The Lightweight Genetic Search Algorithm: An Efficient Genetic Algorithm for Small Search Range Problems

Chun-Hung Lin and Ja-Ling Wu
Communication and Multimedia Lab. Dept. of Comp. Sci. and Inf. Eng.
National Taiwan University, Taipei, 106, Taiwan, R. O. C.

Abstract

In this paper, the effectiveness of the genetic operations of the common genetic algorithms, such as crossover and mutation, are analyzed for small search range situations. As expected, the so-obtained efficiency/performance of the genetic operations is quite different from that of their large search range counterparts. To fill this gap, a lightweight genetic search algorithm is presented to provide an efficient way for generating near optimal solutions for these kinds of applications.

1 INTRODUCTION

Genetic algorithms (GAs) have been developed and applied to a variety of optimization and search problems [1]. In the applications with large amounts of search points, it is impossible, due to execution time and storage space, to perform the brute-force (full) search for visiting all the points in the search space. GAs can help to find the global optima although a few computational overheads of the genetic evolution are required. While compared with the computational complexity of the full search, the evolution overheads are small and worthwhile. Nevertheless, when the search space is very small, the genetic evolution overheads might overtake the computational complexity of applying the full search. This implies, in this situation, it would be better to perform the full search directly. However, the time constraint for these small search range applications is usually very tight. For example, in the motion estimation stage of video coding [2], the ideal execution time for each 16×16 block search must be less than 3.95×10^{-8} second. The computational complexity of the full search is still too high to satisfy the above requirement. The evolution overheads of GAs have to be reduced, so as to meet the embedded strict time constraint in the above applications.

GAs have been applied to these kinds of applications in the literature [3, 4]. The huge computational complexity of the traditional GA-based search algorithms has made them become handicaps in real video coding applications. In this paper, a lightweight genetic search algorithm (LGSA) will be presented. When GAs are applied in searching, the computational complexity comes mainly from the following two parts: (1) the computations of evaluating the similarity between the search points and the reference template; (2) the computations of the genetic evolution. The first part is dominated by the number of the evaluated search points. Fewer computations would be required if the number of the evaluated points is reduced. But, this reduction involves the risk of finding a bad solution. The second part is controlled by the structure of the genetic evolution. Low control overheads bring less computational complexity. In the LGSA, both the number of the evaluated points and the control overheads of genetic evolution can be reduced to meet the time constraint of video coding.

2 ANALYSIS OF CROSSOVER EFFICIENCY

In conventional GAs, the evolution is analyzed based on the schema theory [1]. The crossover disruption rates were defined as the probability that the schema is disrupted by crossover. For a good genetic algorithm, it is hoped that the crossover disruption rate of a highly fit schema will be very low. In [1], an upper bound for the crossover disruption rate of a schema was given.

Based on the primitive results of the schema theory, a further analysis was provided in [6]. It is concluded that at least in two situations where disruption is advantageous: (1) when the population is quite homogeneous, and (2) when the population size is too small to

0-7803-4869-9/98 $10.00©1998 IEEE

provide the necessary sampling accuracy for complex search spaces. It is also known that smaller population sizes tend to become homogeneous more quickly. Because population sizes are small when the search space is not large [7], the disruption will also be favorite in a small search space. The crossover productivity rate (i.e. disruption rate) will be denoted as P_{eff}. It is hoped that the crossover productivity rate will be large enough when the population becomes homogeneous.

Besides the above two situations, disruption is also advantageous when real-time constraint is a must. In this situation, the generation number of genetic evolution is not allowed to be large. In order to probe more search points, the crossover productivity rate should be high, especially when an elitist selection is applied in the reproduction stage.

In the previous analyses, the crossover disruption rates are discussed based on schemata (chromosome patterns) while applying conventional GAs. However, it is difficult to determine the fitness of a schema in the applications with various kinds of search spaces. An analysis of the crossover productivity rates based on a randomly given chromosome will be provided in the following.

