2004

Crossover helps genetic algorithms in non-stationary environment

Yi Long
The University of Montana

Crossover Helps Genetic Algorithms In Non-Stationary Environment

by

Yi Long

B.E. Huazhong University of Science & Technology

presented in partial fulfillment of the requirements

for the degree of

Master of Arts

The University of Montana

May 2004

Approved by

[illegible]

Chairperson

[illegible]

Dean Graduate School

5-27-04

Date

Long, Yi M.S. May 2004 Computer Science

Crossover Helps Genetic Algorithms In Non-Stationary Environment

Director: Alden H. Wright [illegible]

Genetic Algorithms have been proved to be very useful for optimization. Until recent years, most of the research on Genetic Algorithms has been based on a stationary environment that assumes the fitness function will not change from generation to generation. Under a stationary environment, mutation can help to increase the diversity of the population, while crossover may help to keep advantageous alleles for the optimal solution. Re-initialization, adaptive mutation and memory reuse have been the most common genetic algorithm methods to handle optimization in a non-stationary environment. It is not clear what is the role of crossover in a non-stationary environment. In this thesis we will review previous work on GAs in a dynamic environment. We will also do empirical research on crossover's influence in a GA in a dynamic environment. Using a set of bit-strings as the population of a GA and designating a certain bit-string as an advantageous string with higher fitness, i.e. the NEEDLE, the Needle-In-The-Haystack (NEEDLE) fitness function will be used in the experiments while the environment change will be simulated by moving the NEEDLE periodically. Preliminary results clearly show that the crossover operation greatly helps a GA population to adapt itself to dynamic environment changes and moves the population to new optimums.

Acknowledgements

I would like to thank my professor Dr. Alden H. Wright, who has shepherded this work to completion. My gratitude goes deep for his sagacious initial inspiration for the topic of this work and effective guidance on every aspect of this work. His great patience encourages me to keep this work under difficulty situations. Without his advice and help, I could not possibly finish this paper.

The source code of the experiments conducted in this work was modified from the existing source code written by Dr. Alden H. Wright, Joseph D. Zeiler, and Jennifer Parham. Many thanks go to them.

I am grateful to the graduate committee member Jesse Johnson and George McRae. They took time from busy schedules to provide essential advice and information on this paper.

Table of Contents

List of Illustrations

Chapter 1 Previous Work

Since the emergence of genetic algorithms (GAs) in late 1960s, many efforts have been made to understand and utilize the generic mechanisms of biological evolution to improve optimization in computer systems. It is broadly accepted that a GA is a population-based search method that uses selection based on fitness, crossover and mutation. While all those elements are important in GAs, crossover is the key operation that sets GAs apart from other heuristic optimization methods such as hill climbing, evolution strategies, and evolutionary programming. But how crossover contributes to GAs' success is still not well understood. Schema theory [1] proposes that crossover helps to build highly fit schemata from lower order ones while mutation helps to keep the diversity of population. Others [2] think that the role of mutation has been underestimated by schema theory. The exact model [3] precisely describes a simple GA and it gives us more details on the effects of crossover and mutation.

1.1 Schema Theory

In Holland's classical schema theory [1], a schema is a set of bit-strings that can be represented by a string using the symbols 0's, 1's, and asterisks. An asterisk represents a "don't care" bit. While positions containing a 0 or 1 are fixed positions, all asterisk bits are variable positions. The schema 1*0* represents the set {1000, 1001, 1100, 1101}. The order of a schema is the number of fixed bits. For example the schema H = **1**10 is of order-3, or denoted as $o(H) = 3$. The defining length of a schema H is the distance between the first and last fixed bits. It can be denoted as $\delta(H) = 4$. Schemata help us to trace how GAs search for optimum solutions. By keeping and accumulating highly fit schemata while eliminating lower fit schemata, GAs can increase the frequency of highly fit individuals in the population. When majority of individuals in the population are the instances of the optimum schema, we can say the GA has converged the population to the optimum.

The GA's operations can be thought of as a search for schemata of higher than average fitness carried out by sampling individuals in a population and biasing future samples towards schemata that are estimated to have above average fitness. By considering the effects of the three GA operations: selection, crossover, and mutation, Holland's schema theorem gives the lower bound of the expected frequency of a schema after one generation of GA operations. Among the three GA operations, selection is the simplest one to calculate the effect. Let $p(x)$ be the proportion of string x in the population P and $f(x)$ be the fitness of string x. Proportional selection will make the expected number of string x in the population be: $p'(x) = \frac{f(x)p(x)}{\sum_{y \in P} f(y)p(y)}$. If x is an instance of schema H in the search space, then $f(H) = \sum_{x \in H} f(x)p(x)$ is the average fitness of schema H. Similarly, the average fitness of the population P is $f(P) = \sum_{y \in P} f(y)p(y)$.

Therefore after the proportional selection we get $p'(H) = \frac{f(H)}{f(P)}$. Hence it is clear that the selection has a linear effect on $p'(H)$. Schemata with higher fitness will have higher expected frequency in the population after selection operation. The crossover and mutation operations can both destroy and create instances of a schema H.

