the chance of premature convergence. In SHDC, fit chromosome is selected and crossover with a similar one. This exploits the local search process and speeds up convergence. The results suggest that a second crossover probability emphasizing the exploration of search space is better for the Ising-spin-glass-like problem (i.e. knapsack-type problem)[Kellerer *et al.*, 2004].

6 Conclusion

Three different ways to implement the adaptive genetic algorithm are tested and compared using the one-dimensional Ising model for spin glass as the testing problem. The results show that LHDC is best and that all three methods are parameter-free. Since MOGA outperforms other algorithms, including dynamic programming [Kellerer *et al.*, 2004; Simoes and Costa, 2001; 1999], for solving the knapsack problem, our test shows the superiority of LHDC in this kind of search problem. This work also provides a matrix formulation of parameter free adaptive genetic algorithm with both mutation and crossover.

Acknowledgments

N.L.Law and K.Y. Szeto acknowledge interesting discussion from C.W. Ma. K.Y. Szeto acknowledges the support of RGC grant 603203.

References

[Kellerer *et al.*, 2004] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*, pages 22–26. Springer, 2004.

[Ma and Szeto, 2004] Chun Wai Ma and Kwok Yip Szeto. Locus oriented adaptive genetic algorithm: Application to the zero/one knapsack problem. In *Proceeding of The 5th International Conference on Recent Advances in Soft Computing, RASC2004*, pages 410–415, Nottingham, UK, December 2004.

[Simoes and Costa, 1999] A. B. Simoes and E. Costa. Transposition versus crossover: An empirical study. In *Proceeding of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 1999.

[Simoes and Costa, 2001] A. B. Simoes and E. Costa. An evolutionary approach to the zero/one knapsack problem: Testing ideas from biology. In *Proceeding of Fifth International Conference on Aritical Neurakl Network and Genetic Algorithms*, 2001.

[Szeto and Zhang, 2005] Kwok Yip Szeto and J. Zhang. Adaptive genetic algorithm and quasi-parallel genetic algorithm: Application to low-dimensional physics. In *Lecture Notes in Computer Science 3743*, pages 186–196, Sozopol, June 2005. Springer-Verlag.

[Zhang and Szeto, 2005] Jian Zhang and Kwok Yip Szeto. Mutation matrix in evolutionary computation: An application to resource allocation problem. In *Lecture Notes in Computer Science 3612*, pages 112–119, Changsha, China, August 2005. Springer-Verlag.