For an n-point crossover, the crossover productivity rate can be derived as follows. The n crossover points divide each chromosome into $(n+1)$ segments, i.e., $C_a = C_{a_1} + C_{a_2} + \cdots + C_{a_{n+1}}$ and $C_b = C_{b_1} + C_{b_2} + \cdots + C_{b_{n+1}}$. At the initial derivation, n is assumed to be an odd number. An n-point crossover can be identically represented as shown in Fig. 1(b). The corresponding crossover productivity rate can then be calculated as

$$P_{eff}^{(n)} = 1 - P_{inv}^{(n)}, \quad (1)$$

$$\begin{aligned} P_{inv}^{(n)} &= (P_1 \cap P_3 \cap \cdots \cap P_n) \cup (P_2 \cap P_4 \cap \cdots \cap P_{n+1}) \\ &= P_1 P_3 \cdots P_n + P_2 P_4 \cdots P_{n+1} - \prod_{i=1}^{n+1} P_i \\ &= \frac{1}{C_n^{k-1}} \sum_{i=\frac{n+1}{2}}^{k-\frac{n+1}{2}} [C_{\frac{n-1}{2}}^{i-1} C_{\frac{n-1}{2}}^{k-1-i} (\frac{1}{2^i} + \frac{1}{2^{k-i}} - \frac{1}{2^k})] \\ &= \frac{2}{C_n^{k-1}} \sum_{i=\frac{n+1}{2}}^{k-\frac{n+1}{2}} [C_{\frac{n-1}{2}}^{i-1} C_{\frac{n-1}{2}}^{k-1-i} \frac{1}{2^i}] - \frac{1}{2^k}. \end{aligned} \quad (2)$$

3 ANALYSIS OF MUTATION EFFICIENCY

In conventional GAs, mutation is applied to probe the search points that can not be reached by the crossover operations. The mutation probability is usually small so as not to spoil the population and ruin the chromosomes with good fitness. When the crossover operations are taken out from the genetic evolution, due to their inefficiency in a small search space situation, the task of probing new search points must be done by the mutation operations. It should be guaranteed that every legal search point can be reached by mutating any selected chromosomes. The necessary generation number for a selected chromosome to be mutated into any other chromosomes should also be small if the time constraint is an important issue. For a selected chromosome, it is hoped that the chromosome can change to any other chromosomes with higher probability. If an efficient mutation operation is applied, the probability of transferring any given chromosome to the other chromosomes should be identical after a small number of iterations, such that there is not any search point which is hard to be reached from the selected chromosome.

To improve the mutation efficiency, the mutation probability has to be increased; however, the accumulated evolution information will also be destroyed which makes the genetic search somewhat like the random search. Unless an elitist selection is applied in the reproduction stage to maintain the evolution information, the genetic search will degrade to the random search if the mutation probability is high. When an elitist selection is applied to avoid the disruption of population by mutation, the mutation invariance (or unchanged) rate for a selected chromosome would be very low. The probability of transferring one chromosome to other different chromosomes is proportional to the mutation efficiency.

The probability for a chromosome C_i to be mutated into another chromosome C_j is

$$P_{eff}(i,j) = P_m^{H(C_i,C_j)} (1 - P_m)^{k - H(C_i,C_j)}, \quad (3)$$

where P_m is the mutation probability, k the chromosome length, and $H(C_i, C_j)$ the Hamming distance between C_i and C_j. A mutation efficiency matrix can then be formed as

$$M^t = [m_{ij}^{(t)}], \quad (4)$$

where t is the generation number and

$$m_{ij}^{(1)} = P_{eff}(i,j). \quad (5)$$

Assume the selected chromosome is strong (i.e., with high fitness value), such that it and its offsprings will always be selected in the reproduction stage. When the crossover stage is disabled, the mutation efficiency can be calculated according to M^t.

Fig. 2 shows the mutation efficiency for the case of chromosome length $k = 2$. In Fig. 2(a), the average mutation disruption rates with three different mutation probabilities $P_m = 0.01$, 0.5, and 0.99, are depicted, where the mutation disruption rate is defined to be the mean value of the probabilities for a selected chromosome being mutated into distinct chromosomes in the search space. The variances of the probabilities of being mutated into various chromosomes are shown in Fig. 2(b).