Now consider the one-point crossover operation with the applied rate c. Crossover will generate two children by exchange parts of the two parents delimited by this point. If the crossover point is in between of two fixed positions of a schema, the crossover may create two children not in the schema. For example, two parents 10 0, 11 1, with the crossover point between the second and third positions (note this point is also between the two fixed positions), two children will be 101 and 110. Though the first parent is of schema 1*0, neither of the children falls in the given schema. So we say that crossover has destroyed an instance of the schema. If the crossover point occurs in variable positions that are not between any two fixed positions, it will not destroy the instance of the schema. According to the previous reasoning, schema theorem gives the frequency of a schema H in a population after crossover operation with crossover rate c: $p'(H) \geq \left(1 - c\frac{\delta(H)}{\ell - 1}\right)p(H)$. Mutation will flip the bit at the randomly picked position. When mutation occurs in fixed positions, it will destroy the instance of the schema while mutations occurs in variable positions will not destroy this instance. Similarly, we can get the frequency of a schema H in a population after mutation operation with mutation rate μ : $p'(H) \geq (1-\mu)^{o(H)} p(H)$. Holland's schema theorem gives a lower bound of the next generation expected frequency of a schema based on the above consideration of the disruptive effects of crossover and mutation on a schema with crossover rate c and mutation rate μ $p'(H) \geq \frac{f(H)}{f(P)}\left(1 - c\frac{\delta(H)}{\ell - 1}\right)(1-\mu)^{o(H)}$. The theorem describes the effects of selection, crossover, and mutation in an intuitive way and helps people to understand how a GA works.

1.2 Exact Model of Simple GA

As mentioned earlier, the schema theorem considers the disruptive effects of the GA crossover and mutation operators. But to make the consideration simple and intuitive, it ignores the constructive effects of these operators. Therefore it only gives an inequality equation. With consideration of both construction and destruction of crossover and mutation, Vose (with helps from others) gives an exact model [3][4][5] that exactly models the GA's dynamics for infinite population. This model can be used to determine the exact expected frequency of any schema from one generation to the next. It is very helpful for understanding GA precisely but it cannot be applied computationally to most practical situations because of its computational complexity.

1.3 The Role of Crossover

In Holland's early work [1], he used only one-point crossover. Later, this operation was extended to n-point [6] and uniform crossover [7]. Compared to n-point crossover, uniform crossover does not have defining length bias but has more disruption and exploratory power [8]. While the disruption analysis of schema theorem suggests that disruption is not good for GAs and should be avoided, other people [8] think under certain circumstances, such as a homogeneous population or small population, disruption may actually be helpful to the GA by creating new individuals from parents with nearly identical genetic material. Both crossover and mutation have disruptive and constructive effects in GAs. How to understand and compare their roles in GAs? By calculating the probability of destroying an instance from a schema with crossover or mutation, Spears et al. [9] found that mutation can achieve any level of disruption that crossover possibly can achieve. On the other hand, disruption is not the only effect of crossover and mutation. They can also construct new instances of a schema from instances of other schemata. Measured by the probability of creating a new instance from crossover or mutation, crossover has a higher level of construction than that of mutation when the population is diverse (less than 70% ~ 80% convergence). When a population is mostly converged (more than 70% ~ 80%), crossover has lower level of construction than mutation. The definition of convergence they use is the average probability P_{eq} of any two schemata having the same allele at a particular fixed position. When P_{eq} is close to 1, the population has very low diversity. In a later paper [10], he used more sophisticated models to show that crossover and mutation interact in a more complicated way than the earlier disruption and construction theory.

Though the intuitive idea of fitness function is that the fitter schemata have more offspring, Stephens et al. [11] found that fitness landscape alone does not prevent the construction of low fitness schemata. They introduced the concept of effective fitness derived from the exact model showing that schemata of higher than average effective fitness receive an "exponentially" increasing number of trials over time. The intuitive idea behind effective fitness is that the schemata will have high effective fitness if there is high probability that their offspring have high fitness. Thus, effective fitness of schemata is not fixed to their own fitness but reflect the fitness of their offspring. Their schema equation is simple by introducing effective fitness: $p'(H) = \frac{f_{eff}(H)}{f(P)} p(H)$ where $f_{eff}(H)$ is the effective fitness of schema H. The $f_{eff}(H)$ is fairly complicated to compute. A simple example shows the effective fitness is more relevant: Let search space $\Omega = \{00,01,10,11\}$ with a fitness function $f(01) = f(10) = f(11) = 2$, $f(00) = 1$. For mutation rate μ, at $t = 0$, $f_{eff}(11) = 2$, $f_{eff}(01) = f_{eff}(10) = 2 - \mu$, $f_{eff}(00) = 1 + 2\mu$. Therefore, the effective fitness function provides a selective pressure by selecting the schemata having a higher probability to produce fit descendents. While the traditional schema theorem emphasizes the destructive effect of crossover, their model takes into account the reconstructive aspect of crossover as well. They also found from the model that generically there is no preference for short, low-order schemata. In the case where schema reconstruction is favored over schema destruction, large high-order schemata tend to be favored.

Suzuki et al. [12] used Babel-like fitness function (the same as the NEEDLE fitness function in this paper) to explore the role of crossover in GA. The definition of their Babel-like fitness function is that a single sequence denoted by [11...1] has fitness far larger than the others and all other sequences have the same fitness. In the context of this paper, the sequence [11...1] is the NEEDLE of the search space. They defined the domination time as the time until the advantageous string occupied half of the population. By breaking the domination time (T_d) into three parts: diversification time (T_y) (The population starts from homogeneous distribution), creation time (T_c) and spread time (T_s) as $T_d = T_y + T_c N_c + T_s$, they found that with a moderate mutation rate, crossover could greatly reduce the domination time. With the definition of acceleration rate of crossover A_{cross} as the ratio of T_d without crossover and T_d with crossover, A_{cross} can be 1 to 10,000 depending on different crossover, mutation, population and other parameter settings. They disagree with the Building Block Hypothesis's statement that crossover mainly recombines short low-order schemata into long high-order schemata. Instead, they claim that crossover helps not only to create novel schemata but also to combine created schemata into optimal sequence. However, the effectiveness of crossover needs the help of mutation to keep the diversity of population.

Geiringer's theorem [13] gives us the limit of crossover without selection and mutation. Linkage is the association of alleles at different loci in the population. When there is no association between alleles at any loci, the population is said in linkage equilibrium. According to Geiringer's theorem, without selection, the limit of crossover moves the population into linkage equilibrium. In other words, crossover does not change the distribution of alleles at any locus, it merely shuffles those alleles at each locus, therefore de-correlates the alleles at different loci.