In conventional GAs, a small mutation probability is usually selected, for example, $P_m = 0.01$ is suggested in [8]. It can be seen from Fig. 2(a) that the average mutation disruption rate is very low in the early generations. The average disruption rate increases slowly while the generation number increases. The average disruption rate will reach the upper bound 0.25 after more than one hundreds generations. In order to promote the mutation efficiency, the mutation probability is increased. If the mutation probability approaches 1, e.g. $P_m = 0.99$, it is found that the average disruption rate will bounce between two extreme values. Moreover, by comparing Fig. 2(a) and (b), it is found that the variance of the disruption rates is very large when the average disruption rate is high. The probabilities of transferring each chromosome to all other chromosomes are still very low. Hence, increasing the mutation probability will not improve the mutation efficiency well. If a compromised value is used, e.g. $P_m = 0.5$, the average disruption rate will retain to 0.25. Under this condition, the mutation efficiency will be better than that of the previous two cases; however, the invariance rate will still be $1 - 0.25 \times 3 = 0.25$. This invariance rate is still too high to improve the mutation efficiency for each chromosome.

Fig. 3 shows the mutation efficiency for the case of $k = 5$. The results are very similar to that of the case of $k = 2$. Because, when the search space is larger there are more chromosomes, the average disruption rates are therefore becoming smaller since the disruption rates should be always less than 1. The chromosome length is larger, in this case, so that the mutation efficiency is better ($0.03 \times 31 > 0.25 \times 3$). A smaller generation number is required for the curves of different mutation probabilities to converge to the value 1/32. Although the mutation efficiency is better when the chromosome length is larger, the resultant efficiency is still not high enough.

In the LGSA, each gene of a chromosome is altered by adding one of the following three values, $0, 1, -1$, with identical probabilities, for every k generations. In the kth generation, the mutation invariance rate is equal to $1/2^k$. The mutation efficiency of the LGSA, in the kth generation, is shown in Table. 1, where two kinds of chromosome lengths are tested. It is found that, within a small number of generations, the mutation efficiency of the LGSA will be better than that of the conventional ones. Moreover, the corresponding average mutation disruption rates will be higher and the variances of the disruption rates will be smaller. Notice that the mutation invariance rates of the LGSA are reasonably small as compared with that of the conventional mutation approach, as shown in Figs. 2 and 3.

4 THE LIGHTWEIGHT GENETIC SEARCH ALGORITHM

Assume the central point of the two-dimensional search space S locates at $(\hat{x}, \hat{y})$. The ith chromosome, C_i for $i = 0, 1, \cdots, N-1$, of the population set is defined as

$$C_i = \begin{bmatrix} m_i \\ n_i \end{bmatrix} = \begin{bmatrix} a_{i,k-1} & a_{i,k-2} & \cdots & a_{i,0} \\ b_{i,k-1} & b_{i,k-2} & \cdots & b_{i,0} \end{bmatrix} \quad (6)$$

and the relative location is

$$(m_i, n_i) = (x_i - \hat{x}, y_i - \hat{y}), \quad (7)$$

where (x_i, y_i) denotes the location of the search point associated with the chromosome and the codeword size k depends on the size of the search space. If the search space is $\{(i,j) | -w \leq i, j \leq w-1\}$, the value of k will be $\lceil \log_2(2w) \rceil$, where $\lceil \cdot \rceil$ is a ceiling function. The values of the genes are derived from the associated relative location, that is

$$a_{i,j} = \lfloor \frac{m_i + 2w}{2^j} \rfloor \ mod \ 2, \quad (8)$$

$$b_{i,j} = \lfloor \frac{n_i + 2w}{2^j} \rfloor \ mod \ 2, \quad j = 0, 1, \cdots, k-1, \quad (9)$$

where *mod* denotes the module operation and $\lfloor \cdot \rfloor$ is a floor function. The relative location can be one-to-one encoded into a series of genes with values of 0 and 1. And the relative location (m_i, n_i) can be calculated from the values of the genes by

$$m_i = \sum_{j=0}^{k-1} a_{i,j} \cdot 2^j - 2a_{i,k-1}w, \quad (10)$$

$$n_i = \sum_{j=0}^{k-1} b_{i,j} \cdot 2^j - 2b_{i,k-1}w. \quad (11)$$

Although the values of the genes might not equal 0 or 1 after mutation; however, they can be transferred to a relative location without any ambiguity.