While many efforts have been made to prove that crossover does help the GA to converge to optimal solution, additional work has revealed that it is not appropriate to expect the role of crossover has a simple positive or negative effect on a GA.

From Eigen's quasi-species model [14], we know given a population in a sequence space, with a low mutation rate, through either asexual replication (mutation without crossover) or sexual replication (mutation and crossover), the population will crowd together around the fittest sequence(s) (optimum(s)) with relatively small Hamming distance between any two individuals of the population. Such a population distribution is called "quasi-species". But when the mutation rate is too high, the population will lose its ability to cluster around the optimum. This disruptive mutation rate is called the "error threshold". Previous work [15,16] found that crossover lowers the error threshold therefore it has a very important role in determining the optimal mutation rate.

By doing numerical simulations based on the Vose exact model of [3], Wright et al. [17] found that crossover may keep a GA population from moving towards the optimum string with the NEEDLE fitness function (also known as Single-Peak Fitness Landscape in the literature). Crossover has a “catastrophic effect” when the mutation rate is around “error threshold”. The “catastrophic effect” refers the fact that GA loses the power to move the population to any optimum string. Under the “catastrophic effect” the population tends to random distribution. The crossover operation also decreases the error threshold of mutation rate (this is consistent with [16]). There are some very important conclusions from [17]:

- Crossover always pushes the error threshold lower and makes the population distribution sharper..
- The error thresholds are inversely proportional to chromosome length. The larger the bit string, the lower the error threshold.

One notable conclusion is that for NEEDLE fitness function, the crossover operation decreases the frequency of the optimum string (also known as master sequence in the literature), increases the frequency of the near-neighbors of the optimum string, and decreases the frequency of the strings far from the optimum string. Even though these experiments were done in a stationary environment, they could suggest experiments in a non-stationary environment. Because of the movement of the NEEDLE in non-stationary environment, a near neighbor of the optimum string may become a new NEEDLE and therefore they may help the survival of a population under environmental changes. The lowering of mutation "error threshold" by crossover may also help a GA population to survive environmental change because it indirectly magnifies the disruptive effect of mutation.

1.4 Non-Stationary Environment

Whenever there are some changes in a GA occur, such as the optimization goal, or the fitness function, we say the GA is in a dynamic or non-stationary environment. In dynamic optimization, when the environment changes, the optimum of the problem is likely changed also. This complicates the GA application. In this thesis, the definition of non-stationary environment or dynamic environment is that the optimum will change from time to time. In this thesis, the environment of the GA is the fitness function. Fitness function changes reflect the environmental changes.

There are many factors that have effects on the design of non-stationary GA application:

1. How can we detect/determine an environment change happening?
2. How often does the environment change?
3. How severe are the environment changes?

4. How predictable are the environment changes?

Because GAs are likely to push the population to converge to an optimum and lose the diversity, with the less diverse population, it is hard for GAs to get new optimum when the environment changes. Several approaches have been proposed to keep the GAs adaptive enough to follow the changed environment (and the new optimums)[18][19][20]. All methods mentioned here are based on the observation: traditional GAs tend to converge to an optimum. Thus once the population is dominated by advantageous schemata, the GA loses the ability to adapt to a change in the environment. Because the population has lost its diversity, it may be impossible for the GA to find new highly fit schemata in a changed environment.

The simplest approach is to restart the process whenever the environment changes [18]. But there are some difficulties with this approach:

1. Sometimes it is hard to detect when the environment changed;
2. Methods to detect the environment changes often do not fit into the evolutionary computation framework;
3. Even if we can keep track of the changes, restarting the process will lose all previous information and cause high computation cost. And it may make this approach impractical to frequently changed environment.

Other approaches try to modify one or more operations to keep the diversity of the population. The two most used operations are mutation and selection. Efforts can also be made to keep extra population information along with the dynamic environment changes.

The adaptive mutation (or hypermutation) method [19] is based on the observation: the disruptive effects of mutation can be used to increase the diversity of population when needed. By hiking the mutation rate whenever a change happens, this approach can help the GA move to a new optimum when the optimum changes. The difficulty here again is to detect when the environment changes.

Modifying selection is another approach to maintain diversity in non-stationary environment. Weaker selection can increase the diversity of the population. By adjusting the selection method to increase the diversity of the population during GA run. Another common approach is to use sharing [21], i.e., individuals in the same region of the environment share their fitness. Therefore individuals in less populated regions are favored over those of highly populated areas. This approach improves the GA's ability to track optima in slowly changing environments [22].

Another approach is to memorize or save some good schemata for reuse as necessary. But this approach needs to be implemented carefully to memorize the right information about the population while not increasing computational complexity too much. Implicitly we can use a redundant representation to memorize useful schemata. Or we can explicitly use extra memory to store and retrieve useful schemata. Both memory approaches are suit for periodically changing environment [23] because it is easier to define the "useful schemata". Its effectiveness in other non-stationary environments is still an open question [24].

So far, there has not been much discussion of crossover's effect on a non-stationary environment GA. I have not found any research on non-stationary environments that describes an attempt to keep diversity in the population solely by crossover.

Chapter 2 Objective

Understanding the role of crossover in GA is a key to utilizing GAs. It is of interest to examine crossover in dynamic environment. When a GA population adjusts itself to the environmental changes and keeps following the new optimums, we say the GA survives the dynamic environment. Otherwise, we say the GA fails in the dynamic environment. The ability of a GA to survive in a dynamic environment is a very important measure of GA performance. Based on conclusions from [12] and [17], it is intuitive to expect crossover to help a GA population to survive environmental changes. Therefore, the main objective of this thesis is to answer the question: Does crossover help a GA population to survive in a dynamic environment? If it does help the GA in dynamic environment, how does it help and to what extent?