Each chromosome has an associated fitness value which is defined as

$$f_i = U_{D(d_i,\hat{d}_\tau)} \cdot D(d_i, \hat{d}_\tau) + \delta_{D(d_i,\hat{d}_\tau)}, \quad (12)$$

where d_i is the matching value of the search point represented by the ith chromosome, D is a difference function, and U and δ are the unit step function and the delta function, respectively. When the target of the search problem is to find a point with the minimum matching value, the difference function is defined as

$$D(d_i, \hat{d}_\tau) = \hat{d}_\tau - d_i, \quad (13)$$

where $\hat{d}_\tau$ is the τth minimum matching value among the N values, $\{d_i | i = 0, 1, \cdots, N-1\}$. If the target is the global maximum, the difference function is defined as

$$D(d_i, \hat{d}_\tau) = d_i - \hat{d}_\tau, \quad (14)$$

where $\hat{d}_\tau$ is the τth maximum matching value. The constant τ determines how many chromosomes, at most, should be selected as seeds in the reproduction stage for producing a rival population. The chromosomes with larger fitness values, in the current population set, will have higher probability to be selected as seeds for generating the rival population. This probabilistic scheme of selecting the seeds of the new generation is known as the probabilistic reproduction. Because the value of $U_{D(d_i,\hat{d}_\tau)}$ is either 0 or 1, there needs no multiplication for computing the fitness values.

The reproduction method used in this work is similar to the weighted roulette wheel method [1]. For each chromosome C_i, an incidence range r_i is calculated as

$$r_i = \left[\frac{\sum_{k=0}^{i-1} f_k}{\sum_{k=0}^{N-1} f_k}, \frac{\sum_{k=0}^{i} f_k}{\sum_{k=0}^{N-1} f_k} \right), \quad (15)$$

where f_k is the fitness value of the kth chromosome in the population, and '[' and ')' denote closing and opening boundaries, respectively. When the incidence range of each chromosome has been determined, N real numbers α_i for $i = 0, 1, \cdots, N-1$, are randomly generated, where $0 \leq \alpha_i < 1$. The value of α_i will be bounded by some incidence range r_j, that is, $\alpha_i \in r_j$. The jth chromosome C_j is then selected as a seed to generate the rival population. It is possible that one chromosome can be selected twice or more. Because N real random numbers are generated, N seeds will be selected and placed in the mating pool.

After the reproduction stage, the seeds in the mating pool are transferred into candidate chromosomes of the new population set by mutation. Assume the current chromosome to be processed is $[m_i \ n_i]^t$, where $m_i = [a_{i,k-1}a_{i,k-2}\ldots a_{i,0}]$ and $n_i = [b_{i,k-1}b_{i,k-2}\ldots b_{i,0}]$. In the jth generation, two genes $a_{i,z}$ and $b_{i,z}$ are varied, where $z = k - 1 - (j \bmod k)$.

There are eight mutation operators, $\{(\zeta_p, \eta_p) | p = 0, 1, \cdots, 7\}$, which can be adopted in our implementation, that is

$$a'_{i,z} = a_{i,z} + \zeta_p, \quad (16)$$

$$b'_{i,z} = b_{i,z} + \eta_p, \quad (17)$$

where p is a random integer whose value is between 0 and 7. Because the chromosomes are randomly selected and put on the mating pool, it is not necessary to generate a random number for determining the value of p. We simply set p to be $(i \ mod \ 8)$. The mutation operators are therefore defined as,

$$\zeta_p = (-1)^l([p + 1 - l(l+1)] \cdot [1 - (\lfloor 2\sqrt{p+1} \rfloor \ mod \ 2)] + \lceil \frac{1}{2} l \rceil), \quad (18)$$

$$\eta_p = (-1)^l([p + 1 - l(l+1)] \cdot [\lfloor 2\sqrt{p+1} \rfloor \ mod \ 2] - \lceil \frac{1}{2} l \rceil), \quad (19)$$

$$l = \lfloor \sqrt{p+1} \rfloor. \quad (20)$$

N chromosomes are selected from the union of the original population set and the rival population set ($2N$ chromosomes in total) according to the fitness values. Each chromosome can only be selected once. The chromosomes with larger fitness values will be picked up as the members of the new population set and go through the next iteration of the genetic evolution. Although the chromosomes have to be sorted in this survival competition stage, the overhead is not high because the population size is usually not large in the LGSA. In GA, the new chromosomes generated from the original ones are not guaranteed to have larger fitness values. The survival competition stage is included in the LGSA to prevent the chromosomes from being replaced by the new ones with poorer fitness values because the maximum generation number will be restricted to be quite small so as to cope with the tight time constraint.

The chromosome with the maximum fitness value is selected from the current population as a possible solution. The possible solution might be replaced by the other ones from one generation to the others. The iteration will be terminated when the termination conditions are satisfied. There are three termination conditions in the LGSA: (1) the possible solution is not updated for a predetermined period of generations; (2) the matching value of the possible solution is better than a predefined threshold; (3) the iteration number reaches the given maximum generation bound.

The computational cost of generating a random number is not low. In conventional GA-based applica-

tions, the search space is usually large and the time constraint is not an important issue, hence the computational cost required to generate random numbers is tolerable. However, when GAs are applied to time critical applications, such as the block matching of video coding, the cost of generating random numbers becomes a critical issue because the search space is relatively small and the time constraint is extremely tight. In conventional GAs, random numbers have to be generated in the reproduction, the crossover, and the mutation stages. Nevertheless, the random number generator is only called in the reproduction stage of the LGSA.

In conventional GAs, crossover is usually applied. The purpose of performing crossover is to randomly exploit new search points. Because the search space is not large in the LGSA, there are no large amounts of local optima in the search space. The effectiveness of crossover is not prominent; therefore, the crossover stage is not included in the LGSA for complexity reduction.

Because weaker chromosomes might propagate stronger chromosomes, they are not excluded in the new population set of the conventional GAs. The population will be gradually mature after a long period of generations. That is, weaker chromosomes will not hinder the population from being mature; however, they will bring large harms to the LGSA in which the generation number of evolution is restricted to be small. To solve this problem, in the LGSA, a survival competition stage is included. It ensures that the quality of each chromosome in the current population set is better than the old ones.

There are two kinds of mutation operators used in the traditional GA-based implementation: changing a gene's value (i) from 0 to 1, or (ii) from 1 to 0. Generally, the mutation probability is very low so as not to impair the overall quality of a given population. In the LGSA, the mutation probability is relatively high so the evolution of chromosomes is relatively violent. Fortunately, the bad effect of high mutation rate will be totally controlled by the survival competition. Interestingly, lots of search points will be explored due to high mutation rate although there is no crossover stage in the LGSA.

Because the evolution of chromosomes is slow, the maximum and the average generation numbers are large in conventional GAs, and so are the required average computations for finding the extreme value. Therefore, both the control overheads and the cost of performing extreme value finding are tremendous. In the LGSA, the evolution is relatively violent and the quality of chromosomes is well controlled by the survival competition stage, so the maximum generation number is small and the control overheads of chromosome evolution are also reduced. Moreover, the cost for extreme value finding of the LGSA is relatively small because most of the irrelevant search points have been excluded.

5 CONCLUSION

In this paper, some important issues of the genetic evolution, such as the efficiency of crossover and mutation operations and the global convergence property are analyzed. It follows from the analyses, when the search space is small, the efficiency of the crossover operation is not good enough for deserving the required computations. It is also hard to adjust the mutation probability to promote the efficiency of the conventional mutation operations. In the proposed LGSA, the computational complexity is well controlled by taking the characteristics of smaller search space into account.

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning.* Reading: Addison-Wesley, 1989.

[2] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, Dec. 1981, pp. 1799–1808.

[3] Keith Hung-Kei Chow and Ming L. Liou, "Genetic motion search algorithm for video compression," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 3, no. 6, Dec. 1993, pp. 440–445.

[4] In Kwon Kim and Rae-Hong Park, "Block matching algorithm using a genetic algorithm," in *SPIE Symposium on Visual Communications and Image Processing*, Taipei, Taiwan, May 1995, pp. 1545–1552.

[5] M. Wollborn, "Prototype prediction for colour update in object-based analysis-synthesis coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 3, June 1994, pp. 236–245.

[6] K. A. De Jong and W. M. Spears, "An analysis of the interacting roles of population size and crossover in genetic algorithms," in *Parallel Problem Solving from Nature*, H. P. Schwefel and R. Männer, Eds. Springer, Berlin and Heidelberg, 1991, pp. 38–47.