As indicated by [17], crossover will lower the "error threshold" of mutation rate, so within certain range, we expect crossover will help a GA population to survive in small movement dynamic environment because crossover leverages the diversification effect of mutation. In other words, crossover lets the GA get the same disruptive effect with lower mutation rate to keep the necessary diversity of the population in dynamic environment. Because crossover increases near-neighbors of optimal string but decreases far-away strings, when mutation rate is low, i.e. within the "error threshold" boundaries, we expect that crossover helps the GA more in a situation with small environment changes than in a situation with large environment changes.

In dynamic environment, a GA is of little interest if the population can not converge to the new optimum in a timely manner. The ultimate goal of GA is to search for optimal solutions. Therefore to make GA a useful adaptive tool, we expect crossover helps GA populations not only to survive but also to converge to optimum in dynamic environment.

According to [12], crossover also helps the GA to converge to optimal string when the mutation rate is moderate (within error threshold). From the results of [12] and [17], there should be a range of mutation rates that crossover helps both the convergence and the survival of a GA population. We can call this range of mutation rates the optimal range of mutation rates. Within this range, we can expect a GA population will quickly adjust itself to environmental changes and converge to optimums in a timely manner. An intuitive conjecture is that the optimal mutation rate is lower than the error threshold (so the population will stay close to the optimum strings) but close to the error threshold (so the population will maintain necessary diversity).

The questions addressed in this thesis are:

1. What are the effects and effectiveness of crossover on different degrees of environmental change?
2. Does crossover help a GA population retain copies of the optimum in dynamic environments?
3. What are the effective ranges of parameters if crossover helps a GA population survival in dynamic environments?
4. Though survival is an important measure of GA's performance in dynamic environment, it is not the only gauge. The ultimate goal of a GA is to not only survive environmental changes but also to adaptively converge to optimal solutions quickly. Therefore, an effort is made to find the balance point (or optimal crossover and mutation rate range) for both good survival and quick convergence.

Chapter 3 Methodology

While there are different ways to simulate a dynamic environment, we need a method that satisfies the following requirements:

1. It should be easy to control the environmental changes, such as:
    - The time and frequency of the changes
    - The magnitude of the changes
2. It should be easy to watch the population during the process, such as:
    - The distribution of the population
    - The fitness of individuals
    - The overall fitness of the population

The search space of our experiment will be a set of binary strings of fixed-length of ℓ. By manipulating the bits of NEEDLE string, the dynamic environment can be easily simulated and controlled. With the NEEDLE fitness function defined below, such a dynamic environment satisfies all above requirements.

The NEEDLE fitness function is defined as:

$$f = \begin{cases} 1 + \alpha \text{ where } \alpha > 0\text{, the NEEDLE string} \\ 1\text{, all non-NEEDLE string} \end{cases}$$

There is only one NEEDLE string in the search space. Note that at any generation, the whole population may have multiple optimum strings that have the same bit sequence as the NEEDLE string or may not have any optimum string at all. When number of the optimum strings is more than 50% of the population we say the population has converged. When there is no optimum string in a population, we say this population has lost its NEEDLE. When a population loses its NEEDLE, it may need many generations to find the NEEDLE again.

We start the population with a random distribution. Let $\alpha = 1$ so the fitness of the NEEDLE is 2 and the fitness of all other strings is 1. The NEEDLE string is also created randomly. The periodic NEEDLE movement is done randomly by a Hamming distance h where h can be adjusted for different experiment purposes. The NEEDLE movement is a simulation of environment change. When the NEEDLE moves, the used-to-be-needles are not needles any more and the overall fitness of the population changes. The population needs to adapt to the new environment to find and converge to the new NEEDLE. Based on the claim of [17]: "Crossover ... increases the frequency of the near-neighbors of the optimum string, and decreases the frequency of the strings far from the optimum string", we have an intuitive conjecture that crossover might be more effective for small environmental changes. Experiments on different h values may help us verify this.

In a changing environment, a once good individual may turn to be a bad individual and vice versa. The commonly used version of the steady-state GA usually will mislead the fitness evaluation process because it uses a new individual to replace the worst individual in the population based on the previous evaluations. A once good individual may survive forever without being reevaluated in the new environment while a once bad individual may be replaced even though it is a very good one in the new environment. Therefore we use generational algorithm for the selection process. Because every individual's fitness will be evaluated every generation, there will be no individual fall through the evaluation cracks. The selection process can swiftly react to any environmental changes.

The main parameters we have are:

1. String length $\ell = 10$
2. Special fitness increment value for needle $\alpha = 1$
3. Hamming distance of needle movement $h = 1, 2$
4. Population size $p = 1000$
5. Generations between needle movement m : 10
6. Generations of a GA run *g*: 500
7. Repetitions of a GA run *r*: 100

The two experiments are the survival experiment and the convergence experiment.

The algorithm of the survival experiment can be described as following:

1. Start with a population of p randomly chosen length ℓ binary strings
2. Randomly specify the NEEDLE string
3. Test the population to count its optimum individuals. If there are optimum individuals, record this generation as a survival generation. Otherwise if this is the 10th consecutive generation that lacks the optimum individual, go to step 1 for a new round run
4. If the same NEEDLE has been run for m generations, change the NEEDLE by Hamming distance h
5. If this is the generation g, stop the GA run
6. Select two parents from the population proportionally based on individual fitness
7. Create the child from selected parents by uniform crossover with crossover rate c
8. Mutate the child according to the mutation rate μ
9. Add the child into the population of the new generation
10. Go to step 5 until the new generation is full-sized
11. Go to step 3

The survival experiment focuses on the survival cases to see how many generations a population can retain copies of the NEEDLE string in the changing environment. The experiment periodically moves the needle each 10 generations (To let the population get enough diversity from initialization, the first round is 20 generations for survival experiments). Whenever the population loses the needle for 10 consecutive generations, we say the GA fails in the survival run and we record its survival generations without the 10 lost generations. The experiment records the average survival generations from multiple repetition runs. By analyzing those records, we can clearly see whether crossover helps the GA to survive in a dynamic environment.