[7] D. E. Goldberg, "Sizing populations for serial and parallel genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms and Applications*, San Mateo, CA, 1989.

[8] J. J. Greffenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Systems Man and Cybernetics*, vol. 16, no. 1, 1986, pp. 122–128.

Table 1. *Mutation efficiency of the LGSA.*

Chromosome Leng.	Mut. Disrup. Rate	Var. of Mut. Disrup.	Mut. Invariance Rate
2	0.2963	0.0027	0.1111
3	0.1376	0.0017	0.0370
4	0.0661	0.0006	0.0123
5	0.0321	0.0002	0.0041

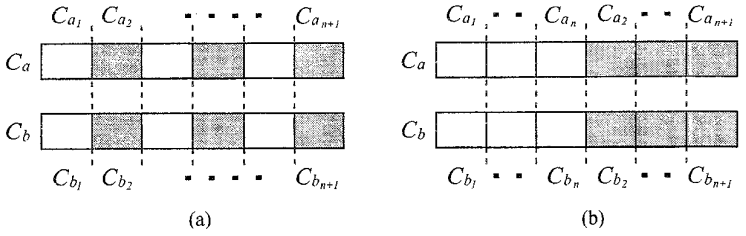

Figure 1. *(a) The N-point crossover, (b) an equivalent representation of an n-point crossover.*

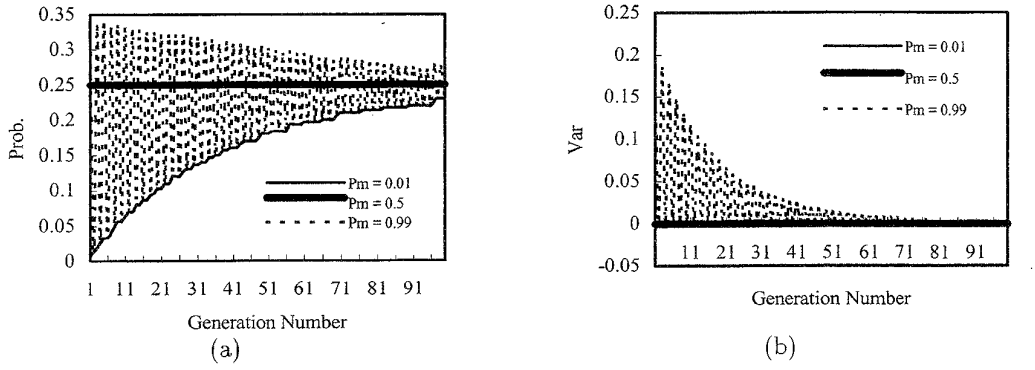

Figure 2. *Mutation efficiency of chromosome length $k = 2$: (a) the average mutation disruption rates, (b) the variances of the disruption rates for different chromosomes.*

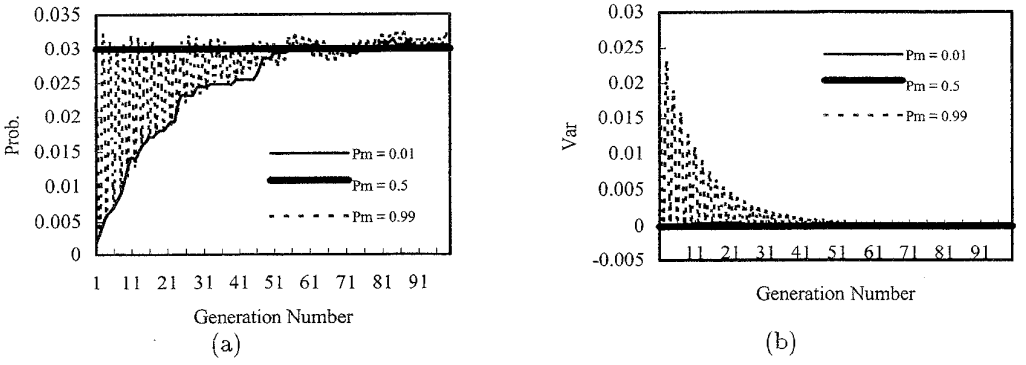

Figure 3. *Mutation efficiency of chromosome length $k = 5$: (a) the average mutation disruption rates, (b) the variances of the disruption rates for different chromosomes.*