The algorithm of the convergence experiment can be described as following:

1. Start with a population of p randomly chosen length ℓ binary strings
2. Randomly specify the NEEDLE string
3. Test the population to count its optimum individuals.
    - If there is no optimum individual, record this generation as a lost generation.
        i. If this is the 10th consecutive lost generation, record this round as a lost round then go to step 1 for a new round run.
    - Else if the number of optimum individuals is less than 50% of the population size, record this generation as a convergence generation.

- Otherwise the population has converged
    1. If the population has converged before the first needle movement, we do not count it as a valid run for the dynamic environment experiment. Go to step 1 for a new round run
    2. Else sum and record the total number of convergence generations then stop the GA

4. If the same NEEDLE has been run for m generations, change the NEEDLE by Hamming distance h
5. Select parents from the population proportionally based on individual fitness
6. Create the child from selected parents by uniform crossover with crossover rate c
7. Mutate the child according to the mutation rate μ
8. Add the child into the population of the new generation
9. Go to step 5 until new generation is full-sized
10. Go to step 3 to test this new generation

The convergence experiment focuses on the convergence cases to see how many generations a population needs to run before it converges. The experiment restarts a new round whenever the population converges 50%. Whenever the population has more than 50% individuals being the same as the needle, we say the GA has converged and if the convergence happens after the first needle movement, we record its convergence generations. The experiment records the average convergence generations from multiple repetition runs. The data tell us whether crossover helps the GA population to converge to new needles in dynamic environment.

Chapter 4 Results

With crossover rates from 0 to 100%, mutation rates from 0.0001 (10^{-4}) to 0.001 (10^{-3}), and needle movement of Hamming distance 1, crossover always increases the average survival generations (ASG) 2 to 5 times compared to a GA without crossover (crossover rate = 0). A notable phenomenon: when mutation rate is more than 0.0005, the increasing of crossover rate does not always increase the ASG while for lower mutation rates, higher crossover rates always give higher ASG. See Figure 1.

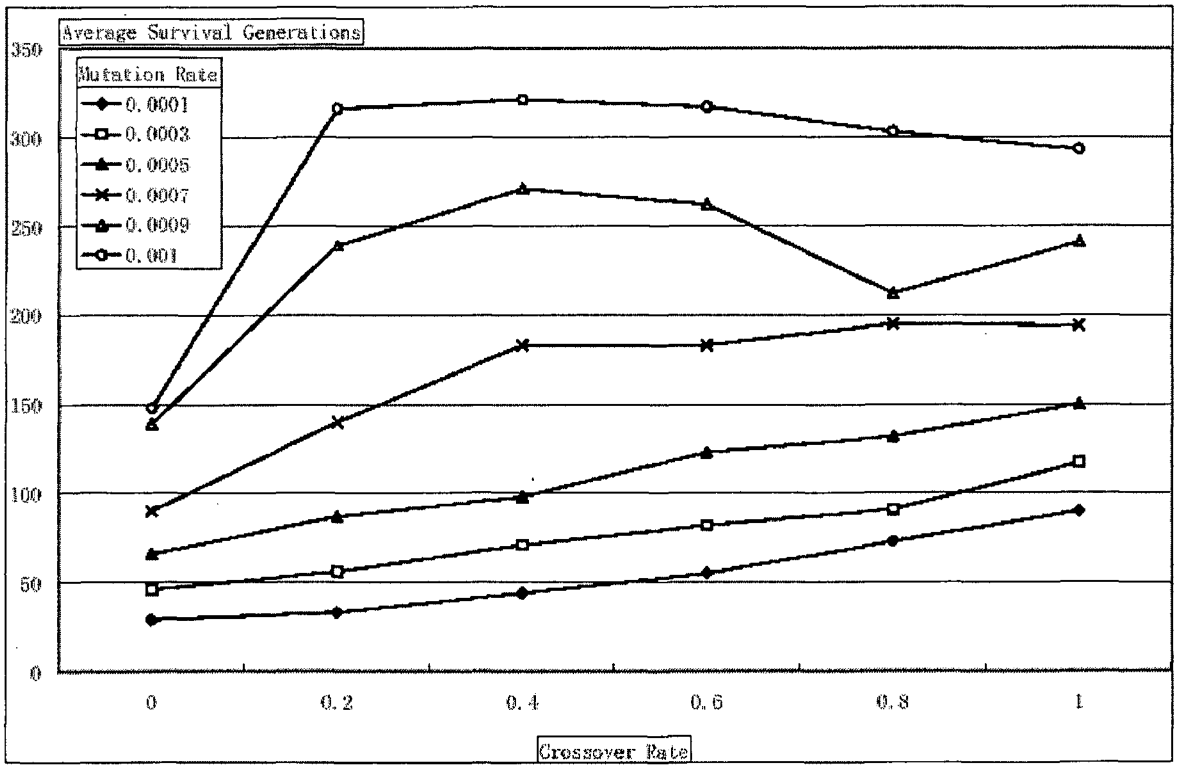

Figure 1 Average Survival Generations for needle movement of Hamming distance 1. Values are averaged over 100 repetitions of 500-generation runs.

For the same crossover and mutation ranges, bigger environmental changes (needle movement at Hamming distance of 2 in Figure 2) show that only higher crossover rates have real power to increase the ASG.

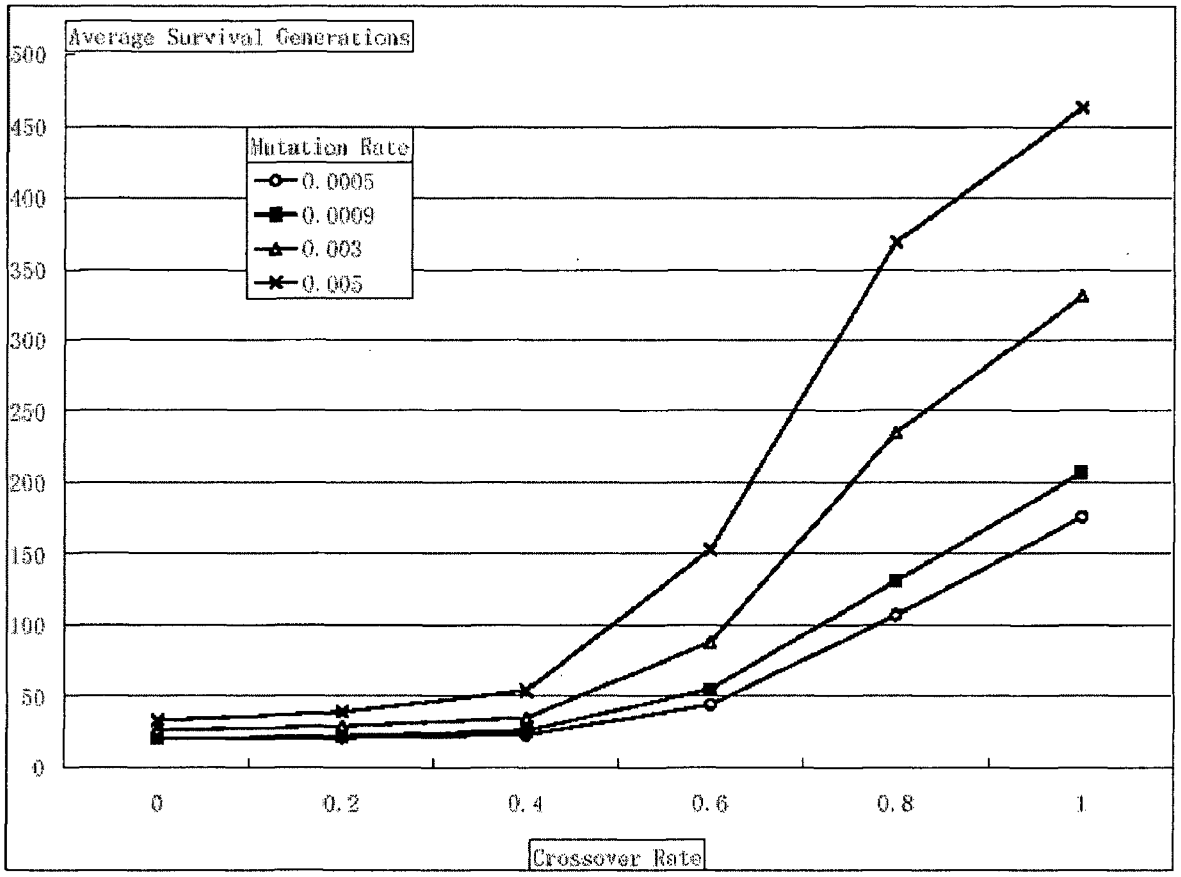

Figure 2 Average Survival Generations for needle movement of Hamming distance 2. Values are averaged over 100 repetitions of 500-generation runs. Note that for crossover rate less than 0.5, increments of ASG are not as effective as higher crossover rates.

With mutation rates from 0.0001 to 0.001, crossover helps the GA population to converge earlier than GA without crossover. But higher crossover rate (100%) does not always help the convergence. See Figure 3, 4. Interestingly, moderate crossover rates (40% ~ 60%) are more effective than higher rates. Note the effective mutation rate is much smaller than the error threshold calculated by the formula from [16]. This is consistent with conclusion of [17].

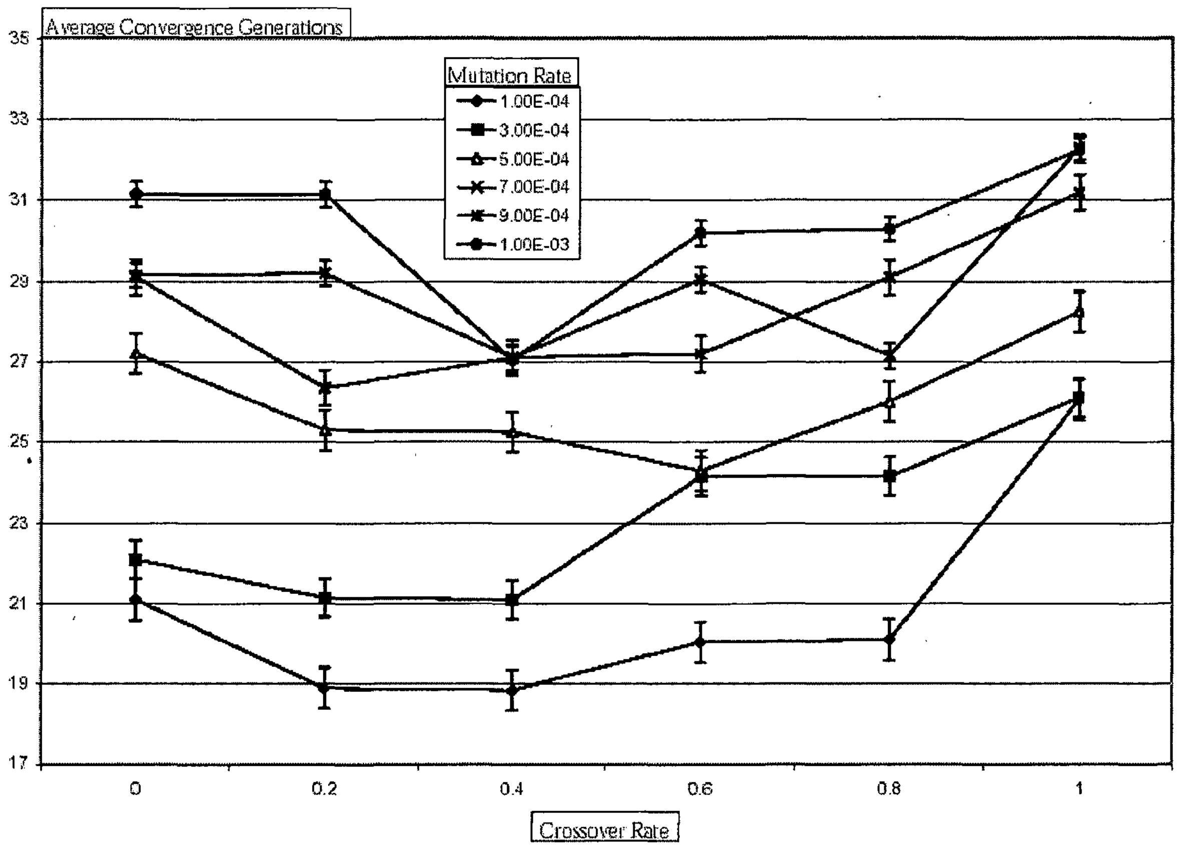

Figure 3 Average Convergence Generations for needle movement of Hamming distance 1. Values are averaged over 500 repeat runs. Error bars are the standard deviations from repeat runs. 0.30 < SD < 0.51. Note that crossover rate 1.0 does not help to decrease the ACG.

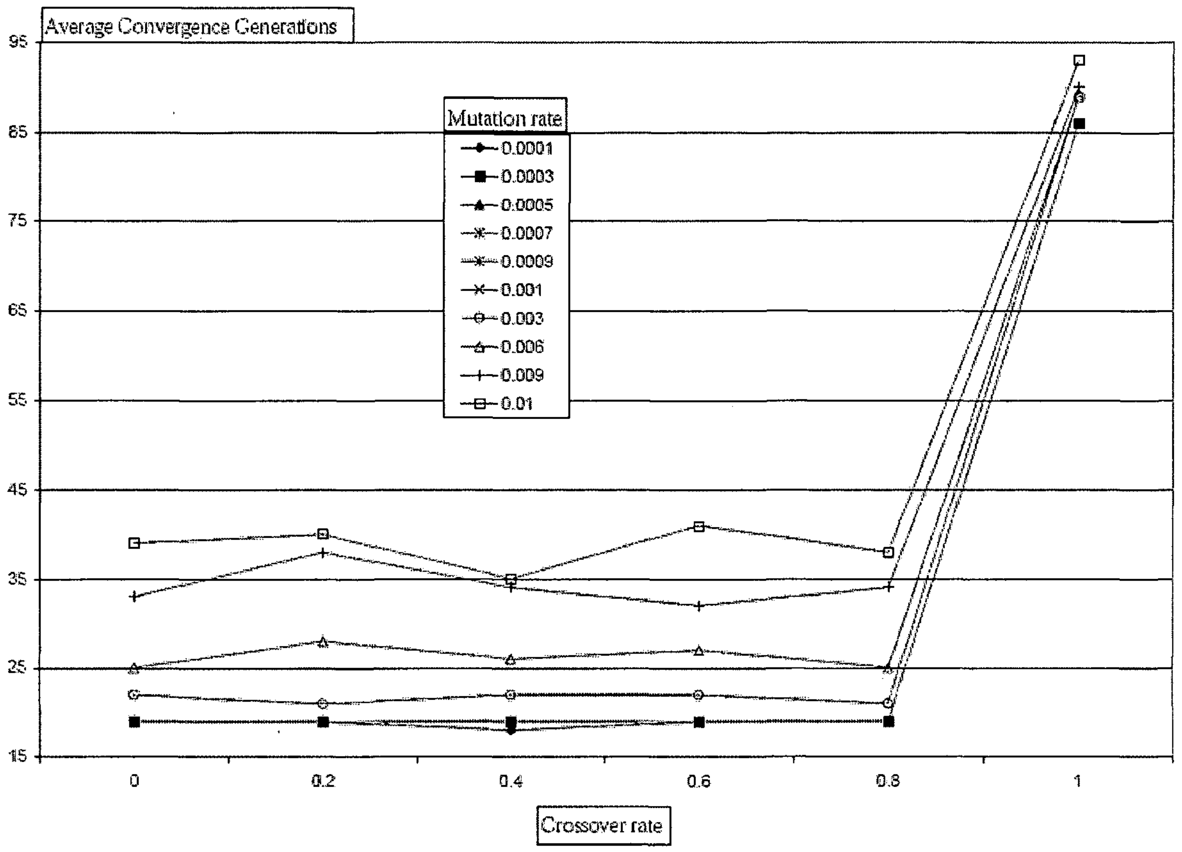

Figure 4 Average Convergence Generations for needle movement of Hamming distance 2. Values are averaged over 100 repeat runs. Note that for low mutation rates (0.0001 – 0.001) crossover has little impact on ACG so the lines are mostly overlapped to each other. Note that crossover rate 1.0 does not help to decrease the ACG.

Figure 3 shows that in the convergence experiments, a relatively low but nonzero crossover rates reduce the ACG more than higher crossover rates.

By comparing figure 3 and figure 4, we can find another interesting thing: for different NEEDLE movements, crossover helps GA convergence more for Hamming distance 1 than for Hamming distance 2. While crossover can shorten the ACG by 13% for Hamming distance 1, there is only insignificant improvement of ACG for Hamming distance 2. This confirms our intuitive conjecture based on

[17] that crossover might be more effective for smaller environmental changes. It will be an interesting experiment to extend the NEEDLE movement to other Hamming distances such as 3, 4, and so on.

Chapter 5 Conclusions

With the results from the previous chapter, we can come to some important conclusions regarding the role of crossover in a dynamic GA environment.

The experiments clearly demonstrate that the crossover operation helps a GA population both to survive environmental changes and to converge to new optimums in dynamic environment. There is an optimal crossover and mutation rate range in which both the survival ability and the convergence ability of a GA are strong. This is very important for crossover being useful in dynamic GA. But there is not a one-fits-all crossover rate. Different crossover rates suit different situations so it should be fine-tuned for specific GAs. Consistent with the conclusion of [17], the effective mutation rates are lower than the theoretical "error threshold" calculated by formula of [16].

Chapter 6 Future Work

This study verifies the intuitive conjectures made in chapter 2 that crossover helps dynamic GAs with low mutation rates for both survival and convergence experiments. However, this is just a first step to explore the role of crossover in dynamic GAs. There are several questions that worth future investigation:

1. The experiments cover a small set of parameters. Future work should explore a larger variety of parameter combinations. Following parameters are of special interest:
   - String length ℓ
   - Hamming distance of needle movement h
   - Population size p
   - Generations between needle movement m

   The experiments show that crossover is more effective to the environment changes of Hamming distance 1 than of Hamming distance 2. What will happen to other Hamming distances?
2. The work does not analyze the population distributions that may reveal more information of the course of dynamic GAs.
3. The interaction between crossover and mutation is still not clear in these experiments, special design of GA runs may help us to get more insight of this very important aspect.

4. Crossover helps dynamic GAs with the simple NEEDLE fitness function. Can similar behavior be repeated in dynamic GAs with other more complex fitness function landscapes?

References:

1 Holland, J. H. (1975). Adaption in Natural and Artificial Systems. Univ. Of Michigan Press, Ann Arbor.
2 Fogel, D. B. and Atmar, J.W. (1990). Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems. Biological Cybernetics 63: 111-114.
3 Vose, Michael D. (1999). The Simple Genetic Algorithm: foundations and theory. MIT Press.
4 Vose, Michael D. (1990). Formalizing Genetic Algorithms. Proc.IEEE wksp. On G.A.s, N.N.s, & S.A. applied to problems in Signal & Image Processing. May 1990, Galsgow, U.K.
5 Vose, Michael D. (1993). Modeling Simple Genetic Algorithms. Foundations of Genetic Algorithms – 2, D. Whitley Ed., Morgan Kaufmann.
6 De Jong, K. (1975). Analysis of the behavior of a class of genetic adaptive systems. Ph. D. thesis, University of Michigan, Ann Arbor.
7 Syswerda, G. (1989), Uniform Crossover in Genetic Algorithms. In International Conference on Genetic Algorithms, Volume 3, pp. 2-9. Morgan Kaufman.
8 Spears, W. & De Jong, K. A. (1991). An Analysis of Multi-point Crossover, Proceedings of the Foundations of Genetic Algorithms Workshop, G. Rawlins(ed.), Morgan Kaufmann Publishers.
9 Spears, W.M. (1992) Crossover or Mutation? In Whitley (ed.) Foundations of Genetic Algorithms-2. 221237.
10 Spears, W.M. (2000) The Equilibrium and Transient Behavior of Mutation and Recombination. FOGA 2000.
11 Stephens, C. R. and Waelbroeck, H. (1999). Schemata evolution and building blocks. Evolutionary Computation 7(2), 109--124.
12 Suzuki, Hideaki (1999) Crossover Accelerates Evolution in Gas with a Babel-like Fitness Landscape: Mathematical Analyses
13 Geiringer, H. (1944). On the probability of linkage in mendilian heredity. Annals of Mathematical Statistics, 15:25-57, 1944.
14 Eigen, M., McCaskill, J., and Schuster, P. (1988) Mocular quasi-species. J. Phys. Chem., 92:6881-6891, 1988
15 M. C. Boerlijst, S. Bonhoeffer, and M.A. Nowak. (1996) Viral quasi-species and recombination. Proc. R. Soc. London. B, 263:1577-1584, 1996
16 Ochoa, G. and Harvey, I. (1997) Recombination and error thresholds in finite populations. In Foundations of Genetics Algorithms 5, pages 245-264, San Mateo, 1997. Morgan Kaufmann.
17 Wright, Alden H., Rowe, Jonathan E. and Neil, James R. (2002) Analysis of the Simple Genetic Algorithm on the Single-peak and Double-peak Landscapes. Accepted for presentation at CEC 2002 and publication in the proceedings of this conference.

18 Grefenstette, J. J. and Ramsey, C. L. (1992) An approach to anytime learning. In D. Sleeman and P. Edwards, editors, Proceeding of the Ninth International conference on Machine Learning, pages 189 –195. Morgan Kaufmann, 1992.
19 Cobb, H. G. (1990) An investigation into the user of hypermutation as an adaptive operator in genetic algorithms having continuouis, time-dependent nonstationary environments. Tachnical Report AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.
20 Goldberg, D. E. and Smith, R. E. (1987) Nonstationary function optimization using genetic algorithms with dominance and dipliody. In Second International Conference on Genetic Algorithms, pages 59-68. Lawrence Erlbaum Associates, 1987.
21 Goldberg, D.E. and Richardson, J. (1987) Genetic algorithms with sharing for multimodal function optimization. In Second International Conference on Genetic Algorithms, pages 41-49, 1987.
22 Anderson, H.C. (1991) An investigation into genetic algorithms, and the relationship between speciation and the tracking of optima in dynamic functions. Honors thesis, Queensland University of Technology, Brisbane, Australia, November 1991.
23 Branke, J. (1999). Memory enhanced evolutionary algorithms for changing optimization problems. In Congress on Evolutionary Computation CEC99, volume 3, pages 1875-1882. IEEE, 1999.
24 Branke, Jürgen (1999). Memory enhanced evolutionary algorithms for changing optimization problems. Congress on Evolutionary Computation CEC99, volumn 3, pages 1875-1882. IEEE, 1999.